Modeling, approximation and simulation using smooth splines on unstructured meshes

Michelangelo Marsala

Modélisation, approximation et simulation à l'aide de splines régulières sur des maillages non structurés

Michelangelo MARSALA
Université Côte d'Azur, Inria AROMATH

Présentée en vue de l'obtention du grade de docteur en **Mathématiques** d'Université Côte d'Azur

Dirigée par: **Bernard MOURRAIN**
Codirigée par: **Angelos MANTZAFLARIS**

Soutenue le : 15 Décembre 2023

Devant le jury, composé de :
Carlotta GIANNELLI, Professeure associée, Università di Firenze, Florence
Carla MANNI, Professeure, Università di Roma "Tor Vergata", Rome
Angelos MANTZAFLARIS, Chargé de recherche, Inria d'Université Côte d'Azur, Sophia-Antipolis
Bernard MOURRAIN, Directeur de recherche, Inria d'Université Côte d'Azur, Sophia-Antipolis
Giancarlo SANGALLI, Professeur, Università di Pavia, Pavie
Thomas TAKACS, Chargé de recherche, Johann Radon Institute for Computational and Applied Mathematics, Linz

Modélisation, approximation et simulation à l'aide de splines régulières sur des maillages non structurés

---

Modeling, approximation and simulation using smooth splines on unstructured meshes

Michelangelo MARSALA

Jury

Rapporteurs:

- **Carlotta GIANNELLI**, Professeure associée, Università di Firenze, Florence
- **Thomas TAKACS**, Chargé de recherche, Johann Radon Institute for Computational and Applied Mathematics, Linz

Examinateurs:

- **Carla MANNI**, Professeure, Università di Roma "Tor Vergata", Rome
- **Giancarlo SANGALLI**, Professeur, Università di Pavia, Pavie

Directeurs:

- **Angelos MANTZAFLARIS**, Chargé de recherche, Inria d'Université Côte d'Azur, Sophia-Antipolis
- **Bernard MOURRAIN**, Directeur de recherche, Inria d'Université Côte d'Azur, Sophia-Antipolis

Abstract

In this thesis we investigate novel spline constructions over unstructured meshes to be applied for modeling purposes, approximation problems and in the numerical resolution of partial differential equations.

Being able to describe accurately a complex shape is not an easy task in geometric modeling, and it becomes even harder if we need the result to be suitable for numerical simulations. This is the challenge which motivated the topic of this work: explore new spline constructions that have potential to reproduce faithfully complicated geometries and, at the same time, are suitable to run isogeometric analysis experiments.

First we present the derivation of a globally G^1 smooth family of surfaces, defined by smoothing masks, approximating the well-known Catmull-Clark subdivision surface scheme. The resulting surface is a collection of Bézier patches, which are biquintic and join with G^1 smoothness around extraordinary vertices and bicubic elsewhere. Each Bézier point is computed using a locally defined mask which ensures, by means of quadratic gluing data, G^1 regularity around extraordinary vertices of the corresponding patches.

We continue with the description of a set of basis functions generating the space of biquintic G^1 spline over a quadrangular mesh. The basis is represented in terms of biquintic Bézier polynomials on each quadrilateral face. Starting from the equation defining the G^1 relations between two patches, achieved by quadratic gluing data functions, we perform an extraction procedure in order to obtain the values of the control point defining the different elements.

The latter basis functions, due to their definition, turn out not to be analysis-suitable. Driven by this, we investigate a new construction of G^1 spline basis functions with bidegree 5 suitable for isogeometric analysis simulations. The construction is made considering knot vectors composed of knots of multiplicity 5 and imposing C^1 regularity at the inner part of the resulting patches. Similarly to the earlier case, the coefficients defining the different functions are obtained by an extraction technique plus knot insertion.

The previous constructions are then used to solve two practical problems: the conversion of CAD models into smooth spline objects and the resolution of the shallow-water equation. The conversion is performed by fitting a point cloud representing a discretized model of the original CAD geometry, while the shallow-water equation is solved in the case of shallow lakes whose shape is faithfully approximated by a planar quad mesh.

Lastly, we present the definition of three cubic C^2 quasi-interpolation operators over arbitrary triangulations. The quasi-interpolants are locally generated by simplex spline basis functions defined on each triangle of the triangulation, which is subdivided according to the Wang-Shi split. The coefficients defining the operators in the simplex basis are computed easily by solving an Hermite problem, whose differential data is either given in input or reconstructed by using local cubic polynomials attached to the different features of the triangulation.

Key words: spline constructions, multipatch domains, gluing data, extraordinary vertices, isogeometric analysis, approximation theory

Résumé

Dans cette thèse, nous étudions de nouvelles constructions de splines sur des maillages non structurés à appliquer à des fins de modélisation, à des problèmes d'approximation et à la résolution numérique d'équations aux dérivées partielles.

Pouvoir décrire avec précision une forme complexe n'est pas une tâche facile en modélisation géométrique, et cela devient encore plus difficile si nous avons besoin que le résultat soit adapté aux simulations numériques. C'est le défi qui a motivé le sujet de ce travail : explorer de nouvelles constructions de splines qui ont le potentiel de reproduire fidèlement des géométries compliquées et qui, en même temps, sont adaptées à l'exécution d'expériences d'analyse isogéométrique.

Nous présentons tout d'abord la dérivation d'une famille de surfaces globalement G^1 lisses, définies par des masques de lissage, approchant le schéma bien connu de la surface de subdivision de Catmull-Clark. La surface résultante est une collection de points de Bézier, qui sont biquintiques G^1 autour de sommets extraordinaires et bicubiques ailleurs. Chaque point de Bézier est calculé à l'aide d'un masque défini localement qui assure, au moyen de données de collage quadratique, la régularité G^1 autour des sommets extraordinaires des patchs correspondants.

Nous poursuivons avec la description d'un ensemble de bases générant l'espace des splines G^1 biquintiques sur une maille quadrangulaire. La base est représentée en termes de polynômes de Bézier biquintiques sur chaque face du quadrilatère. sur chaque face du quadrilatère. A partir de l'équation définissant les relations G^1 entre deux patchs, obtenue par des fonctions de données de collage quadratique, nous effectuons une procédure d'extraction afin d'obtenir les valeurs du point de contrôle définissant les différentes bases.

Ces dernières bases, en raison de leur définition, s'avèrent ne pas être adaptées à l'analyse. C'est pourquoi nous étudions une nouvelle construction de fonctions de base spline G^1 avec un bidegree 5 convenant aux simulations d'analyse isogéométrique. La construction est réalisée en considérant des vecteurs de nœuds composés de nœuds de multiplicité 5 et en imposant une régularité C^1 dans la partie interne des patchs résultants. Comme pour la construction des bases précédentes, les coefficients définissant les différentes fonctions sont obtenus par une technique d'extraction et d'insertion de nœuds.

Les constructions précédentes sont ensuite utilisées pour résoudre deux problèmes pratiques: la conversion de modèles CAD en objets splines lisses et la résolution de l'équation des eaux peu profondes. La conversion est obtenue en ajustant un nuage de points représentant un modèle discrétisé de la géométrie CAD originale, tandis que l'équation des eaux peu profondes est résolue dans le cas de lacs peu profonds dont la forme est fidèlement approximée par un maillage quadratique planaire.

Enfin, nous présentons la définition de trois opérateurs quasi-interpolants cubiques C^2 sur des triangulations arbitraires. Les quasi-interpolants sont générés localement par des fonctions de base spline simplex définies sur chaque triangle de la triangulation, qui est subdivisée selon la division de Wang-Shi. Les coefficients définissant les quasi-interpolants dans la base simplex

sont calculés facilement en résolvant un problème d'Hermite, dont les données différentielles sont soit données en entrée, soit reconstruites en utilisant des polynômes cubiques locaux attachés aux différentes caractéristiques de la triangulation.

Mots-clés: constructions de splines, domaines multipatch, données de collage, vertices extraordinaires, analyse isogéométrique, théorie de l'approximation

Acknowledgements

This PhD thesis has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 860843.

Contents

Introduction

Describing accurately a complex shape in terms of its main geometric features is a major challenge in geometric modeling. The latter aims at providing compact and efficient models for approximating or representing precisely geometric objects, favoring the exploitation of such models in simulation and optimization processes.

In this thesis, we investigate the construction of new smooth spline spaces over meshes and their applications in geometric modeling, approximation theory and IsoGeometric Analysis (IGA)-based numerical simulations.

Spline representations make a step towards this objective, by describing smooth shapes in terms of control points, which express the features of such parametrized surfaces. Thanks to their simple definition and malleability, splines are applied in a wide range of different topics; one of their main usage can be found in Computer-aided Design and Manufacturing (CAD-CAM) where, typically, surfaces are trimmed and glued together in patchworks to establish the entirety of a geometric object's shape. As a consequence, shape descriptions emerge that might lack precision, showing leaking patches, stretched components, and other deviations that don't align with the desired geometry of the objects.

An alternative approach explored to tackle this challenge is the utilization of subdivision surfaces. Beginning with a coarse mesh that governs the surface's geometric characteristics, a subdivision process is sequentially employed to achieve progressively finer meshes, ultimately converging toward a limit surface. Subdivision creates smooths objects but, unfortunately, they correspond to an infinite set of spline rings nearby an Extraordinary Vertex (EV) of the initial mesh. Moreover, the surface exhibits unwanted oscillations around EVs.

As another option to model smooth spline objects over meshes of any topology we have multipatch constructions. In this approach, the sought geometry is defined as a collection of single spline functions joining each other with a prescribed smoothness. The tangent plane continuity, or G^1 continuity, extensively investigated in the past few decades, is a proper example of commonly utilized smoothness in multipatch geometries.

Other types of regularity, e.g. standard C^r, are more feasible when dealing with triangulations where the continuity can be simply achieved by using macro-elements split. This approach is common in the construction of quasi-interpolation operators or B-spline-like bases.

The investigation of novel smooth spline constructions over unstructured meshes is a revived topic due to its importance for isogeometric analysis discretizations. In fact, high quality results can be achieved by using G^1 basis functions when running numerical simulations to solve partial differential equations, also when dealing with challenging high-order equations.

Brief history of splines

From their first appearance back in 1946 in Schoenberg's paper [Sch46], spline constructions have raised great interest for their simple formulation and wide spectrum of applications. In the early 1960s, Bézier, thanks to Bernstein's work [Ber12], defined the homonymous curves and applied them for CAD at French automaker Renault; around the same years de Casteljau,

employee of Citroën, developed a numerically stable algorithm for evaluating the above curves.

In [CS66] splines were described as linear combination of "fundamental spline functions"; these functions, subsequently renamed B-splines, were deeply investigated, for example, in [Mar70; Boo72]. During these years there were already some results for tensor-product splines, but their most innovative application at that time was in the context of subdivision surfaces early introduced by Catmull and Clark in [CC78], giving rise to a new fruitful line of research in CAD.

Also, further developments on triangular domains made their entrance: among the many works concerning the development on this topic we cite [Fre71; PS77; Far79]. The age from 1980 to late 1990s had as main character multivariate spline constructions and their applications. A novel construction dated in those years were box-splines, about which we mention, for instance, [Boe84; CLR84] and the classic monographs [BHR93]; moreover, in [Loo87] Loop presented an innovative subdivision scheme whose limit surface is a box-spline.

The so-called *geometric continuity* began to be utilized in computer graphics, thanks to new works of the time such as [Far82], as well as innovative spline-like constructions of which we quote [Pot93; Rei97].

Until then, geometric modeling and numerical simulations had little in common. In fact, engineers were implementing finite element methods for solving partial differential equations approximating the domain by polygonal meshes. Before 2005 there were already few works proposing numerical treatments of PDEs based on spline constructions, but the seminal paper proposing to unify the two can be identified with the work by Hughes, Cottrell and Bazilevs [HCB05], in which, for the first time, it was rigorously described how to use CAGD tools to efficiently solve differentials problems. Undoubdably this is, to date, one of the most prolific research area in numerical analysis: isogeometric analysis was born.

Context of the thesis

Dealing with smooth objects having complex geometry is not an easy task. One of the mostly used tools to perform geometric modeling are tensor-product B-splines. These functions enjoy good properties, such as local support, nonnegativity, partition of unity; moreover, they form a basis for the space of piecewise polynomials, on a certain rectangular region of the plane, with fixed maximal degree d. Similar constructions presenting analogous properties to the tensor-product case can be achieved on triangular partitions of the plane by using the so-called triangular Bernstein polynomials (see, e.g., [Far86; LS07]). Thanks to the interesting characteristics of these functions, they are widely involved in geometric modeling: in fact, in order to reproduce an object in terms of spline surfaces it is only necessary to trace out a piecewise linear approximation of the sought geometry, defining its control net. Nowadays, the literature presents high number of extensions of B-spline constructions in order to satisfy the most varied applications: among all the works we recall Hierarchical B-splines [Kra97; BC13], Generalized B-spline [KS99; MPS11], T-splines [Sed03; TMH21], Tchebycheffian B-spline [Maz04; RMS23], Truncated Hierarchical B-splines (THB) [GJS12; Eva18] and Locally refined B-splines (LR) [DLP13; Bre13]. Many studies have also been focusing on analysing spline spaces, providing dimension formulas (or upper bounds for the latter) and basis computations. A recurring problem in applications is whether using constructions with high degree, which provide more freedom feasible to obtain smoother results but slower to evaluate numerically, or lower degree splines which are more computationally friendly but allow less flexibility. A possible trade off is the use of spline objects with inner

knots or opt for multipatch constructions.

In the case of planar triangulations, high smoothness spline constructions of relatively low degree can be achieved by using the so-called *macro-structures*. Famous examples are the Clough-Tocher split [CT65; LS07; Sab85] and the Powell-Sabin 6 and 12 splits [AS02; PS77; LS07; SS06]; of common interest are spline quasi-interpolants construction over such domains as [GS18; Spe13a; Spe13b; MS07]. Other spline quasi-interpolation operators are presented, for example, in [Bar08; DRS13; LMS08; Buf16], while hierarchical spline variants have been proposed in [Bra16; GJM20; SM16].

An excellent alternative to overcome this degree-regularity dispute is also provided by *subdivision surfaces*. A subdivision scheme is an iterative algorithm that produces, "at infinity", smooth objects called subdivision surfaces, when applied to an input coarse mesh of general topology. The smoothness of the limit surface depends on the weights defining the scheme itself and it is analyzed carrying out a spectral analysis on the *subdivision matrix*, i.e. a matrix containing the weights of the scheme. It is a common issue in subdivision to have a drop of regularity in the subdivision surface nearby the so-called *extraordinary* or *irregular vertices*; moreover, in the vicinity of the above extraordinary points, the subdivision surface cannot be represented using a finite number of B-spline patches. The first, and well-known subdivision scheme, is the Catmull-Clark scheme [CC78] acting on quadrilateral meshes, while we have the Loop scheme [Loo87] as initiator for subdivision on triangular meshes: both limit surfaces are C^2 continuous in regular regions and C^1 while approaching the irregular vertices, and they can both be directly evaluated thanks to elegant closed formulas presented in [Sta98; Sta01]. In recent years, a new class of subdivision schemes, called *guided subdivision*, has been published, in which the input mesh is used as guide to construct very smooth limit surfaces; in fact, this approach allows to get C^2 continuous limit surface around extraordinary vertices. Some of the first results in this respect are [KP17a; KP17b] and [KP23]. However, also in these situations the final surface has no finite description in terms of B-spline functions. To overcome this problem one could choose constructions which approximate the limit surface of a certain subdivision scheme with one or multiple B-spline patches: examples are [Li11] regarding the Loop subdivision and the works [Pet00; LS08] or the novel construction in Chapter 2 of this thesis, where smooth approximations of the Catmull-Clark limit surface are provided.

Multipatch approach proves to be an excellent allied when dealing with complex geometries. In these constructions the shape is obtained as a collection of several different spline patches of possibly different degree and topology by means of the so-called *geometry map*. An interesting advantage of this structure is producing watertight objects of complicated shapes without resorting, for example, to trimming. On the other hand, if we require the multipatch geometry to be suitable for numerical simulations, we need to impose high regularity between its adjacent patches. If we look for C^1 constructions, among the several existing works we cite [NP16; TSH17] and references therein, where bicubic constructions are provided for unstructured quad meshes; moreover, the latter also provide basis functions extraction to be applied for isogeometric simulations when dealing with T-splines. Regarding basis computations on general quad meshes, a problem that arises is the dimensional dependence from the geometry. As shown, for instance, in [Kap15], the dimension of a C^1 spline space depends on the geometry itself; this is a big limitation we would like to avoid.

To overcome this issue, one could go for other types of regularity such as the tangent plane continuity (or geometric continuity, or G^1 continuity). G^1 smoothness is easier to enforce than C^1 regularity when dealing with extraordinary vertices, and it returns comparable results with

respect to C^1 constructions in both visual quality and performances in numerical simulations. As suggested by the name, this continuity ensures that neighbouring patches share the same tangent plane when crossing their common edge. This result is achieved by using C^1 functions called *gluing functions*, which are defined along the edges of the considered geometry and solely depend on the valences of the two vertices identifying the edge; we have examples of different gluing data functions in [BMV17; BMX20; BH14].

Representing objects of complex shapes is common in CAD industry. Unfortunately, CAD representations present imperfections in the geometry which are unpleasant: in fact, such objects might present trimmed faces as well as gaps between neighbouring patches, making them non suitable for numerical experiments. In real applications, having proper representations of smooth objects which are also eligible for simulations is a must nowadays: that's a reason why one of the most prolific research topic in multipatch geometries is about isogeometric analysis suitable constructions.

From the seminal paper [HCB05] dated back in 2005 and the subsequent monograph [CHB09], isogeometric analysis has gained great interest. IGA is an efficient Galerkin-based method to numerically solve partial differential equations whose main idea lays on the *isogeometric paradigm*: it consists in reproducing in an exact way the domain of interest for our simulation via spline functions and use the same basis utilized in the representation of the domain to solve the target partial differential equation. Many papers provide challenging improvements in efficiency when compared to standard Finite Elements approaches, where the solution is obtained by using a set of local polynomial bases defined on a discretization of the selected domain. Particularly interesting are the constructions of G^1-smooth B-spline basis functions to run numerical simulations on challenging multipatch geometries, that is the case when dealing with real-life applications. It is impossible to briefly give the proper credit to all the existing works on the topic; among all we refer to [JQ14; Wan18; ZQ19] for analysis-suitable G^1 bases on triangular meshes and to [CST16; KST18; KTC22; TT23] for basis functions defined over quadrilateral meshes. We also mention [Hug21] for a more general description of multipatch discretizations and [Ver23] for a comparison of smooth basis constructions. Adaptive constructions on multipatch domains are also available, e.g. [BG15; Bra20; Bra23a; Bra23b], where the basis functions are locally refined around particular features of the considered geometry. Moreover, regarding G^1 spline basis functions on meshes with mixed topology we mention [MVV16; Gro24]. Lately, isogeometric analysis has been also used to solve partial differential equations using the so-called Boundary Element Method (BEM), where the search for a solution of a given equation in a certain domain is translated into some integral concerning only the boundary of the domain of interest. This approach gives rise to the IGA-BEM method, which can be more efficient than the standard approach in certain situations; examples are the works [Hel17; Kos18; Fal18] and references therein.

Motivated by the previous aspects, the objective of this thesis is to present novel smooth spline constructions on unstructured meshes that are both easy to construct and manipulate, and that produces high quality objects approximating complex shapes. We reach this goal by using multipatch spline geometries, whose control points are obtained from simple closed formulas ensuring smooth connections between patches. Being also interested in numerical simulations, we focus our interest on constructions providing spline objects that are also suitable for IGA-based numerical resolution of partial differential equations, i.e. possessing good approximation properties. In particular, this has been achieved investigating the construction of smooth basis functions that generate the sought spline space.

Overview

This thesis is organized in the following chapters:

- **Chapter 1** recalls basics concepts and results which are necessary for the description of the novel constructions of this work; it also introduces the notation we will use throughout the thesis. In detail, the notions of B-spline curve and surface are introduced as well as a multivariate generalization given by simplex spline formulation. It also contains an overview on subdivision surfaces and on the numerical treatments of PDEs via isogeometric analysis approach.

- **Chapter 2** is dedicated to the explicit construction of a G^1 Bézier surface obtained by means of smoothing masks on a quadrilateral mesh. After presenting the constraints ensuring the sought regularity, an explicit case-by-case solving strategy is deeply investigates. The chapter ends with several numerical experiments that demonstrate the quality of the novel proposal.

- **Chapter 3** is devoted to the description and study of a set of basis functions generating the space of G^1 splines over a quad mesh. Following the topology of the given mesh, the different functions are obtained by solving simple linear systems derived from the equations defining the geometric continuity across two adjacent patches. Various numerical experiments confirm their power in point cloud fitting problems.

- **Chapter 4** introduces an analysis-suitable extension of the basis functions developed in Chapter 3. This new formulation allows a space refinement through knot insertion which makes the construction suitable for the numerical solution of partial differential equations, as stated by the presented results.

- **Chapter 5** gives a practical application of the constructions introduced in Chapters 2 to 4: it concerns a complete pipeline to convert CAD models to G^1 spline objects. The translation is achieved by fitting a point cloud, extracted from the input CAD model, with the G^1 basis functions we propose; this conversion also makes the resulting geometry suitable for numerical simulations. Several computational tests confirm the above.

- **Chapter 6** presents another use of the tools developed in Chapters 2 to 4 regarding the free vibrations of a shallow lake. The equation strongly depends on the geometry of the lake's bathymetry, which we accurately reproduce by fitting a point cloud underlying its shape thanks to our G^1 basis functions. The cases of two real lakes are used to show the efficiency of this approach.

- **Chapter 7** contains the constructions of three cubic C^2 quasi-interpolation operators defined over a general triangulation. The quasi-interpolants are locally spanned by a simplex spline basis obtained by subdividing each triangle according to the Wang-Shi split, and their associated linear functionals are simply derived solving an Hermite problem. The approximation quality of the different operators is shown by the proposed numerical examples.

Publications

The contributions of the thesis are based on the following works:

- [MMM22] **G^1-Smooth biquintic approximation of Catmull-Clark subdivision surfaces**, with Angelos Mantzaflaris and Bernard Mourrain, in: Computer Aided Geometric Design. https://doi.org/10.1016/j.cagd.2022.102158.

- [MMM24] **G^1 spline functions for point cloud fitting**, with Angelos Mantzaflaris and Bernard Mourrain, in: Applied Mathematics and Computation. https://doi.org/10.1016/j.amc.2023.128279.

- [MMM23] **Analysis-suitable G^1 bases on quadrilateral meshes**, with Angelos Mantzaflaris and Bernard Mourrain, in preparation.

- [Mar23] **From CAD to representations suitable for isogeometric analysis: a complete pipeline**, with Angelos Mantzaflaris, Bernard Mourrain, Sam Whyman and Mark Gammon, preprint. https://hal.science/hal-04185850.

- [MMS23] **Maximally smooth cubic spline quasi-interpolants on arbitrary triangulations**, with Carla Manni and Hendrik Speleers, submitted for publication.

Implementations

The results present in this thesis have been obtained by using the following languages/libraries:

- The construction of the G^1 surfaces, as well as the basis functions, were obtained using Julia language [Bez17]. The developed codes are available in the repository https://gitlab.inria.fr/AlgebraicGeometricModeling/G1Splines.jl.

- The least squares fitting and the IGA simulations have been carried out using the G+Smo library [Man20; Lan15; Jüt14]. The developed codes are available at https://github.com/gismo/gismo.

- The quasi-interpolation operators have been implemented in MATLAB language [MAT20].

Chapter 1
Preliminaries

This chapter is dedicated to the basic concepts in the fields of approximation theory, Computer Aided Geometric Design and numerical simulation of partial differential equations. We start introducing B-spline basis functions, as well as B-spline curves, and then move on to the definition of bivariate tensor-product B-spline. Fundamental geometric algorithms to deal with B-spline object will be also provided, e.g. the knot insertion and the degree elevation algorithm. Later on, making use of the idea of geometric continuity, spline constructions on unstructured domains like quadrilateral mesh or triangulation are presented. Subdivision schemes, a powerful tool to reconstruct smooth surfaces starting from a coarse mesh, are also introduced together with an overview of classical schemes in the literature. Finally, in the context of numerical simulation of partial differential equation, the concepts of isogeometric analysis will be introduced.

1.1 B-spline functions

A basis spline function, better known as B-spline, is a piecewise polynomial function of a certain degree whose polynomial pieces join in order to achieve the desired global regularity. There are several ways to define a B-spline, e.g. truncated power functions or blossoming; here we use the Cox-de Boor recurrence formula, which also gives an efficient algorithm for their implementation. We refer the reader to [Boo78; PT95; PBP02] for further details and proofs of the properties presented in this subsection.

1.1.1 Definitions and properties

Let $\mathbf{t} = \{t_0, \dots, t_m\}$ be a sequence of $m+1$ non-decreasing numbers, i.e. $t_i \leq t_{i+1}$, $i = 0, \dots, m$, called *knot vector* and whose elements are referred as *knots*. The i-th B-spline of degree d $N_{i,d}$ is defined as

$$N_{i,d}(t) = \frac{t - t_i}{t_{i+d} - t_i} N_{i,d-1}(t) + \frac{t_{i+d+1} - t}{t_{i+d+1} - ti + 1} N_{i+1,d-1}(t), \tag{1.1}$$

where

$$N_{i,0}(t) = \begin{cases} 1 & \text{if } \ t_i \leq t \leq t_{i+1} \ , \\ 0 & \text{otherwise.} \end{cases}$$

The quotients in (1.1) may happen to be of the form $0/0$, which we define to be 0. For brevity we often write $N_{i,d}$ instead of $N_{i,d}(t)$. Figures 1.1 and 1.2 present a set of quadratic and cubic B-spline basis obtained from an uniform and non-uniform knot vector, respectively.

B-spline functions enjoy interesting properties:

- *Local support*: $N_{i,d}(t) = 0$ if $t \notin [t_i, t_{i+d+1})$,
- *Nonnegativity*: $N_{i,d}(t) \geq 0 \; \forall\, i, d, t$,
- *Partition of unity*: for an arbitrary *knot span*, i.e. the interval $[t_j, t_{j+1})$, we have

$$\sum_{\ell=j-d}^{j} N_{\ell,d}(t) = 1 \quad \forall\, t \in [t_j, t_{j+1}),$$

- In a given knot span $[t_j, t_{j+1})$ we have at most $d+1$ non-zero B-spline basis functions, namely $N_{j-d,d}, \ldots, N_{j,d}$,
- Except for the case $d = 0$, $N_{i,d}(t)$ reaches exactly one maximum value,
- *Derivatives formula*: the k-th derivatives of a basis function is given by

$$N_{i,d}^{(k)}(t) = \frac{d}{t_{i+d} - t_i} N_{i,d-1}^{(k-1)}(t) - \frac{d}{t_{i+d+1} - t_{i+1}} N_{i+1,d-1}^{(k-1)}(t), \tag{1.2}$$

- At a knot $N_{i,d}(t) \in C^{d-\mu}$, where μ is the multiplicity of the knot.

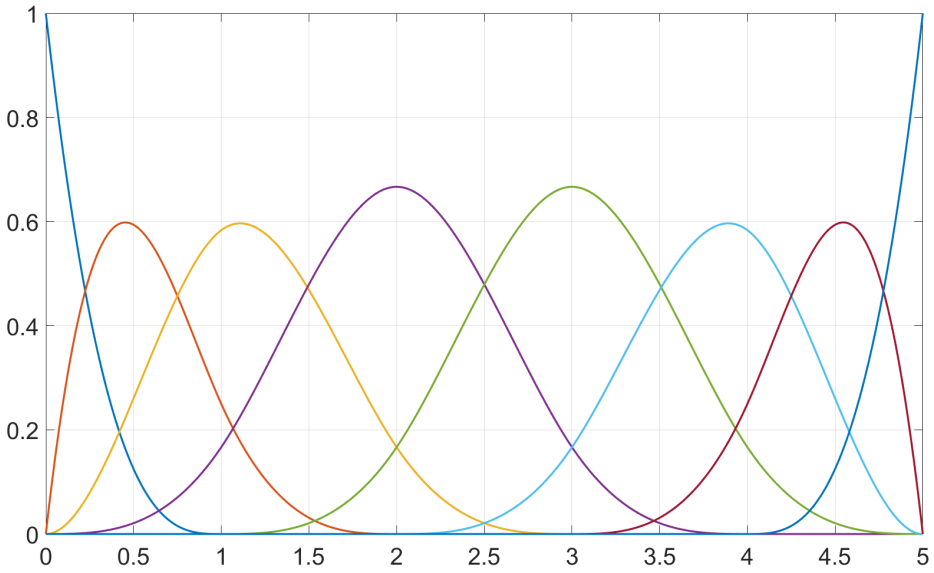


Figure 1.1. Quadratic B-splines defined on the uniform knot vector $\mathbf{t} = \{0, 1, 2, 3, 4, 5\}$.

1.1.2 B-spline curves

Let $\{\mathbf{P}_i\}_{i=0}^{n}$ be a set of $n+1$ points in $\mathbb{R}^2$ or $\mathbb{R}^3$ called *control points* and $\{N_{i,d}\}$ the d-th degree B-spline functions defined on the knot vector $\mathbf{t} = \{\underbrace{a, \ldots, a}_{d+1}, t_{d+1}, \ldots, t_{m-d-1}, \underbrace{b, \ldots, b}_{d+1}\}$

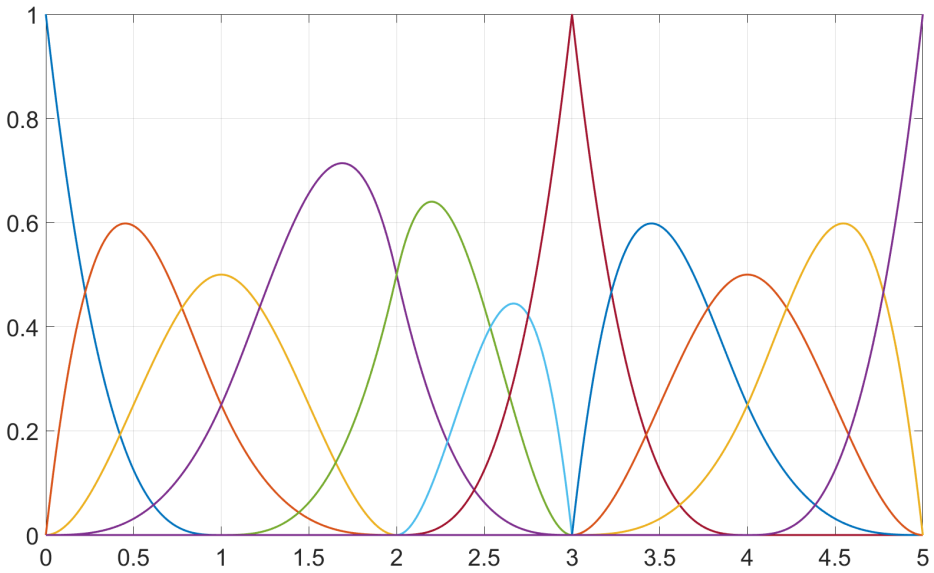


Figure 1.2. Cubic B-splines defined on the nonuniform knot vector $\mathbf{t} = \{0, 0, 0, 0, 1, 2, 2, 3, 3, 3, 4, 5, 5, 5, 5\}$.

consisting of $m + 1$ knots. Unless stated otherwise, we assume that $a = 0$ and $b = 1$. A B-spline curve of degree d is defined by

$$C(t) = \sum_{i=0}^{n} \mathbf{P}_i N_{i,d}(t), \quad 0 \le t \le 1,$$

where the number of control points $n + 1$ is given by the relation $n = m - d - 1$. The polygon formed by the $\{\mathbf{P}_i\}$ is called the *control polygon*. Figure 1.3 shows an example of B-spline curves. The properties of B-spline curves, which will be now listed, follow directly from the properties of the basis functions $N_{i,d}$ presented in Section 1.1.1:

- In the case $n = d$ and $\mathbf{t} = \{0, \ldots, 0, 1, \ldots, 1\}$, the curve $C(t)$ is a Bézier curve. This means that it can be rewritten as

$$C(t) = \sum_{i=0}^{n} \mathbf{P}_i B_d^i(t), \quad \text{with} \quad B_d^i(t) = \binom{d}{i} t^i (1-t)^{d-i} \tag{1.3}$$

  the Bernstein polynomials of degree d,

- *Convex hull*: the curve is contained in the convex hull of its control polygon,

- *Endpoints interpolation*: it results

$$C(0) = \mathbf{P}_0, \quad C(1) = \mathbf{P}_n,$$

- *Affine invariance*: the affine transformation of a B-spline curve is obtained just transforming its control points,

- *Local modification*: moving a control point $\mathbf{P}_i$ only affects $C(t)$ in the interval $[t_i, t_{i+d+1})$,

- The control polygon provides a piecewise linear approximation of the curve,

- *Variation diminishing*: the curve crosses any plane at most as many times as it intersects its control polygon,
- *Derivatives formula*: using (1.2), the k-th derivatives of a B-spline curve can be computed by the formula

$$C^{(k)}(t) = \sum_{i=0}^{n-k} \mathbf{P}_i^{(k)} N_{i,d-k}, \quad \text{where} \quad \mathbf{P}_i^{(k)} = \begin{cases} \mathbf{P}_i & k{=}0, \\ \frac{d-k+1}{t_{i+d+1}-t_{i+k}} \left(\mathbf{P}_{i+1}^{(k-1)} - \mathbf{P}_i^{(k-1)} \right) & k{>}0, \end{cases}$$

- $C(t) \in C^{d-\mu}$ at a knot of multiplicity μ.

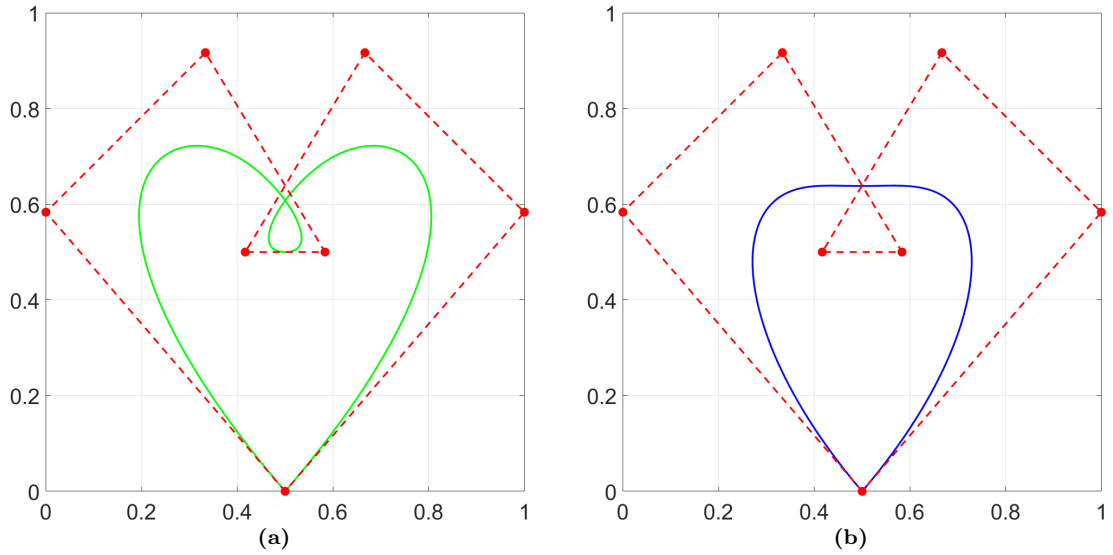


Figure 1.3. Cubic B-spline curve on $\mathbf{t} = \{0, 0, 0, 0, 1/3, 1/2, 1/2, 2/3, 1, 1, 1, 1\}$ (a) and degree 7 Bézier curve (b) obtained from the same control polygon with coinciding endpoints (red).

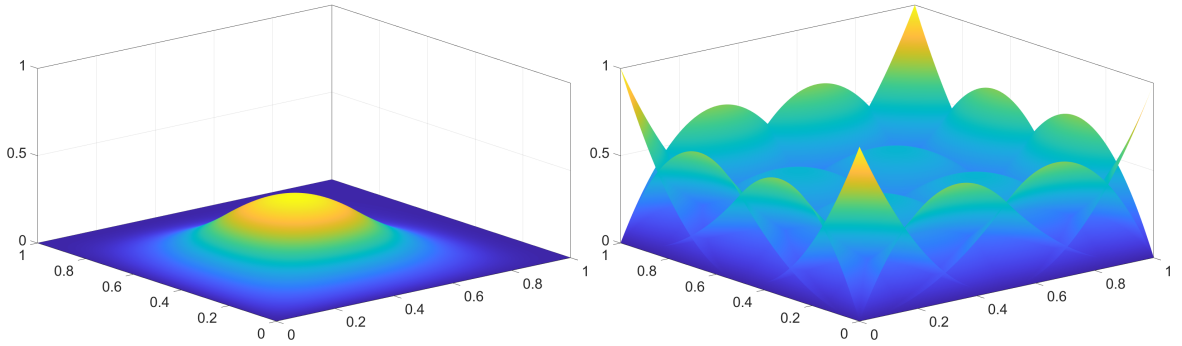
1.1.3 Tensor-product B-spline surfaces

The simplest bivariate B-spline construction is given by the tensor-product extension. It is obtained by taking a net of control points $\{\mathbf{P}_{i,j}\}_{i=0,j=0}^{n,r}$, defining the *control net*, and the product of the univariate B-spline functions $\{N_{i,d}\}_{i=0}^{n}$, of degree d over the knot vector $\mathbf{t}$ and degree q over the knot vector $\mathbf{s}$, i.e. $\{N_{j,q}\}_{j=0}^{r}$. More precisely, the B-spline surface $S(t, s)$ of bidegree (d, q) is given by

$$S(t, s) = \sum_{i=0}^{n} \sum_{j=0}^{r} \mathbf{P}_{i,j} N_{i,d}(t) N_{j,q}(s), \quad 0 \leq t, s, \leq 1,$$

with $\mathbf{t}$ and $\mathbf{s}$ defined similarly as in Section 1.1.2.

Figure 1.4 illustrates an example of bivariate tensor-product B-spline basis functions from which the B-spline surface in Figure 1.5 is obtained. It benefits from the same properties of B-spline curves, here shown for completeness:



(a) **(b)**

Figure 1.4. Single biquadratic B-spline basis function (a) and entire bases set (b) defined on $\mathbf{t} \times \mathbf{s}$, with $\mathbf{t} = \mathbf{s} = \{0, 0, 0, 1/2, 1, 1, 1\}$.

- if $n = d$, $r = q$, $\mathbf{t} = \mathbf{s} = \{0, \dots, 0, 1, \dots, 1\}$, then $S(t, s)$ is a Bézier surface,
- *Convex hull*: if $(t, s) \in [t_{\widetilde{i}}, t_{\widetilde{i}+1}) \times [s_{\widetilde{j}}, s_{\widetilde{j}+1})$, then $S(t, s)$ belongs to the convex hull of the points $\mathbf{P}_{i,j}$, $\widetilde{i} - d \leq i \leq \widetilde{i}$ and $\widetilde{j} - q \leq j \leq \widetilde{j}$,
- *Corner points interpolation*: it results

$$S(0,0) = \mathbf{P}_{0,0}, \quad S(1,0) = \mathbf{P}_{n,0}, \quad S(0,1) = \mathbf{P}_{0,r}, \quad S(1,1) = \mathbf{P}_{n,r},$$

- *Affine invariance*: an affine transformation is applied to the B-spline surface by applying it to the control net,
- *Local modification*: if the control point $\mathbf{P}_{i,j}$ is moved, $S(t, s)$ only changes in the rectangle $[t_i, t_{i+d+1}) \times [s_j, s_{j+q+1})$,
- The control net forms a piecewise planar approximation of the surface,
- *Derivatives formula*: the general formula returning the (k, l)-th derivative of a B-spline surface is given by

$$\frac{\partial^{k+l}}{\partial^k t \partial^l s} S(t,s) = \sum_{i=0}^{n-k} \sum_{j=0}^{r-l} \mathbf{P}_{i,j}^{(k,l)} N_{i,d-k}(t) N_{j,q-l}(s), \quad \text{with} \quad \mathbf{P}_{i,j}^{(k,l)} = (q-l+1) \frac{\mathbf{P}_{i,j+1}^{(k,l-1)} - \mathbf{P}_{i,j}^{(k,l-1)}}{s_{j+q+1} - s_{j+l}},$$

- $S(t, s)$ is $d - \mu$ (respectively $q - \mu$) times differentiable in the direction $\mathbf{t}$ ($\mathbf{s}$) at a knot t (s) of multiplicity μ.

1.1.3.1 Smooth joints between B-spline patches

Let us consider here two adjacent B-spline surfaces of bidegree (d, q)

$$S^L(u,v) = \sum_{i,j=0}^{n,m} \mathbf{P}_{i,j}^L N_{i,d}(u) N_{j,q}(v) \quad \text{and} \quad S^R(u,v) = \sum_{i,j=0}^{n,m} \mathbf{P}_{i,j}^R N_{i,d}(u) N_{j,q}(v).$$

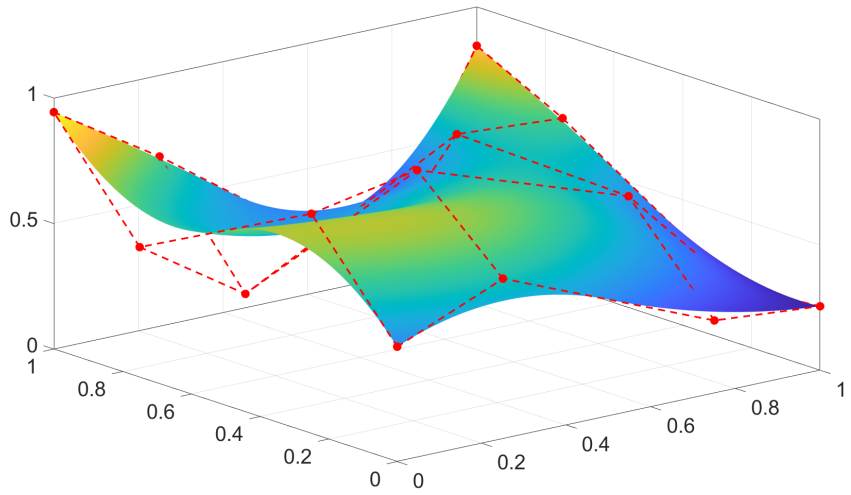


Figure 1.5. Biquadratic B-spline surface and its control net (red) obtained by using the basis functions in Figure 1.4.

We say that the patches $S^L(u,v)$ and $S^R(u,v)$ join C^r continuous if the points

$$\mathbf{P}^L_{n-r,l},\dots,\mathbf{P}^L_{n,l}=\mathbf{P}^R_{l,0},\dots,\mathbf{P}^R_{l,r} \tag{1.4}$$

constitute the control polygon of some Bézier curve of degree r $\forall\, l=0,\dots,m$. In particular, C^0 continuity just requires that $\mathbf{P}^L_{m,l}=\mathbf{P}^R_{0,l}$ for all $l=0,\dots,m$.

Tensor-product schemes are very easy to use in presence of a regular rectangular net. However, in real life we have to deal with general quadrangular shapes. In this setting imposing higher smoothness conditions between adjacent patches e.g. C^1 smoothness is not straightforward; in fact, the C^1 relations between control points across an edge form a linear system of $3(d+1)$ equations whose rank depends on the geometry of the quadrilaterals (see [BM17; Kap15] for more details). A simpler type of continuity which can be used in these situations is the so called *tangent plane continuity* or G^1 continuity. In words, we say that two patches join G^1 smoothly if they can be reparametrized so that their first derivatives are identical along a common boundary curve; in Section 1.3.2 we will properly define this geometric continuity in the context of splines over topological quad meshes.

1.1.4 NURBS

Non Uniform Rational B-spline, or NURBS, curves and surfaces, are an extension of the concepts in Section 1.1.2 and Section 1.1.3 which allow more flexible. Their more interesting feature is the ability to exactly reproduce section of conics, which was not possible with B-spline objects.

Formally, a NURBS curve of degree d is defined by

$$C(t)=\frac{\displaystyle\sum_{i=0}^{n} w_i\mathbf{P}_i N_{i,d}(t)}{\displaystyle\sum_{i=0}^{n} w_i N_{i,d}(t)},\quad 0\le t\le 1,$$

where the basis functions $\{N_{i,d}\}_{i=0}^{n}$ are defined over the knot vector $\mathbf{t}$ an in Section 1.1.2 and $\{w_i\}_{i=0}^{n}$, $w_i \in \mathbb{R}$, are the *weights* associated to the control points $\{\mathbf{P}_i\}_{i=0}^{n}$. The higher the value of the weight, the closer the curve will be to the corresponding control point. In case of negative weights the curve results moved away from its associated control point.

Analogously, we define a NURBS surface of bidegree (d, q) on the knot vectors $\mathbf{t}$ and $\mathbf{s}$ as Section 1.1.2 by

$$S(t,s) = \frac{\displaystyle\sum_{i=0}^{n}\sum_{j=0}^{r} w_{i,j}\mathbf{P}_{i,j}N_{i,d}(t)N_{j,q}(s)}{\displaystyle\sum_{i=0}^{n}\sum_{j=0}^{r} w_{i,j}N_{i,d}(t)N_{j,q}(s)}, \quad 0 \le t,s \le 1.$$

Both NURBS curves and surfaces enjoy similar properties and formulas to the B-spline case; see, e.g., [PT95] for a more in-depth look at the topic.

1.1.5 Fundamental algorithms

When dealing with B-splines it may be convenient, in some computation, to reformulate their algebraic expressions but without affecting the actual shapes. Two tools providing this kind of transformation are given by the *knot insertion algorithm* and *degree elevation algorithm*.

1.1.5.1 Knot insertion

Let $C(t) = \sum_{i=0}^{n} \mathbf{P}_i N_{i,d}(t)$ be a B-spline curve defined on the knot vector $\mathbf{t} = \{t_0, \dots, t_m\}$. Suppose we want to insert a knot $\bar{t} \in [t_k, t_{k+1})$ in the vector $\mathbf{t}$ defining the new knot vector $\bar{\mathbf{t}} = \{\bar{t}_0 = t_0, \dots, \bar{t}_k = t_k, \bar{t}_{k+1} = \bar{t}, \bar{t}_{k+2} = t_{k+1}, \dots, \bar{t}_{m+1} = t_m\}$; if $\mathcal{S}_{\mathbf{t}}$ and $\mathcal{S}_{\bar{\mathbf{t}}}$ identify the vector spaces of B-spline curves on $\mathbf{t}$ and $\bar{\mathbf{t}}$ respectively, clearly results that $\mathcal{S}_{\mathbf{t}} \subset \mathcal{S}_{\bar{\mathbf{t}}}$ and $\dim(\mathcal{S}_{\bar{\mathbf{t}}}) = \dim(\mathcal{S}_{\mathbf{t}}) + 1$. Hence $C(t)$ has a representation on $\mathbf{t}$ of the form

$$C(t) = \sum_{i=0}^{n+1} \mathbf{Q}_i \bar{N}_{i,d}(t),$$

where $\bar{N}_{i,d}$ are the B-spline basis functions on $\bar{\mathbf{t}}$. The new control points $\mathbf{Q}_i$ can be calculated either by direct computation, i.e. solving the linear system

$$\sum_{i=0}^{n} \mathbf{P}_i N_{i,d}(t) = \sum_{i=0}^{n+1} \mathbf{Q}_i \bar{N}_{i,d}(t)$$

in the unknowns $\mathbf{Q}_i$, or using geometric identities given by the properties in Section 1.1.2. Both ways lead to the solution

$$\mathbf{Q}_i = \alpha_i \mathbf{P}_i + (1-\alpha_i)\mathbf{P}_{i-1}, \quad \text{where} \quad \begin{cases} 1 & i \le k-d, \\ \frac{\bar{t}-t_i}{t_{i+d}-t_i} & k-d+1 \le i \le k, \\ 0 & i \ge k+1. \end{cases}$$

When dealing with curves with no inner knots, the repeated knot insertion algorithm coincides with the de Casteljau algorithm for Bézier curves. In case of surfaces, the same algorithm can be applied separately first to the control points along the direction $\mathbf{t}$ and then to control points along the direction $\mathbf{s}$, or vice versa.

1.1.5.2 Degree elevation

The goal of this algorithm is to identify a B-spline curve of degree d as an element of a curve space of higher degree $d+1$. As for the knot insertion algorithm, this procedure do not affects the geometry of the curve. Now let $C(t) = \sum_{i=0}^{n} \mathbf{P}_i N_{i,d}(t)$ be a B-spline curve of degree d on the knot vector $\mathbf{t}$. Since $C(t)$ is a piecewise polynomial curve, it is possible to elevate its degree to $d+1$ just by elevating the degree of its basis functions. Thus, there must exist control points $\widehat{\mathbf{Q}}_i$ and a knot vector $\widehat{\mathbf{t}}$ such that

$$\sum_{i=0}^{n} \mathbf{P}_i N_{i,d}(t) = \sum_{i=0}^{\widehat{n}} \widehat{\mathbf{Q}}_i N_{i,d+1}(t).$$

If the vector $\mathbf{t}$ has the form $\mathbf{t} = \{\underbrace{0,\dots,0}_{d+1}, \underbrace{t_1,\dots,t_1}_{\mu_1},\dots,\underbrace{t_m,\dots,t_m}_{\mu_m}, \underbrace{1,\dots,1}_{d+1}\}$, with $\mu_1,\dots,\mu_m$ the multiplicity of inner knots, in order to maintain the same regularity as the initial curve at the knots it must result that $\widehat{n} = n+m+1$ and $\widehat{\mathbf{t}} = \{\underbrace{0,\dots,0}_{d+2}, \underbrace{t_1,\dots,t_1}_{\mu_1+1},\dots,\underbrace{t_m,\dots,t_m}_{\mu_m+1}, \underbrace{1,\dots,1}_{d+2}\}$. Again, the new control points can be computed by using the properties of B-spline curves in Section 1.1.2 thus obtaining

$$\widehat{\mathbf{Q}}_1 = (1-\widehat{\alpha}_i)\mathbf{P}_i + \widehat{\alpha}_i \mathbf{P}_{i-1}, \quad \text{with} \quad \widehat{\alpha}_i = \frac{i}{d+1}, \quad i = 0,\dots,d+1.$$

To degree elevate a B-spline surface it is sufficient to apply the algorithm separately to the control points of the net aligned to the two directions $\mathbf{t}$ and $\mathbf{s}$.

1.2 Beyond tensor-product

The tensor-product structure is easy to deal with, but in real applications domains are not always rectangular. In this section we will introduce spline constructions over more general domains, that is triangulations and topological quad meshes. Moreover, we introduce the concept of geometric continuity, which is a widely used kind of regularity in multipatch approaches.

1.2.1 Splines on triangulations

We start this subsection introducing a local reference system, namely the barycentric coordinates system, that is much more useful than the usual Cartesian coordinates when dealing with triangles. Later, they will be used to define the so-called *Triangular Bernstein polynomials* and Bézier surfaces on a triangle. The notions presented in this subsection rely on [BS16; LS07].

1.2.1.1 Barycentric coordinates

Suppose $\triangle$ is a nondegenerate triangle in $\mathbb{R}^2$ with vertices $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$. It will be useful to write $\triangle = \langle \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3 \rangle$. Every point $\mathbf{x} \in \mathbb{R}^2$ has a unique representation in terms of its barycentric coordinates $(\beta_1, \beta_2, \beta_3)$ with respect to $\triangle$ such that

$$\mathbf{x} = \beta_1 \mathbf{p}_1 + \beta_2 \mathbf{p}_2 + \beta_3 \mathbf{p}_3, \quad \text{with} \quad \beta_1 + \beta_2 + \beta_3 = 1.$$

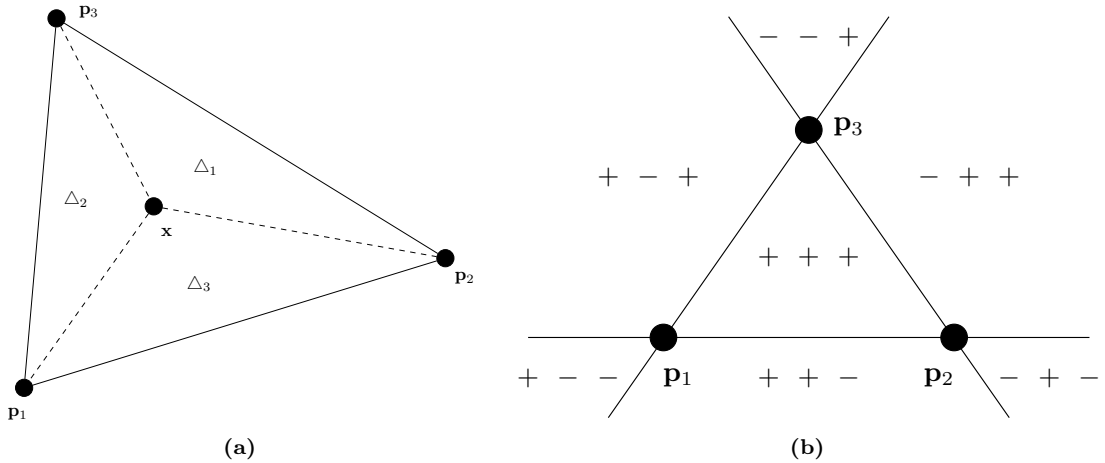


Figure 1.6. Geometric interpretation of barycentric coordinates (a) and subregions of $\mathbb{R}^2$ defined by their signs (b).

If the point $\mathbf{x}$ lies inside the triangle $\triangle$, then its barycentric coordinates are all positive; otherwise, some of them will result to be negative. Let $\triangle_1$ be the subtriangle $\triangle_1 = \langle \mathbf{x}, \mathbf{p}_2, \mathbf{p}_3 \rangle$ and similarly $\triangle_2 = \langle \mathbf{x}, \mathbf{p}_3, \mathbf{p}_1 \rangle$, $\triangle_3 = \langle \mathbf{x}, \mathbf{p}_1, \mathbf{p}_2 \rangle$ (cf. Figure 1.6); it results that

$$\beta_i = \frac{|\triangle_i|}{|\triangle|}, \quad i = 1, 2, 3,$$

where $|\triangle|$ indicates the signed area of the triangle $\triangle$. Moreover, if $\mathbf{x}_1$ and $\mathbf{x}_2$ are two points in $\mathbb{R}^2$ and $\triangle$ their reference triangle, the barycentric directional coordinates $(\delta_1, \delta_2, \delta_3)$ of the vector $\mathbf{x}_2 - \mathbf{x}_1$ are defined as the difference of their corresponding barycentric coordinates.

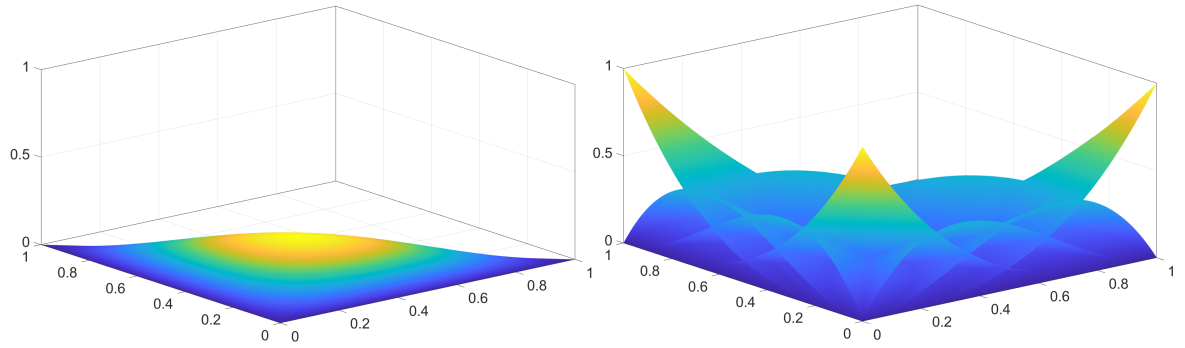
1.2.1.2 Triangular Bernstein polynomials

Given a triangle $\triangle$, we define the triangular Bernstein polynomials of degree d as

$$B_d^{i,j,k}(\mathbf{x}) = \frac{d!}{i!j!k!} \beta_1^i \beta_2^j \beta_3^k, \quad \forall\, i + j + k = d,$$

where $(\beta_1, \beta_2, \beta_3)$ are the barycentric coordinates of the point $\mathbf{x}$ w.r.t. $\triangle$. Figure 1.7 presents an example of these polynomials in the case $d = 3$. It is important to note that along each edge of a triangle, up to reordering of the indices, the triangular Bernstein polynomial coincides with the univariate Bernstein polynomial (1.3). They also possess properties which are similar to the univariate case, see Section 1.1.1. These properties are:

- *Nonnegativity*: $B_d^{i,j,k}(\mathbf{x}) \geq 0, \quad \mathbf{x} \in \triangle,$
- *Partition of unity*:
$$\sum_{i+j+k=d} B_d^{i,j,k}(\mathbf{x}) = 1, \quad \forall\, \mathbf{x} \in \mathbb{R}^2,$$
- *Recurrence relation*:
$$B_d^{i,j,k}(\mathbf{x}) = \beta_1 B_{d-1}^{i-1,j,k}(\mathbf{x}) + \beta_2 B_{d-1}^{i,j-1,k}(\mathbf{x}) + \beta_3 B_{d-1}^{i,j,k-1}(\mathbf{x}),$$



(a) **(b)**

Figure 1.7. A cubic triangular Bernstein polynomial (a) belonging to the set of 10 cubic triangular Bernstein polynomials (b).

with $(\beta_1, \beta_2, \beta_3)$ the barycentric coordinates of the point $\mathbf{x}$ w.r.t $\triangle$ and $B_d^{i,j,k}(\mathbf{x}) = 0$ in case of negative indices i, j, k,

- *Degree elevation*:

$$B_d^{i,j,k}(\mathbf{x}) = \frac{i+1}{d+1} B_{d-1}^{i-1,j,k}(\mathbf{x}) + \frac{j+1}{d+1} B_{d-1}^{i,j-1,k}(\mathbf{x}) + \frac{k+1}{d+1} B_{d-1}^{i,j,k-1}(\mathbf{x}), \tag{1.5}$$

- *Directional derivatives*: if $\mathbf{u}$ is a vector in $\mathbb{R}^2$ with barycentric directional derivatives $(\delta_1, \delta_2, \delta_3)$ w.r.t. $\triangle$, it holds that

$$D_{\mathbf{u}} B_d^{i,j,k}(\mathbf{x}) = d \left[\delta_1 B_{d-1}^{i-1,j,k}(\mathbf{x}) + \delta_2 B_{d-1}^{i,j-1,k}(\mathbf{x}) + \delta_3 B_{d-1}^{i,j,k-1}(\mathbf{x}) \right].$$

- *Unimodal behavior*: for $i \geq 1, j \geq 1, k \geq 1$ it results that $D_{\mathbf{u}} B_d^{i,j,k}(\mathbf{x}) = 0$ for any direction $\mathbf{u}$ and $\mathbf{x} \in \triangle$ if and only if

$$\mathbf{x} = \frac{i\mathbf{p}_1 + j\mathbf{p}_2 + k\mathbf{p}_3}{d} =: \boldsymbol{\xi}_{i,j,k}.$$

The points $\boldsymbol{\xi}_{i,j,k}$ are called *Greville points* or *domain points*. The domain points are uniformly distributed over the triangle, and they can be triangulated by linking each pair of domain points $\boldsymbol{\xi}_{i_1,j_1,k_1}$ and $\boldsymbol{\xi}_{i_2,j_2,k_2}$ such that

$$|i_1 - i_2| + |j_1 - j_2| + |k_1 - k_2| = 2. \tag{1.6}$$

1.2.1.3 Bézier surfaces

Let $\mathbf{P}_{i,j,k} \in \mathbb{R}^3$, $i + j + k = d$. A *Bézier surface* is defined as

$$S(\mathbf{x}) = \sum_{i+j+k=d} \mathbf{P}_{i,j,k} B_d^{i,j,k}(\mathbf{x}),$$

where $\mathbf{P}_{i,j,k}$ are its control points. Connecting these points as in (1.6) we obtain the *control net* of $S(\mathbf{x})$. Figure 1.8 shows an example of cubic Bézier surface.

They enjoy analogous properties to those of the tensor-product case:

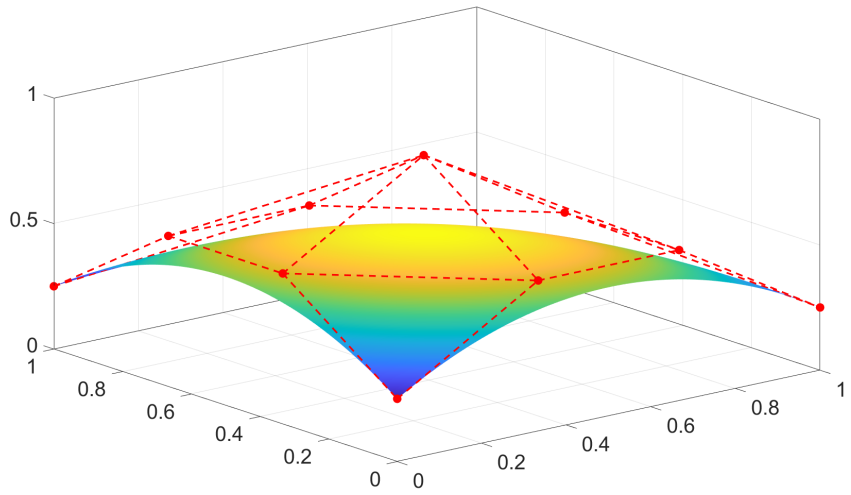


Figure 1.8. Cubic Bézier surface and its control net (red).

- *Convex hull*: $S(\mathbf{x})$ is contained in the convex hull of its control points $\mathbf{P}_{i,j,k}$,
- *Corner points interpolation*: $S(1,0,0) = \mathbf{P}_{3,0,0}$, $S(0,1,0) = \mathbf{P}_{0,3,0}$, $S(0,0,1) = \mathbf{P}_{0,0,3}$,
- *Directional derivatives formula*: let $\mathbf{u} \in \mathbb{R}^2$ be a vector. The m-th directional derivative is given by

$$D_{\mathbf{u}}^{(m)} S(\mathbf{x}) = \frac{d!}{(d-m)!} \sum_{i+j+k=d-m} \Delta_m \left(\mathbf{P}_{i,j,k}\right) B_{d-m}^{i,j,k}(\mathbf{x}), \quad m \leq d,$$

  where $\Delta_m \left(\mathbf{P}_{i,j,k}\right) := \mathbf{P}_{i,j,k}^{[d-m]}$ are the quantities obtained applying m steps of the triangular de Casteljau algorithm to the control points of $S(\mathbf{x})$, i.e.

$$\mathbf{P}_{i,j,k}^{[l-1]} = \beta_1 \mathbf{P}_{i+1,j,k}^{[l]} + \beta_2 \mathbf{P}_{i,j+2,k}^{[l]} + \beta_3 \mathbf{P}_{i,j,k+1}^{[l]},$$

  and $(\beta_1, \beta_2, \beta_3)$ the barycentric coordinates of $\mathbf{x}$.

- *Degree elevation*: by using (1.5), we obtain

$$\sum_{i+j+k=d} \mathbf{P}_{i,j,k} B_d^{i,j,k}(\mathbf{x}) = \sum_{i+j+k=d+1} \widehat{\mathbf{P}}_{i,j,k} B_{d+1}^{i,j,k}(\mathbf{x}),$$

  where

$$\widehat{\mathbf{P}}_{i,j,k} = \frac{i}{d+1} \mathbf{P}_{i-1,j,k} + \frac{j}{d+1} \mathbf{P}_{i,j-1,k} + \frac{k}{d+1} \mathbf{P}_{i,j,k-1}$$

  and $\mathbf{P}_{i,j,k} = 0$ if $i < 0$ or $j < 0$ or $k < 0$.

1.2.2 A brief summary of simplex splines

Simplex splines represent an extension to the multivariate case of classical B-spline functions. In fact, they benefit from the same malleability and similar properties to the univariate case. This subsection is based on [LMS22], while we point to, e.g. [PBP02], for the proofs of the properties listed here.

To define formally a m-variate simplex spline of a given degree d, let $\boldsymbol{\Xi} = \{\boldsymbol{\xi}_1, \ldots, \boldsymbol{\xi}_{n+1}\}$ be a sequence of possibly repeated points (called *knots*) in $\mathbb{R}^m$, with $n = d + m$. We assume that the convex hull of $\boldsymbol{\Xi}$ is nondegenerate, i.e. $\mathrm{vol}_m(\langle\boldsymbol{\Xi}\rangle) > 0$. Consider now any simplex in $\mathbb{R}^n$, $\boldsymbol{\sigma} = \{\bar{\boldsymbol{\xi}}_1, \ldots, \bar{\boldsymbol{\xi}}_{n+1}\}$ with $\mathrm{vol}_n(\boldsymbol{\sigma}) > 0$, satisfying the following projection property (cf. Figure 1.9):

$$\pi : \mathbb{R}^n \longrightarrow \mathbb{R}^m, \quad \pi(\bar{\boldsymbol{\xi}}_i) = \boldsymbol{\xi}_i, \quad i = 1, \ldots, n+1.$$

Hence, the (normalized) simplex spline $M_{\boldsymbol{\Xi}}$ can be defined as

$$M_{\boldsymbol{\Xi}} : \mathbb{R}^m \longrightarrow \mathbb{R}, \quad M_{\boldsymbol{\Xi}}(\mathbf{x}) = \frac{\mathrm{vol}_{n-m}(\boldsymbol{\sigma} \cap \pi^{-1}(\mathbf{x}))}{\mathrm{vol}_n(\boldsymbol{\sigma})}.$$

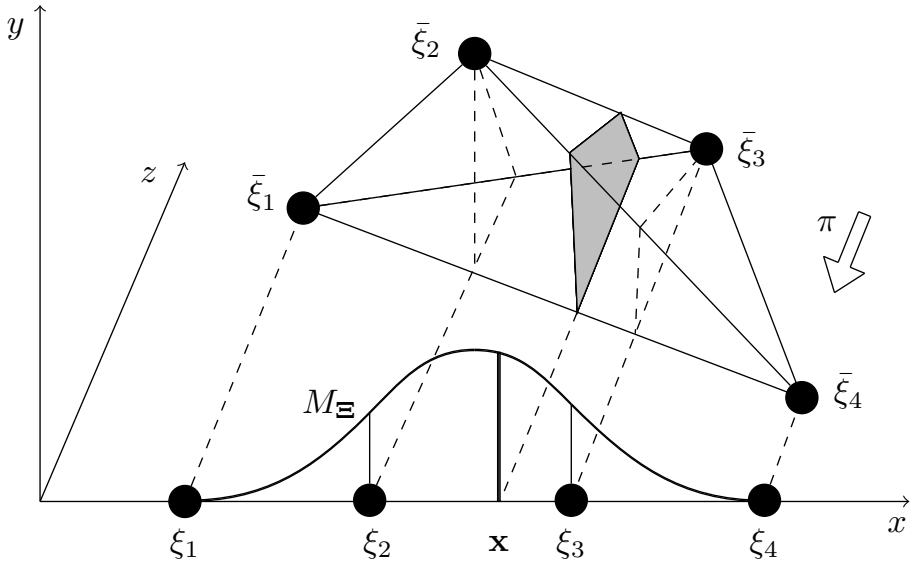


Figure 1.9. Univariate $M_{\boldsymbol{\Xi}}$ as projection of a tetrahedron over a line.

They benefit from properties analogous to the univariate case, such as

- *Local support*: $M_{\boldsymbol{\Xi}}$ has support $\langle\boldsymbol{\Xi}\rangle$,
- *Nonnegativity*: $M_{\boldsymbol{\Xi}}(\mathbf{x}) \geq 0, \quad \forall\, \mathbf{x} \in \langle\boldsymbol{\Xi}\rangle$,
- *Normalization*: $M_{\boldsymbol{\Xi}}$ has unit integral,
- *Knot dependence*: $M_{\boldsymbol{\Xi}}$ only depends on $\boldsymbol{\Xi}$; in particular, the ordering of the knots is not affecting the simplex spline. It is also independent of the choice of the simplex $\boldsymbol{\sigma}$,
- *Derivatives formula*: if $\mathbf{u} \in \mathbb{R}^m$ s.t. $\sum_i a_i \boldsymbol{\xi}_i = \mathbf{u}$, $\sum_i a_i = 0$, then

$$D_{\mathbf{u}} M_{\boldsymbol{\Xi}} = (d+m) \sum_{i=1}^{d+m+1} a_i M_{\boldsymbol{\Xi} \setminus \boldsymbol{\xi}_i},$$

- *Recurrence formula*: for any $\mathbf{x} \in \mathbb{R}^m$ s.t. $\sum_i b_i \boldsymbol{\xi}_i = \mathbf{x}$, $\sum_i b_i = 1$, it results

$$M_{\boldsymbol{\Xi}}(\mathbf{x}) = \frac{d+m}{d} \sum_{i=1}^{d+m+1} b_i M_{\boldsymbol{\Xi} \setminus \boldsymbol{\xi}_i}(\mathbf{x}),$$

as well as the *knot insertion algorithm* (similarly to Section 1.1.5): for any $\mathbf{y} \in \mathbb{R}^m$ s.t. $\sum_i c_i \boldsymbol{\xi}_i = \mathbf{y}$, $\sum_i c_i = 1$, we have

$$M_{\boldsymbol{\Xi}} = \sum_{i=1}^{d+m+1} c_i M_{\boldsymbol{\Xi} \cup \mathbf{y} \setminus \boldsymbol{\xi}_i}.$$

If $m = 1$, then $M_{\boldsymbol{\Xi}}$ is the normalized univariate B-spline of degree d with knot vector $\boldsymbol{\Xi}$. If $m = 2$, i.e. the bivariate case, we define *knot lines* the lines in the complete graph of $\boldsymbol{\Xi}$, which provide a polygonal partition of $\langle \boldsymbol{\Xi} \rangle$. Furthermore, in this setting the simplex spline $M_{\boldsymbol{\Xi}}$ is a polynomial of degree $d = \#\boldsymbol{\Xi} - 3$ in each region of this partition, and across a knot line $M_{\boldsymbol{\Xi}} \in C^{d+1-\mu}$, with μ the number of knots that lie on the considered knot line (counted with multiplicity).

1.3 Splines on meshes

This section is devoted to present the construction of smooth spline objects over quad and triangular meshes: in particular, we present the constraints defining C^1 continuity across a triangular planar mesh and the more general G^1 smoothness for splines over quadrilateral meshes.

1.3.1 Splines on triangular planar meshes

Similarly to the case of rectangular B-spline patches, we present here the C^r continuity conditions between adjacent triangular patches. Let $\triangle^L = \langle \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3 \rangle$ and $\triangle^R = \langle \mathbf{p}_4, \mathbf{p}_3, \mathbf{p}_2 \rangle$ be two adjacent triangles on which the polynomials

$$S^L(\mathbf{x}) = \sum_{i+j+k=d} \mathbf{P}^L_{i,j,k} B_d^{i,j,k}(\mathbf{x}), \quad S^R(\mathbf{x}) = \sum_{i+j+k=d} \mathbf{P}^R_{i,j,k} B_d^{i,j,k}(\mathbf{x})$$

are defined. Then $S^L(\mathbf{x})$ and $S^R(\mathbf{x})$ join C^r-smooth if and only if

$$\mathbf{P}^R_{m,j,k} = \sum_{\tilde{i}+\tilde{j}+\tilde{k}=m} \mathbf{P}^L_{\tilde{i},k+\tilde{j},j+\tilde{k}} B_m^{\tilde{i},\tilde{j},\tilde{k}}(\mathbf{p}_4), \qquad \begin{matrix} j+k = d-m, \\ m = 0, \dots, r. \end{matrix} \tag{1.7}$$

The relation (1.7) is the triangular variant of (1.4) in the tensor-product setting. C^0 continuity only requires that $\mathbf{P}^L_{i,j,0} = \mathbf{P}^R_{i,j,0}$, for all $i + j = d$, while, in addiction, C^1 smoothness implies that the pair of triangles

$$\langle \mathbf{P}^L_{i,j,1}, \mathbf{P}^L_{i+1,j,0}, \mathbf{P}^L_{i,j+1,0} \rangle, \quad \langle \mathbf{P}^R_{i,j,1}, \mathbf{P}^R_{i+1,j,0}, \mathbf{P}^R_{i,j+1,0} \rangle,$$

for each $i + j = d - 1$, are coplanar. Even when dealing with triangulations imposing (global) C^r continuity between adjacent triangles is not straightforward and resorting to geometric continuity can be a valid alternative (see [PBP02]).

We conclude the subsection introducing the definition of spline space over a triangulation. Let $\mathcal{T}$ be a triangulation of a polygonal domain $\Omega \in \mathbb{R}^2$, i.e. a partition of Ω consisting of non-overlapping triangles. The spline space of degree d on $\mathcal{T}$ with C^r continuity is defined as

$$\mathbb{S}^r_d(\mathcal{T}) := \left\{ s \in C^r(\Omega) \,:\, s\Big|_{\triangle_i} \in \mathbb{P}_d, \ \triangle_i \in \mathcal{T} \right\},$$

where we identify with $\mathbb{P}_d$ the space of polynomials in two variables and total degree d.

1.3.2 G^1-smooth splines over topological quad meshes

Here we introduce the concept of splines over topological quad meshes, with a particular focus on geometrically smooth spline spaces. In order to define these objects, the concepts of quadrilateral mesh, together with its features, as well as transition map and gluing data functions will be introduced.

We start introducing the concept of *topological surface* $\mathcal{M}$ (or *mesh*) as a collection of faces and edges together with their adjacency relations. An edge can be glued with at most one other edge and it cannot be glued with itself. Here we are interested in meshes which contain only quadrilateral faces and we shall refer to such a structure as quadrilateral mesh. An essential notion is the *valence* of a vertex, denoted with N, which is defined as the number of faces containing the vertex. An interior vertex is said to be *regular* (RV in short) if its valence is equal to 4, and it is called *irregular* (or *extraordinary* - EV) otherwise.

Let σ_0, σ_1 be two faces of the mesh $\mathcal{M}$ sharing the common edge $\mathbf{e}$; we can associate to each face a coordinate system, namely (u_0, v_0) and (u_1, v_1) on σ_0 and σ_1, respectively (see Figure 1.10). Now, we can therefore define the (first order) *transition map* ϕ_{σ_0,σ_1} from the face σ_1 to σ_0 as the function

$$\phi_{\sigma_0,\sigma_1} : \mathbb{R}^2 \longrightarrow \mathbb{R}^2, \quad (u_1, v_1) \longmapsto (u_0, v_0) = \begin{pmatrix} v_1 \mathfrak{b}_{N,N'}(u_1) + O(v_1^2) \\ u_1 + v_1 \mathfrak{a}_{N,N'}(u_1) + O(v_1^2) \end{pmatrix}, \tag{1.8}$$

which characterizes the change of coordinates from σ_1 to σ_0.

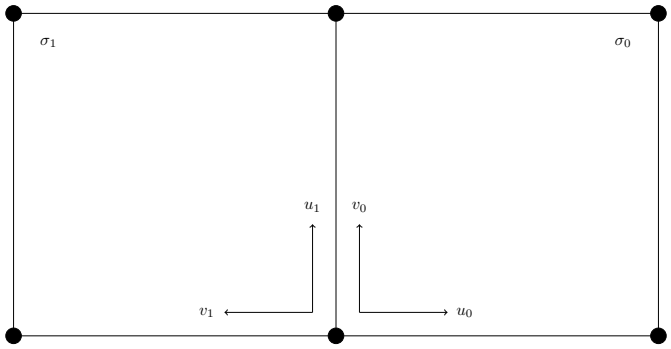


Figure 1.10. Reference systems on two adjacent patches.

Moreover, the functions in (1.8)

$$\mathfrak{a} : \mathbf{e} \longrightarrow \mathbb{R}, \quad \mathfrak{b} : \mathbf{e} \longrightarrow \mathbb{R},$$

are C^1 functions called *gluing data* and they only depend on the two valences N and N' of the vertices identifying the edge $\mathbf{e}$. We have all the ingredient to introduce the definition of G^r continuity across an edge shared by two patches. Consider a multipatch function f, i.e. a collections of functions $f = (f_\sigma)_{\sigma \in \mathcal{M}}$ defined over the faces of a mesh $\mathcal{M}$; it is said to be G^r-smooth if for any two functions f_{σ_0}, f_{σ_1}

$$J^r_{\mathbf{e}}(f_{\sigma_1})(u_1, v_1) = J^r_{\mathbf{e}}(f_{\sigma_0} \circ \phi_{\sigma_0,\sigma_1})(u_1, v_1), \tag{1.9}$$

where $J^r_{\mathbf{e}}$ indicates the jet (or Taylor expansion) of order r along the common edge $\mathbf{e}$.

In particular, for the case $r = 1$ we are mostly interested in, previous (1.9) leads to the following two relations for each $u_1 \in [0, 1]$:

$$\begin{cases} f_1(u_1, 0) = f_0(0, u_1), \\ \dfrac{\partial f_1}{\partial v_1}(u_1, 0) = \mathfrak{b}_{N,N'}(u_1)\dfrac{\partial f_0}{\partial u_0}(0, u_1) + \mathfrak{a}_{N,N'}(u_1)\dfrac{\partial f_0}{\partial v_0}(0, u_1), \end{cases} \tag{1.10}$$

where $f_0 = f|_{\sigma_0}, f_1 = f|_{\sigma_1}$ are the restrictions of f on the faces σ_0, σ_1. Geometrically, (1.10) means that the functions f_0 and f_1 share the same tangent plane along their interface, from where the name *tangent plane continuity*.

Finally, the spline space of bidegree (d, d) with regularity G^1 on a topological quad mesh $\mathcal{M}$ is defined as

$$\mathbb{S}^1_d(\mathcal{M}) := \left\{ (f_\sigma)_{\sigma\in\mathcal{M}} \,:\, f_i = f\Big|_{\sigma_i} \in \mathbb{P}_d,\ f_i, f_j \text{ join } G^1 \text{ across } \mathbf{e} = \sigma_i \cap \sigma_j \,\forall\, i, j \right\}.$$

We point to [BMV17; MVV16; Man23] for a deeper insight on geometrically smooth spline spaces.

1.4 Subdivision surfaces

In this subsection we shortly recall the very basics of Subdivision surfaces: the definition of mask and subdivision matrix will be introduced as well as the concept of limit surface. Also, some classical schemes will be recalled and analyzed in order to have a general idea on how they behave. For a complete and accurate description of subdivision schemes, see, for example, [PR08; AS10], on which this paragraph is founded.

1.4.1 Definitions and properties

Let us consider a mesh $\mathcal{M}$. The idea behind subdivision is to obtain smooth objects via iterative refinement of an input mesh; different refinement strategies lead to different limit objects which differ from both smoothness and shape. Each subdivision scheme is defined by its *masks* (or *stencils*): in general, masks relate to the vertices, edges and faces of the mesh. A mask is determined by a set of normalized coefficient (called *weights*), responsible for the definition of the recursive rule, which is applied to the considered vertex of the mesh and its *first* and *second neighborhood*. We call first neighborhood of a vertex $\mathbf{v}$ the set of points directly connected to $\mathbf{v}$ by an edge and, when dealing with quadrilateral meshes, second neighborhood of $\mathbf{v}$ the set of points belonging to the patches sharing $\mathbf{v}$ and opposite to it. Figure 1.11 presents an example of subdivision masks with underlined colors for the two neighborhoods.

In order to apply these subdivision operators (i.e. the masks) we make use of the so-called *subdivision matrix*, which is a matrix whose columns contains the subdivision masks and its size depends on the mesh size. If $\mathbf{p} = \mathbf{p}^{(0)}$ is a initial vector of points and $S = S^{(0)}$ is its related subdivision matrix, one step of subdivision is obtained by $\mathbf{p}^{(1)} = \mathbf{p}^{(0)}S^{(0)}$, which can be generalized to an arbitrary subdivision step as

$$\mathbf{p}^{(k+1)} = \mathbf{p}^{(k)}S^{(0)} \quad \text{or} \quad \mathbf{p}^{(k+1)} = \mathbf{p}^{(0)}S^{(k)},$$

with $S^{(k)}$ the subdivision matrix concerning the k-th subdivision step. The repeated subdivision converges to a surface called *limit surface*, whose regularity properties can be deduced carrying out a spectral analysis on the subdivision matrix defining the scheme.

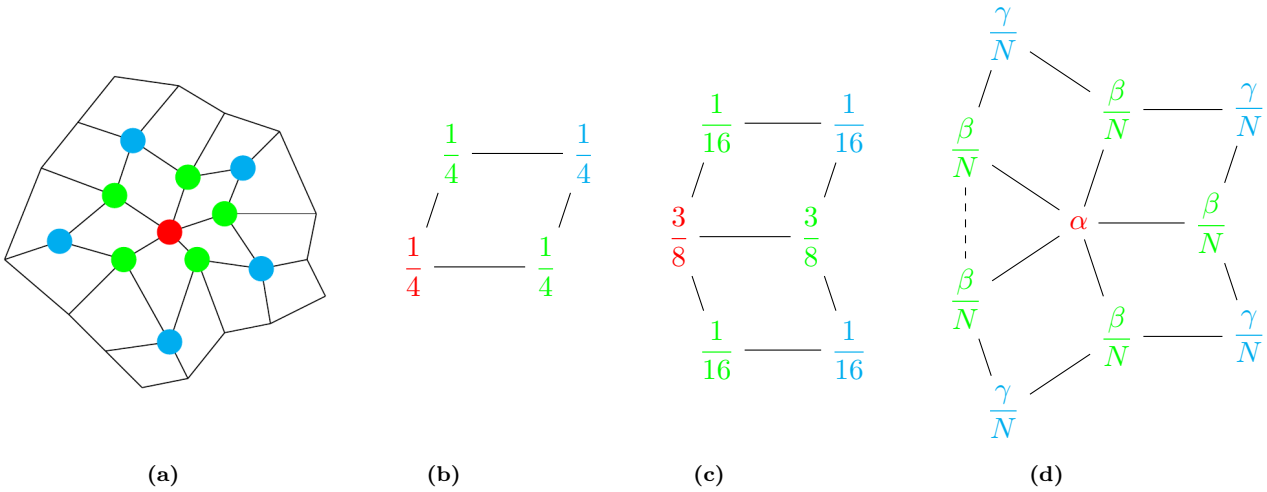


Figure 1.11. (a): first neighborhood (green) and second neighborhood (blue) of a vertex (red) with valence $N = 5$. Masks for a new face point (b), edge point (c) and vertex (d) defining the Catmull-Clark algorithm. $\alpha = 1 - 7/4N, \beta = 3/2N, \gamma = 1/4N$.

1.4.2 An overview on classical subdivision surfaces

The aim of this subsection is to give a visual idea of how different schemes act on meshes, generating different limit surfaces. We will consider four classical subdivision schemes for surfaces, namely the Catmull-Clark [CC78], Doo-Sabin [DS78], Loop [Loo87] and Butterfly [DLG90] schemes.

1.4.2.1 Catmull-Clark scheme

The Catmull-Clark scheme acts on quad meshes. It is an approximating scheme, i.e. the vertices of the initial mesh are not maintained in the subdivision process, and its limit surface is devised as a generalization of bicubic B-spline surfaces with C^2 regularity everywhere except at EVs, where it is C^1-smooth. Figure 1.12 shows the behaviour of this scheme on a quad mesh. Moreover, a direct evaluation of the limit surface is possible by using the formulas provided in [Sta98].

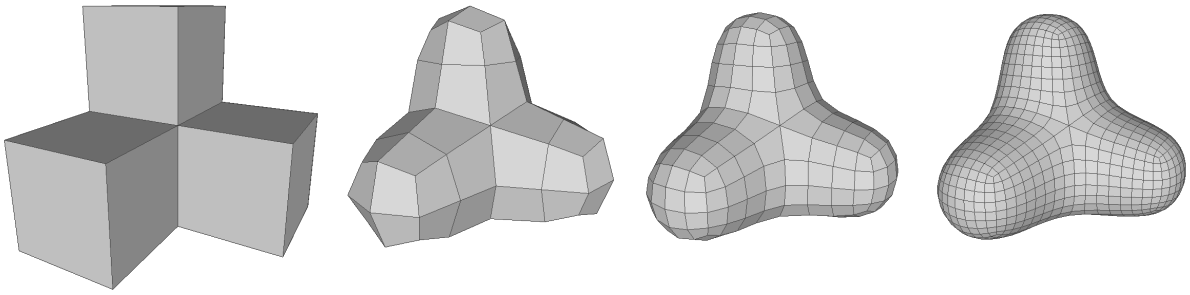


Figure 1.12. Three iterations of the Catmull-Clark scheme starting from the input mesh on the left.

1.4.2.2 Doo-Sabin scheme

Similarly to the Catmull-Clark scheme, Doo-Sabin rules are defined for quad meshes. It is an approximating scheme and it is based on a generalization of biquadratic uniform B-splines producing limit surfaces which are C^2 regular everywhere except at EVs, where we have C^1 smoothness; it is interesting to note that this scheme produces N-gons around EVs of valence N (see Figure 1.13). The limit surface evaluation proposed in [Sta98] is applicable also for this scheme.

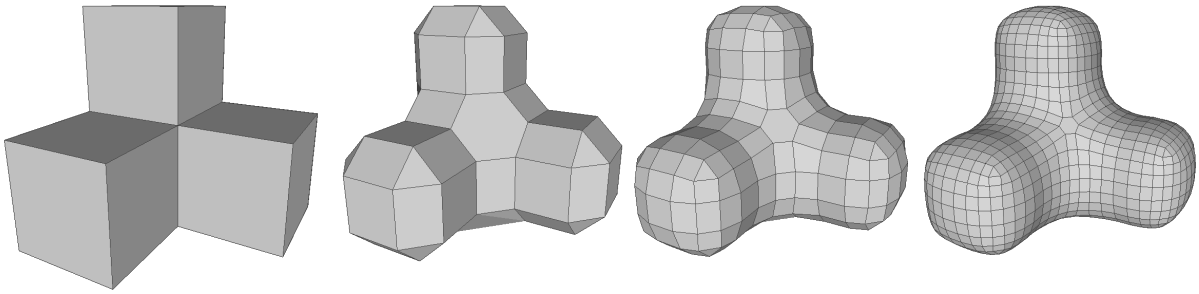


Figure 1.13. Three iterations of the Doo-Sabin scheme starting from the input mesh on the left.

1.4.2.3 Loop scheme

The Loop method for subdivision surfaces is an approximating scheme developed for triangular meshes. It is based on a quartic box spline and the limit surfaces it creates are C^2 continuous in regular regions while around EVs the smoothness drops to C^1 (EVs in triangular meshes are vertices with valence $N \neq 6$); Figure 1.14 presents an example of these surfaces. An explicit evaluation of the Loop limit surfaces can be achieved from [Sta01].

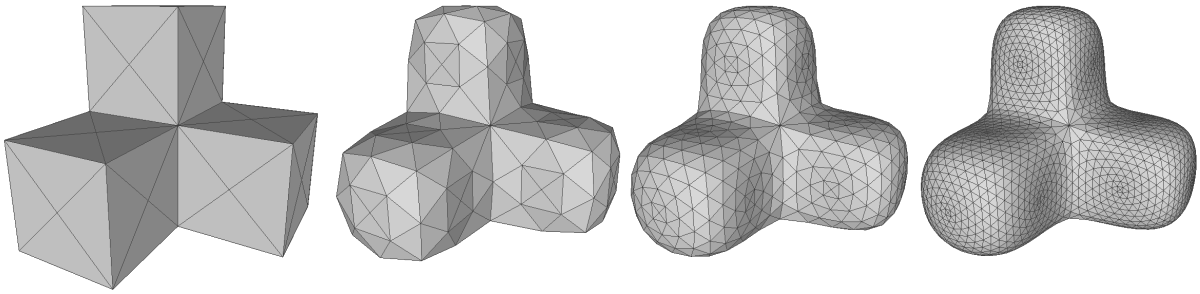


Figure 1.14. Three iterations of the Loop scheme starting from the input mesh on the left.

1.4.2.4 The Butterfly scheme

An example of interpolatory scheme is the Butterfly scheme: in fact, as can be noticed in Figure 1.15, the vertices of the initial triangular mesh are also vertices of the resulting subdivided surfaces. The limit surface it generates, as established in [Zor00] several years after the publication of the method itself, is C^1-smooth everywhere but around vertices with valences $N = 3$ and $N \geq 8$ where it is only continuous.

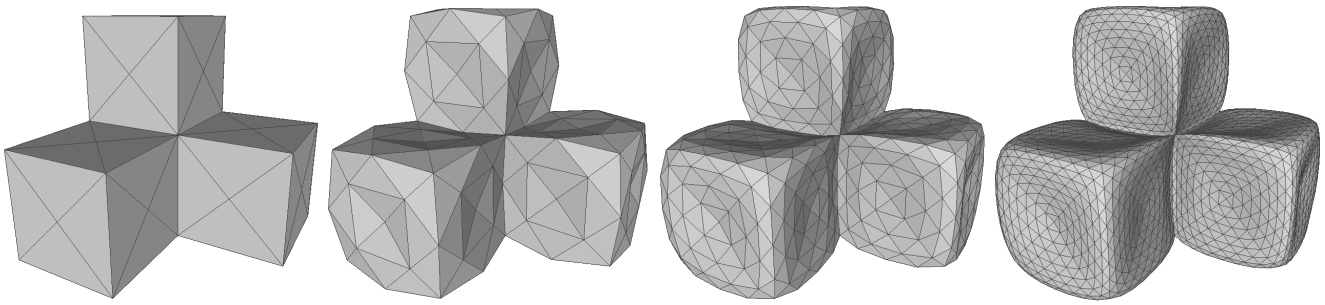


Figure 1.15. Three iterations of the Butterfly scheme starting from the input mesh on the left.

1.5 Isogeometric analysis

This last subsection is devoted to the introduction of isogeometric analysis. IsoGeometric Analysis (IGA) is a highly efficient technique for solving PDEs numerically. Its basic idea, presented in [CHB09], which unifies the Finite Elements Method (FEM in short) approach and Computer Aided Geometric Design, is to use the *same* basis functions for both reproducing exactly the computational domain, *and* for the numerical approximation of the PDE; more precisely, it is involved in the computation of the weak solution of a PDE. We start recalling some basics on PDEs, like their strong and weak formulation, and then continue introducing the so-called *isogeometric paradigm*. The content of this subsection has been extracted from [CHB09; BS16; Qua16], to which we also refer for the proofs of the results we recall.

1.5.1 PDEs: strong and weak formulation

Partial differential equations are differential equations containing the derivatives (spatial and/or temporal) of an unknown function. In particular, denote by u the unknown function in the $d+1$ independent variables $\mathbf{x} = (x_1, \dots, x_d)$ and t, a generic PDE can be denoted as

$$\mathcal{F}\left(t, \mathbf{x}, u, \frac{\partial u}{\partial t}, \frac{\partial u}{\partial x_1}, \dots, \frac{\partial u}{\partial x_d}, \dots, \frac{\partial^{p_1+\dots+p_d+p_t} u}{\partial x_1^{p_1} \dots \partial x_d^{p_d} \partial t^{p_t}}, g\right) = 0, \tag{1.11}$$

with $p_1, \dots, p_d, p_t \in \mathbb{N}$ and g the set of data on which the equation will depend (e.g. boundary conditions). We also define the order of a PDE as the maximum order of derivatives appearing in the equation, i.e. the value assumed by $p_1 + \dots + p_d + p_t$. A PDE written in the form (1.11) is said to be in its *strong* or *differential form* (or *formulation*). An example of PDE in a strong form is the following 2D *Poisson's equation*:

$$-\Delta u = f \quad \text{in } \Omega, \tag{1.12}$$

where $\Omega \subset \mathbb{R}^2$ is a bounded domain and we denote by

$$\Delta u = \sum_{i=1}^{d} \frac{\partial^2 u}{\partial x_i^2}$$

the *Laplace operator* or *laplacian*. In case $f = 0$, (1.12) is know as *Laplace equation*. In order to obtain a unique solution for (1.12), we need to add some extra conditions regarding the behaviour of the solution on the boundary of the domain $\partial\Omega$; the two standard boundary

conditions are assigning the value of the solution on $\partial\Omega$ (*Dirichlet condition*) or the value of the normal derivative of the solution along $\partial\Omega$ (*Neumann condition*). These choices are used to define the general problem, stated as follow: find u such that

$$\begin{cases} -\Delta u = f & \text{in } \Omega, \\ u = h_D & \text{on } \Gamma_D, \\ \dfrac{\partial u}{\partial \widehat{\mathbf{n}}} = h_N & \text{on } \Gamma_N, \end{cases}$$

where

$$\partial\Omega = \Gamma_D \cup \Gamma_N,$$

h_D and h_N are some given functions,

$$\frac{\partial u}{\partial \widehat{\mathbf{n}}} = \nabla u \cdot \widehat{\mathbf{n}}$$

is the *normal derivative* of u, $\widehat{\mathbf{n}}$ is the outward unit normal of Γ_N (or $\partial\Omega$ in general), the dot $\cdot$ represents the standard Euclidean scalar product (possibly also denoted as $\langle\cdot|\cdot\rangle$), and

$$\nabla : \mathbb{R}^d \longrightarrow \mathbb{R}^d, \quad u \longmapsto \nabla u = \left(\frac{\partial u}{\partial x_1}, \dots, \frac{\partial u}{\partial x_d}\right)$$

is the *gradient operator*. There exist in the literature other kinds of boundary conditions depending on the problem you are facing, e.g. *Robin condition*; for further details on this aspect we refer to [Qua16; BS08].

We derive now the so-called *weak form* of the Poisson's equation with homogeneous Dirichlet conditions, that is $h_D = 0$; the derivation of the weak formulation for other PDEs with eventually different boundary conditions is analogous. To do this, let v be a *test function*, i.e. a sufficiently smooth function which vanishes on $\partial\Omega = \Gamma_D$; later on we will properly define this function. Multiplying both members of (1.12) by v and integrating over Ω we get

$$-\int_\Omega \Delta u v \, d\Omega = \int_\Omega f v \, d\Omega. \tag{1.13}$$

By using the vectorial identity

$$\operatorname{div}(\psi \nabla \mathbf{A}) = \psi \operatorname{div}(\nabla \mathbf{A}) + \nabla \mathbf{A} \cdot \nabla \psi,$$

with $\mathbf{A}$ a vector field and ψ a scalar function, and the *divergence* or *Gauss theorem*

$$\int_\Omega \operatorname{div}(\mathbf{A}) \, d\Omega = \int_{\partial\Omega} A \cdot \widehat{\mathbf{n}} \, d\gamma,$$

where $d\gamma$ represents the infinitesimal boundary element of $\partial\Omega$, (1.13) becomes

$$\int_\Omega \nabla u \cdot \nabla v \, d\Omega = \int_\Omega f v \, d\Omega,$$

where we used the fact that v is null on the boundary of the domain. Therefore, we arrive at the following *weak formulation* of the Poisson's problem:

$$\text{find } u \in H_0^1(\Omega) \; : \; \int_\Omega \nabla u \cdot \nabla v \, d\Omega = \int_\Omega f v \, d\Omega \quad \forall v \in H_0^1(\Omega), \tag{1.14}$$

where, in order to have feasible operations, we take $f \in L^2(\Omega)$ and we set

$$H^1(\Omega) := \left\{ \varphi \in L^2(\Omega) \,:\, \frac{\partial \varphi}{\partial x_i} \in L^2(\Omega),\, i = 1, 2 \right\},$$
$$H_0^1(\Omega) := \left\{ \varphi \in H^1(\Omega) \,:\, \varphi\Big|_{\partial\Omega} = 0 \right\}.$$

In general, we can define the *Hilbert space* $H^k(\Omega)$ as

$$H^k(\Omega) := \left\{ \varphi \in L^2(\Omega) \,:\, D^{\boldsymbol{\alpha}}\varphi \in L^2(\Omega) \;\; \forall \boldsymbol{\alpha},\, |\boldsymbol{\alpha}| \leq k \right\}$$

with associated seminorms and norms

$$|\varphi|_{H^k(\Omega)} = \sqrt{\sum_{|\boldsymbol{\alpha}|=k} \int_\Omega (D^{\boldsymbol{\alpha}}\varphi)^2 \; d\Omega}, \quad \|\varphi\|_{H^k(\Omega)} = \sqrt{\sum_{m=0}^{k} |\varphi|^2_{H^k(\Omega)}},$$

where $\boldsymbol{\alpha}$ is a multi-index and the derivatives must be understood as distributional derivatives (cf. [Bre11]). It is interesting to notice that the problem (1.14) is equivalent to the following *variational problem*:

$$\text{find } u \in H_0^1(\Omega) \,:\, J(u) = \inf_{v \in H_0^1(\Omega)} \frac{1}{2} \int_\Omega |\nabla v|^2 \; d\Omega - \int_\Omega f v \; d\Omega.$$

To conclude, we introduce now a more compact formulation of the weak form (1.14). Let V be an Hilbert space ($V = H_0^1(\Omega)$ in our case). We can define the bilinear form

$$a : V \times V \longrightarrow \mathbb{R}, \quad a(u, v) = \int_\Omega \nabla u \cdot \nabla v \; d\Omega$$

and the linear functional

$$F : V \longrightarrow \mathbb{R}, \quad F(v) = \int_\Omega f v \; d\Omega$$

such that the problem (1.14) becomes

$$\text{find } u \in V \,:\, a(u, v) = F(v) \quad \forall v \in V. \tag{1.15}$$

We need the followings: a bilinear form $a : V \times V \longrightarrow \mathbb{R}$ is said to be

- *continuous* if $\exists\, M > 0$ such that

$$|a(u, v)| \leq M \|u\|_V \|v\|_V \quad \forall u, v \in V;$$

- *coercive* if $\exists\, \alpha > 0$ such that

$$a(v, v) \geq \alpha \|v\|_V^2 \quad \forall v \in V.$$

The existence and uniqueness of a solution for the problem (1.15) is ensured by the following [BS08, Theorem 2.7.7]

Theorem 1.5.1 (Lax-Milgram). *Let V be an Hilbert space, $a(\cdot,\cdot) : V \times V \longrightarrow \mathbb{R}$ a continuous and coercive bilinear form, $F(\cdot) : V \longrightarrow \mathbb{R}$ a continuous linear functional. Then exist a unique $u \in V$ such that*

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

When dealing with nonhomogeneous Dirichlet conditions, i.e. $h_D \neq 0$, the solution can be decomposed as

$$u = u_0 + u_D,$$

where $u_0 \in H_0^1(\Omega)$ is obtained from (1.14), and $u_D \in H_D^1(\Omega)$, with

$$H_D^1(\Omega) := \left\{ \varphi \in H^1(\Omega) \, : \, \varphi\Big|_{\partial\Omega} = h_D \right\},$$

is given.

1.5.1.1 Strong and weak formulation of some classical PDEs

The aim of this paragraph is to introduce some classical PDEs, together with their weak formulations, that we are going to utilize later on in this work; ambient spaces, domains of definitions and boundary conditions will be properly defined when necessary.

- ***Biharmonic equation***

$$\Delta^2 u = f \qquad \underset{strong}{\overset{weak}{\rightleftharpoons}} \qquad \int_\Omega \Delta u \cdot \Delta v \; d\Omega = \int_\Omega f v \; d\Omega, \tag{1.16}$$

  where Δ^2 represents the *biharmonic operator* defined as

$$\Delta^2 u = \Delta(\Delta u) = \sum_{i,j=1}^{d} \frac{\partial^4 u}{\partial x_i^2 \partial x_j^2}.$$

- ***Linear time-independent shallow-water equation***

$$h\Delta u + \nabla h \cdot \nabla u + \kappa^2 u = 0 \qquad \underset{strong}{\overset{weak}{\rightleftharpoons}} \qquad \begin{aligned} &\int_\Omega h \nabla u \cdot \nabla v \; d\Omega + \int_\Omega v \nabla h \cdot \nabla u \; d\Omega \\ &+ \kappa^2 \int_\Omega uv \; d\Omega = 0, \end{aligned} \tag{1.17}$$

  with h the *bathymetry* function and $\kappa^2 = \sigma^2/g \in \mathbb{R}$, where g is the gravitational acceleration and σ the frequency of the oscillations.

- ***Heat equation***

$$\frac{\partial u}{\partial t} = c^2 \Delta u \qquad \underset{strong}{\overset{weak}{\rightleftharpoons}} \qquad c^2 \int_\Omega \nabla u \cdot \nabla v \; d\Omega = \int_\Omega \frac{\partial u}{\partial t} v \; d\Omega, \tag{1.18}$$

  with $c \in \mathbb{R}$ the *conductivity* coefficient.

1.5.2 Galerkin's method and isogeometric spaces

Galerkin's approach for IGA is based on the decomposition of the variational problem into smaller ones defined onto subspaces of the initial space. Formally, let V be an Hilbert space and V_h, $h \in \mathbb{R}$, be a one-parameter family of spaces such that

$$V_h \subset V, \;\; \dim(V_h) < \infty \;\; \forall h > 0.$$

The *Galerkin's problem* (or *approximate problem*) for (1.15) can be stated as

$$\text{find } u_h \in V_h \; : \; a(u_h, v_h) = F(v_h) \quad \forall v_h \in V_h. \tag{1.19}$$

If $\{\phi_i\}_{i \in \mathcal{I}}$, $\mathcal{I} = \{1, \dots, \dim(V_h)\}$, is a basis of V_h, then it's enough that relation (1.19) is satisfied for each basis function, i.e.

$$a(u_h, \phi_i) = F(\phi_i), \quad i \in \mathcal{I}. \tag{1.20}$$

Therefore, since $u_h \in V_h$,

$$u_h(\mathbf{x}) = \sum_{j \in \mathcal{I}} u_j \phi_j(\mathbf{x}),$$

with u_j, $j \in \mathcal{I}$ unknown coefficients, (1.20) becomes

$$\sum_{j \in \mathcal{I}} u_j a(\phi_j, \phi_i) = F(\phi_i), \quad i \in \mathcal{I}. \tag{1.21}$$

Denoting with K the *stiffness* matrix with entries

$$k_{i,j} = a(\phi_j, \phi_i),$$

$\mathbf{f}$ the vector of components $f_i = F(\phi_i)$ and with $\mathbf{u}$ the vector of the unknown coefficients u_j, (1.21) is equivalent to the linear system

$$K\mathbf{u} = \mathbf{f}.$$

For completeness, we also introduce the *mass* matrix M, whose elements are given by

$$m_{i,j} = m(\phi_j, \phi_i), \quad \text{with} \quad m(u, v) = \int_\Omega uv \; d\Omega;$$

this matrix may appear in the discretization of some variational formulation, e.g. (1.17). The existence and uniqueness of a solution of the Galerkin's problem is again ensured by the Lax-Milgram theorem (1.5.1), since V_h is an Hilbert space (being a closed subspace of V) and the form $a(\cdot, \cdot)$ and functional $F(\cdot)$ are the same as in (1.15). Now, let assume that our domain Ω is given through a spline transformation of the parametric domain $\widehat{\Omega} = [0, 1]^2$. This means that there exist a map $\mathbf{G}$, called *geometry map*, that is

$$\mathbf{G} : \widehat{\Omega} \longrightarrow \mathbb{R}^m, \; m = 2, 3, \quad \mathbf{G}(\boldsymbol{\xi}) = \sum_{i,j \in \mathcal{J}} \mathbf{c}_{i,j} \psi_{i,j}(\boldsymbol{\xi}),$$

where $\boldsymbol{\xi} = (\xi_1, \xi_2)$, $\psi_{i,j} : \widehat{\Omega} \longrightarrow \mathbb{R}$ are spline basis functions of a certain degree d spanning the space $\mathcal{S}_d$ and $\mathbf{c}_{i,j} \in \mathbb{R}^2$ are their control points on a suitable indices set $\mathcal{J} = \{1, \dots, \dim(\mathcal{S}_d)\}$,

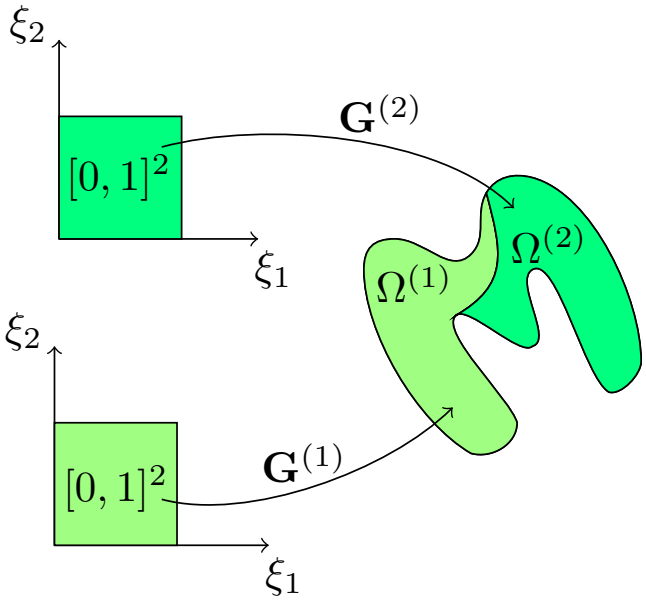


Figure 1.16. A multipatch domain Ω consisting of two patches $\Omega^{(1)}, \Omega^{(2)}$ with their associated geometry mappings $\mathbf{G}^{(1)}, \mathbf{G}^{(2)}$.

such that $\Omega = \mathbf{G}(\widehat{\Omega})$. The founding idea behind isogeometric analysis is that the basis used to exactly describe the geometry will also serve as basis for the computation of the numerical solution. This concept brings us to the definition of the *isogeometric function space*

$$V_h = \left\{ v_h \in L^2(\Omega) \,:\, v_h \in \mathcal{S}_d \circ \mathbf{G}^{-1} \right\}$$

on which we solve our approximate problem. For an exhaustive description on mathematics of IGA we refer the reader to [Bei14].

1.5.3 PDEs on manifolds

It is common in applications to deal with physical domains which are not planar. In this case, we need to reformulate the PDE in terms of operators on manifolds such that the *Laplace-Beltrami* operator.

Manifolds are usually described as multipatch domains, i.e. as the union of single patch $\Omega^{(k)}$, each of them defined through a different geometry map $\mathbf{G}^{(k)}$, with $k \in \mathcal{I}_\Omega$ in some index set. Figure 1.16 shows this idea when dealing with a shape consisting of 2 patches.

Let us consider the patch $\Omega^{(k)}$ of the 2-manifold domain $\Omega \subset \mathbb{R}^3$. The Jacobian $J^{(k)}$ of the map $\mathbf{G}^{(k)}$, defined as

$$\widehat{J}^{(k)} : \widehat{\Omega} \longrightarrow \mathbb{R}^{3\times 2}, \quad \boldsymbol{\xi} \longmapsto \widehat{J}^{(k)}(\boldsymbol{\xi}), \quad \widehat{J}^{(k)}(\boldsymbol{\xi}) = \left[\frac{\partial G^{(k)}}{\partial \xi_1}(\boldsymbol{\xi}) \;\; \frac{\partial G^{(k)}}{\partial \xi_2}(\boldsymbol{\xi}) \right],$$

is used to define the *first fundamental form* of the mapping $\widehat{G}^{(k)}$

$$\widehat{G}^{(k)} : \widehat{\Omega} \longrightarrow \mathbb{R}^{2\times 2}, \quad \boldsymbol{\xi} \longmapsto \widehat{G}^{(k)}(\boldsymbol{\xi}), \quad \widehat{G}(\boldsymbol{\xi}) = \left(\widehat{J}^{(k)}(\boldsymbol{\xi}) \right)^T \widehat{J}^{(k)}(\boldsymbol{\xi}),$$

together with its determinant $\widehat{g}^{(k)}$, which is

$$\widehat{g}^{(k)} : \widehat{\Omega} \longrightarrow \mathbb{R}, \quad \boldsymbol{\xi} \longmapsto \widehat{g}^{(k)}(\boldsymbol{\xi}), \quad \widehat{g}^{(k)}(\boldsymbol{\xi}) = \sqrt{\det\left(\widehat{G}^{(k)}(\boldsymbol{\xi})\right)}.$$

Let $\varphi^{(k)} \in C^2(\Omega^{(k)})$ and $\widehat{\varphi}^{(k)}(\boldsymbol{\xi}) = \left(\varphi^{(k)} \circ \mathbf{G}^{(k)}\right)(\boldsymbol{\xi})$. Finally, we have all the tools to define the gradient operator on the patch $\Omega^{(k)}$ of the manifold Ω,

$$\nabla_{\Omega^{(k)}} \varphi^{(k)}(\mathbf{x}) = \left[\widehat{J}^{(k)}(\boldsymbol{\xi}) \left(\widehat{G}^{(k)}\right)^{-1}(\boldsymbol{\xi}) \widehat{\nabla} \widehat{\varphi}^{(k)}(\boldsymbol{\xi})\right] \circ \left(\mathbf{G}^{(k)}\right)^{-1}(\boldsymbol{\xi}), \quad \mathbf{x} \in \Omega^{(k)}, \tag{1.22}$$

with $\widehat{\nabla}$ representing the gradient operator in the parametric space, and the Laplace-Beltrami operator

$$\Delta_{\Omega^{(k)}} \varphi^{(k)}(\mathbf{x}) = \left[\frac{1}{\widehat{g}^{(k)}(\boldsymbol{\xi})} \widehat{\nabla} \cdot \left(\widehat{g}^{(k)}(\boldsymbol{\xi}) \left(\widehat{G}^{(k)}\right)^{-1}(\boldsymbol{\xi}) \widehat{\nabla} \widehat{\varphi}^{(k)}(\boldsymbol{\xi})\right)\right] \circ \left(\mathbf{G}^{(k)}\right)^{-1}(\boldsymbol{\xi}), \quad \mathbf{x} \in \Omega^{(k)}. \tag{1.23}$$

Hence, we can reformulate both strong and weak formulations of the PDEs (1.14) (1.16), (1.17) and (1.18) just replacing the gradient and laplacian operators with, respectively, the gradient operator on the manifold (1.22) and the Laplace-Beltrami operator (1.23). For further details on the description of PDEs on manifolds we refer to [DQ15; Far23].

Chapter 2

A G^1 approximation of Catmull-Clark surfaces

Catmull-Clark subdivision scheme is undoubtedly the most famous and used scheme to obtain smooth surfaces from quad meshes. However, the limit surfaces that it generates may not be smooth enough for applications, e.g. in computer-aided design. The purpose of this chapter is to present a new, smooth and completely explicit construction of a globally G^1 family of Bézier surfaces, defined by smoothing masks, approximating the Catmull-Clark limit surface.

Starting from the C^0 scheme of Loop and Schaefer [LS08], we impose G^1 regularity around EVs by means of quadratic gluing data depending solely on their valences; this procedure leads to a system of equations of which we explore the space of solutions in order to achieve explicit formulas for the masks defining the patches' control points. Finally, we come up with four different schemes. In order to understand which scheme provides the most regular surface, we conduct isophotes and curvature analysis on an extensive benchmark, both visually and numerically. This chapter is based on [MMM22].

2.1 Bicubic Approximate Catmull-Clark (ACC_3) and degree elevated (ACC_5)

The bicubic approximate Catmull-Clark scheme of [LS08] is a smoothing scheme reconstructing a bicubic surface starting from a quad-mesh which approximates the limit surface of the classical Catmull-Clark subdivision scheme; the application of this scheme to a mesh returns, on each face, 16 control points defining a bicubic Bézier patch. The surface obtained from ACC_3 is C^2 everywhere except in a neighborhood of the EVs, in which it is only C^0; moreover, the vertices generated by ACC_3 interpolate the Catmull-Clark limit surface. Usually, similarly to subdivision schemes, a smoothing scheme is defined through its smoothing matrix $S = (\mathrm{M}_{i,j}^{(k)})$ as an operator such that

$$\mathbf{b} = \mathbf{p}\, S, \quad \mathbf{b}_{i,j}^{(k)} = \sum_{\ell} \mathbf{p}_\ell \left(\mathrm{M}_{i,j}^{(k)}\right)_\ell,$$

where $\mathbf{p} = (\mathbf{p}_1, \dots, \mathbf{p}_m) \in \mathbb{R}^{3\times m}$ is the set of mesh vertices and $\mathbf{b} = (\mathbf{b}^{(k)})$, where $\mathbf{b}^{(k)}$ is the matrix containing the control points of the k-th patch (see Figure 2.2 for their labeling). The smoothing matrix contains in its columns the different masks M defining the scheme and in this construction the weights of a smoothing (or Bézier) mask can be, possibly, negative. We will refer to smoothing masks using the notation $\mathrm{M}=[\omega_\bullet, \boldsymbol{\omega}, \boldsymbol{\omega'}]^T$, where $\omega_\bullet$ is the weight for the vertex point and $\boldsymbol{\omega} = \{\omega_i\}_{i=1}^N$ and $\boldsymbol{\omega'} = \{\omega'_i\}_{i=1}^N$ are the weights for its first and second

neighborhoods, respectively. Lastly, in this construction we will consider meshes with isolated EVs, i.e. surrounded by regular vertices; this assumption is not restrictive since any linked EVs can be decoupled with one subdivision of the mesh.

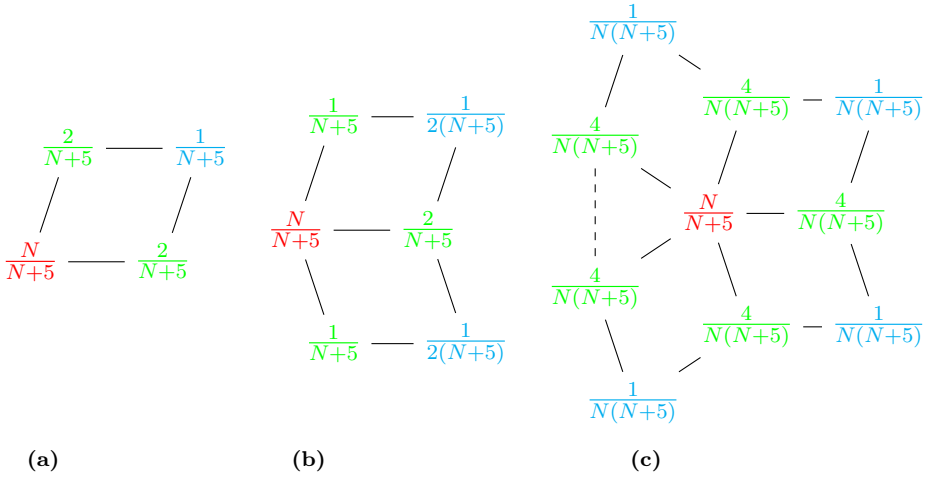


Figure 2.1. From (a) to (c): ACC_3 masks proposed in [LS08] for face ($\widehat{\mathbf{b}}_{1,1}$), edge ($\widehat{\mathbf{b}}_{1,0}$ and $\widehat{\mathbf{b}}_{0,1}$) and vertex ($\widehat{\mathbf{b}}_{1,1}$) Bézier points, respectively. The coloring refers to Figure 1.11-(a).

To recover regularity around these EVs without affecting the interpolatory property and C^1 (or C^2) smoothness of the surrounding regular vertices, linear gluing data and cubic or even quartic patches do not suffice. Indeed the G^1 relations for Bézier patches of bidegree $(3,3)$ or $(4,4)$ propagate up to control points neighboring the regular vertices, affecting then their properties. To avoid these effects, we will work with gluing data functions of degree two and biquintic patches, defined by 36 Bézier control points each. As a starting point of our construction we use the degree elevated masks ACC_5, which are obtained after applying twice the degree elevation algorithm (Section 1.1.5.2) to ACC_3 masks; we will use underlined symbols to denote ACC_5 masks i.e. $\underline{\mathrm{M}} = [\underline{\omega}_\bullet, \underline{\boldsymbol{\omega}}, \underline{\boldsymbol{\omega}}']^T$. For example, the ACC_5 masks $\underline{\mathrm{M}}_{2,2}$ and $\underline{\mathrm{M}}_{4,0}$ returning the Bézier points $\mathbf{b}_{2,2}$ and $\mathbf{b}_{4,0}$, respectively, are given by

$$\underline{\mathrm{M}}_{2,2} = \frac{1}{100}\left(\widehat{\mathrm{M}}_{0,0} + 6\widehat{\mathrm{M}}_{1,0} + 3\widehat{\mathrm{M}}_{2,0} + 6\widehat{\mathrm{M}}_{0,1} + 36\widehat{\mathrm{M}}_{1,1} + 18\widehat{\mathrm{M}}_{2,1} + 3\widehat{\mathrm{M}}_{0,2} + 18\widehat{\mathrm{M}}_{1,2} + 9\widehat{\mathrm{M}}_{2,2}\right),$$
$$\underline{\mathrm{M}}_{4,0} = \frac{1}{5}\left(3\widehat{\mathrm{M}}_{2,0} + 2\widehat{\mathrm{M}}_{3,0}\right),$$

where $\widehat{\mathrm{M}}_{i,j}$ are the ACC_3 bicubic masks (see Figure 2.1). Note that the degree elevation is directly applied on the considered masks to obtain the new weights. The masks for Bézier points can be written in vectorial form; for instance, the corner point mask $\underline{\mathrm{M}}_{0,0}$ is given as follows:

$$\underline{\mathrm{M}}_{0,0} \in \mathbb{R}^{2N+1}, \text{ with } \underline{\mathrm{M}}_{0,0} = \frac{1}{N(N+5)}[N^2, 4, \ldots, 4, 1, \ldots, 1]^T = \frac{1}{N(N+5)}[N^2, \mathbf{4}, \mathbf{1}]^T, \tag{2.1}$$

where the first coordinate is the weight for the EV, followed by the weights for the vertices of the first neighborhood and then the second neighborhood; we will adopt this notation throughout the thesis.

2.2 G^1 constraints on Bézier patches

As we said in the previous section, our aim is to construct new smoothing masks defining Bézier patches that join G^1 at extraordinary regions. This can be obtained by the description in Section 1.3.2 as follow: let $[\mathfrak{a}_{N,N'}, \mathfrak{b}_{N,N'}]$ be quadratic symmetric gluing data functions [BH14; HBC08; Hah89]

$$\begin{aligned} &\mathfrak{a}_{N,N'}(u) = a_0 B_2^0(u) + a_2 B_2^2(u), \quad \text{where} \quad a_0 = 2\cos\left(\frac{2\pi}{N}\right),\ a_2 = 2\cos\left(\frac{2\pi}{N'}\right), \\ &\mathfrak{b}_{N,N'}(u) = -1, \end{aligned} \tag{2.2}$$

with B_d^i the univariate Bernstein polynomials in (1.3) defined along the edge $\mathbf{e}$ shared by two adjacent patches. Since our construction is developed in the parametric domain, we take $\mathbf{e} = [0, 1]$; moreover, in the cases we study we will assume that $N' = 4$ so that $a_2 = 0$, that is an EV is surrounded by regular vertices.

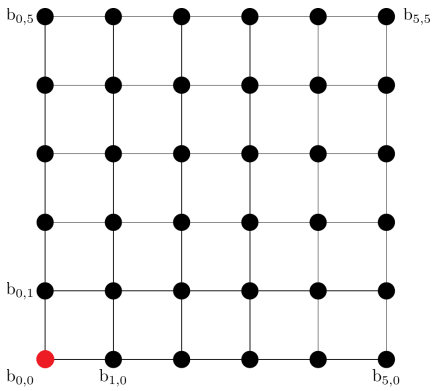


Figure 2.2. Control points labeling in a biquintic patch.

Now, if we specialize (1.10) defining the G^1 constraints between two adjacent patches with f_0 and f_1 being two biquintic Bézier surfaces and using (2.2), we obtain

$$\begin{cases} \displaystyle\sum_{i=0}^{5} \mathbf{b}_{i,0}^{(1)} B_5^i(u) = \sum_{i=0}^{5} \mathbf{b}_{0,i}^{(0)} B_5^i(u), \\ \displaystyle\sum_{i=0}^{5} \left(\mathbf{b}_{i,1}^{(1)} - \mathbf{b}_{i,0}^{(1)} + \mathbf{b}_{1,i}^{(0)} - \mathbf{b}_{0,i}^{(0)}\right) B_5^i(u) = a_0 B_2^0(u) \left(\displaystyle\sum_{i=0}^{4} \left(\mathbf{b}_{0,i+1}^{(0)} - \mathbf{b}_{0,i}^{(0)}\right) B_4^i(u)\right). \end{cases} \tag{2.3}$$

Calling with $\mathrm{M}_{i,j}^{(k)}$ the smoothing mask returning the control point $\mathbf{b}_{i,j}^{(k)}$ belonging to the patch k, solving (2.3) with respect to $\mathrm{M}_{0,1}^{(1)}, \mathrm{M}_{1,1}^{(1)}, \mathrm{M}_{2,1}^{(1)}, \mathrm{M}_{3,1}^{(1)}, \mathrm{M}_{4,1}^{(1)}, \mathrm{M}_{5,0}^{(1)}, \mathrm{M}_{5,1}^{(1)}$, after some algebraic

reductions we obtain the following relations to be satisfied:

$$\begin{aligned}
\mathrm{M}^{(1)}_{0,1} + \mathrm{M}^{(0)}_{1,0} &= \bar{a}_0\mathrm{M}^{(1)}_{0,0} + a_0\mathrm{M}^{(1)}_{1,0}\,, && (2.4)\\
5(\mathrm{M}^{(1)}_{1,1} + \mathrm{M}^{(0)}_{1,1}) &= a_0\mathrm{M}^{(1)}_{0,0} + 5\bar{a}_0\mathrm{M}^{(1)}_{1,0} + 4a_0\mathrm{M}^{(1)}_{2,0}\,, && (2.5)\\
10(\mathrm{M}^{(1)}_{2,1} + \mathrm{M}^{(0)}_{1,2}) &= -a_0\mathrm{M}^{(1)}_{0,0} + 5a_0\mathrm{M}^{(1)}_{1,0} + 10\bar{a}_0\mathrm{M}^{(1)}_{2,0} + 6a_0\mathrm{M}^{(1)}_{3,0}\,, && (2.6)\\
10(\mathrm{M}^{(1)}_{3,1} + \mathrm{M}^{(0)}_{1,3}) &= a_0\mathrm{M}^{(1)}_{0,0} - 5a_0\mathrm{M}^{(1)}_{1,0} + 10a_0\mathrm{M}^{(1)}_{2,0} + 10\bar{a}_0\mathrm{M}^{(1)}_{3,0} + 4a_0\mathrm{M}^{(1)}_{4,0}\,, && (2.7)\\
\mathrm{M}^{(1)}_{4,1} + \mathrm{M}^{(0)}_{1,4} &= 2\mathrm{M}^{(1)}_{4,0}\,, && (2.8)\\
\mathrm{M}^{(1)}_{5,1} + \mathrm{M}^{(0)}_{1,5} &= 2\mathrm{M}^{(1)}_{5,0}\,, && (2.9)\\
10(\mathrm{M}^{(1)}_{3,0} - \mathrm{M}^{(1)}_{2,0}) &= \mathrm{M}^{(1)}_{0,0} - 5\mathrm{M}^{(1)}_{1,0} + 5\mathrm{M}^{(1)}_{4,0} - \mathrm{M}^{(1)}_{5,0}\,, && (2.10)
\end{aligned}$$

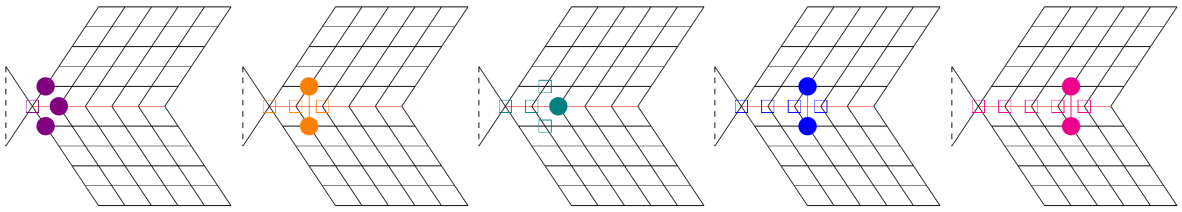
with $\bar{a}_0 = 2 - a_0$.

Masks verifying (2.4) to (2.10) generate a family of G^1 smoothing schemes. From now on the dependency of the mask from the patch will be specified only when it is different from patch 1. Particularly interesting is (2.10): if we use quadratic gluing data, the degree of the left-hand side of (1.10) is ≤ 5 while the right-hand side has degree $4 + 2 = 6$. (2.10) reflects the constraint that the left-hand side has degree ≤ 5: in fact it is saying that even if we are working with biquintic patches, along the edge we have a quartic curve. Moreover, we notice that (2.8) and (2.9) are always satisfied by ACC_5 masks [LS08, Figure 3] : since we are in the vicinity of a regular vertex, we know that the ACC_5 surface is C^2 there; it means that the first derivatives along opposite directions across the regular vertex are the same, i.e. $\mathrm{M}^{(0)}_{1,5} - \mathrm{M}^{(0)}_{0,5} = -(\mathrm{M}^{(1)}_{5,1} - \mathrm{M}^{(1)}_{5,0})$ as $\mathrm{M}^{(0)}_{0,5} = \mathrm{M}^{(1)}_{5,0}$, which is in fact (2.9). Then, since the limit surface is also at least C^1 across regular vertices (or C^2 in some cases), we have, additionally, equal second order derivatives along opposite directions i.e. $\mathrm{M}^{(0)}_{1,4} - \mathrm{M}^{(0)}_{0,4} = -(\mathrm{M}^{(1)}_{4,1} - \mathrm{M}^{(1)}_{4,0})$ that is exactly (2.8) since $\mathrm{M}^{(0)}_{0,4} = \mathrm{M}^{(1)}_{4,0}$.

2.3 Explicit Bézier masks derivation

In this section we are going to solve the system (2.4) to (2.10) to obtain masks for the Bézier patches ensuring G^1 regularity of the resulting surface. The process is done incrementally with respect to the order of the Bézier mask where the order refers to the sum of the subscripts of the Bézier mask. In particular, for Bézier masks of order zero, i.e. vertex points, we take $\mathrm{M}_{0,0} = \underline{\mathrm{M}}_{0,0}$ as in (2.1) in order to not alterate the interpolating property we have at the vertices. For Bézier points of order one we will solve (2.4); since the latter involves Bézier points on different patches all around the EV, (2.4) translates into a circulant system to be solved. (2.5) presents constraints regarding masks of second order and it can be solved in two different ways; if we want to solve it with respect to $\mathrm{M}_{1,1}$, since it is related to points around the EV in different patches similarly to (2.4), it translates into a circulant system. On the other hand, if we want to solve it with respect to the mask $\mathrm{M}_{2,0}$ it can be directly obtained from ACC_5 masks. Then, (2.6) and (2.7) regarding Bézier points of order three and four, respectively, can be solved by direct computation in either a symmetric or asymmetric approach. Moreover, the masks $\mathrm{M}_{4,0}, \mathrm{M}_{5,0}, \mathrm{M}_{4,1}, \mathrm{M}_{5,1}, \mathrm{M}_{2,2}, \mathrm{M}_{3,2}, \mathrm{M}_{4,2}, \mathrm{M}_{5,2}, \mathrm{M}_{2,3}, \mathrm{M}_{3,3}, \mathrm{M}_{4,3}, \mathrm{M}_{5,3}$ as well as $\mathrm{M}_{0,4}, \mathrm{M}_{1,4}, \mathrm{M}_{2,4}, \mathrm{M}_{3,4}, \mathrm{M}_{4,4}, \mathrm{M}_{5,4}, \mathrm{M}_{0,5}, \mathrm{M}_{1,5}, \mathrm{M}_{2,5}, \mathrm{M}_{3,5}, \mathrm{M}_{4,5}, \mathrm{M}_{5,5}$ are equal to those of ACC_5. Throughout the construction we assume that the first neighborhood of an EV is composed

solely by regular vertices; this assumption is not restrictive, since it is satisfied by applying at most one step of any 4-split algorithm. Moreover, biquintic Bézier points will be computed only in the faces adjacent to EVs, otherwise we will use the standard bicubic patch of ACC_3. First we focus on inner EVs and postpone the discussion of boundaries until Section 2.3.4.



(a) **(b)** **(c)** **(d)** **(e)**

Figure 2.3. Bézier points involved in G^1 constraints. (a): disposal of the control points involved in the left-hand side (circles) and right-hand side (squares) of (2.12) involved in the computation of the first derivatives. (b)-(c): disposal of Bézier points regarding the left-hand side and right-hand side of (2.17) and (2.28), respectively, involved in the computation of the second derivatives. (d)-(e): disposal of the control points utilized, respectively, in the left-hand side and right-hand side of (2.29) and (2.31) for the computation of the third and fourth derivatives.

2.3.1 Bézier masks of order one

We compute first the masks $\mathrm{M}_{1,0} = \mathrm{M}_{1,0}^{(1)}$ and $\mathrm{M}_{0,1}^{(0)}$ using (2.4) as follows. If the valence of the EV is $N = 3$, (2.4) is automatically satisfied by ACC_5 masks. Then, we do not need to determine new masks for this relation. Otherwise, for valences $N \geq 5$ we have to compute new masks verifying (2.4); given the mask for the vertex point $\mathrm{M}_{0,0} = \underline{\mathrm{M}}_{0,0} = [\underline{\alpha}_\bullet, \underline{\boldsymbol{\alpha}}, \underline{\boldsymbol{\alpha}'}]^T$, we shall compute the mask $\mathrm{M}_{1,0} = [\beta_\bullet, \boldsymbol{\beta}, \boldsymbol{\beta}']^T$. We observe that the masks $\mathrm{M}_{0,1} = \mathrm{M}_{0,1}^{(1)}$ and $\mathrm{M}_{1,0}^{(0)}$ can be obtained by permuting the weights of $\mathrm{M}_{1,0}$. In particular, let $C = \mathrm{diag}(1, \widehat{C}, \widehat{C})$ be a $(2N+1) \times (2N+1)$ block diagonal matrix where

$$\widehat{C} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \\ 1 & 0 & & \cdots & 0 \end{pmatrix} = \mathrm{Circ}(0, 1, 0, \ldots, 0) \tag{2.11}$$

is a $N \times N$ circulant matrix; it follows that $\mathrm{M}_{1,0}^{(0)} = C\,\mathrm{M}_{1,0}$, $\mathrm{M}_{0,1} = C^{-1}\mathrm{M}_{1,0}$, where $C^{-1} = \mathrm{diag}(1, \widehat{C}^{-1}, \widehat{C}^{-1})$ and $\widehat{C}^{-1} = \widehat{C}^{N-1}$. Therefore (2.4) translates into the following system to be solved:

$$(-a_0 I + C + C^{N-1})\mathrm{M}_{1,0} = (2 - a_0)\underline{\mathrm{M}}_{0,0}, \tag{2.12}$$

where I is the $(2N+1) \times (2N+1)$ identity matrix; Figure 2.3-(a) shows the position of the control points involved in this construction. Our investigation towards explicit solutions leads to the following

Proposition 2.3.1. *Let $A = -a_0 I + C + C^{N-1}$ be as in (2.12). The matrix A is symmetric and singular.*

Proof. From the particular shape of A, its symmetry is straightforward since $C^T = C^{-1} = C^{N-1}$. Now, the matrix A can be decomposed as

$$A = -a_0 I + C + C^{N-1} = C^{N-1}(C^2 - a_0 C + I) = C^{N-1}(C - z_1 I)(C - z_{N-1} I)\,,$$

where z_1, z_{N-1} are the roots of the polynomial $x^2 - a_0 x + 1$, with $a_0 = 2\cos(\theta)$, $\theta = 2\pi/N$. If $\mathtt{i} = \sqrt{-1}$ is the imaginary unit, they are obtained as

$$z_1 = e^{\mathtt{i}\theta}, \quad z_{N-1} = e^{-\mathtt{i}\theta},$$

i.e. they are the N-th roots of unity $z_k = e^{\mathtt{i}\theta k}$ corresponding to the angle θ for $k = 1$ and $k = N - 1$. Now,

$$\det(C - z_i I) = (1 - z_i)\det\left(\widehat{C} - z_i \widehat{I}\right)^2, \quad i \ = 1, N-1,$$

where $\widehat{I}$ is the $N \times N$ identity matrix; since $\widehat{C}^N - \widehat{I} = 0$ we have

$$\det\left(\widehat{C} - \lambda \widehat{I}\right) = (-1)^N(\lambda^N - 1) \tag{2.13}$$

from which we deduce that

$$\det\left(\widehat{C} - z_i \widehat{I}\right) = (-z_i)^N + (-1)^{N+1} = 0, \quad i = 1, N-1.$$

This shows that $A = C^{N-1}(C - z_1 I)(C - z_{N-1} I)$ is not invertible. □

The system (2.12) can be therefore solved; the following theorem ensures its solvability.

Theorem 2.3.2. *The system* (2.12) *admits a four-dimensional affine space of solutions.*

Proof. Let $\mathbf{b} = (2 - a_0)\underline{\mathrm{M}}_{0,0}$ be the right-hand side of (2.12); we want to show that $\mathbf{b} \in \mathrm{Im}(A)$ or, equivalently, $W\mathbf{b} = \mathbf{0}$, where $\mathrm{Im}(W^T) = \mathrm{Leftker}(A) = \mathrm{Ker}(A)$ since A is symmetric. Recalling the identity

$$\mathbf{0} = C^N - I = \prod_{j=0}^{N-1}(C - z_j I),$$

with $z_k = e^{\mathtt{i}\theta k}$ the N-th roots of unity, it is easy to prove that we can take

$$W^T = \prod_{\substack{j=0 \\ j \neq 1}}^{N-2}(C - z_j I) = \mathrm{diag}(0, \widehat{W}^T, \widehat{W}^T),$$

in which

$$\widehat{W}^T = \prod_{\substack{j=0 \\ j \neq 1}}^{N-2}(\widehat{C} - z_j \widehat{I}) \tag{2.14}$$

and $\widehat{C}$ as in (2.11): in fact

$$WA = (AW^T)^T = \left(C^{N-1}(C - z_1 I)(C - z_{N-1} I) \prod_{\substack{j=0 \\ j\neq 1}}^{N-2} (C - z_j I) \right)^T$$
$$= \left(C^{N-1} \prod_{j=0}^{N-1} (C - z_j I) \right)^T = \mathbf{0}$$

since the matrices $(C - z_j I)$ commute. Collecting the constants from the product we have

$$W\mathbf{b} = \left(\mathbf{b}^T W^T\right)^T = \left(\frac{2 - a_0}{N(N+5)} \mathrm{diag}(N^2, 4\widehat{I}, \widehat{I})(1 - z_0, \mathbf{1} - \boldsymbol{z_0}, \mathbf{1} - \boldsymbol{z_0}) \prod_{j=2}^{N-2} (C - z_j I) \right)^T = \mathbf{0}$$

being $z_0 = 1$; due to the particular block structure of A, we deduce that $\mathrm{corank}(A) = 4$. Thus the solutions of (2.12) form an affine space of dimension 4. □

From the previous Theorem 2.3.2, in order to compute a solution of (2.12) we need to add 4 extra constraints to the coefficient matrix; then, since we arrive to an overdetermined system, the solution we get has to be meant in the least squares sense. To do that, let $\widehat{\mathbf{k}}_1$ and $\widehat{\mathbf{k}}_2$ be the two (column) vectors of one basis of $\widehat{W}^T$: with this choice it results that

$$\mathrm{Im}(W^T) = \mathrm{Span}\{\mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3, \mathbf{k}_4\},$$

where

$$\mathbf{k}_1 = (0, \widehat{\mathbf{k}}_1^T, \mathbf{0})^T, \quad \mathbf{k}_2 = (0, \widehat{\mathbf{k}}_2^T, \mathbf{0})^T, \quad \mathbf{k}_3 = (0, \mathbf{0}, \widehat{\mathbf{k}}_1^T)^T, \quad \mathbf{k}_4 = (0, \mathbf{0}, \widehat{\mathbf{k}}_2^T)^T.$$

Here we decided to impose the 4 extra constraints such that the new masks results to be as close as possible to $\underline{\mathrm{M}}_{1,0}$; this can be achieved projecting their difference onto $\mathrm{Im}(W^T)$; this projection, in principle, can be executed arbitrary, i.e. choosing any direction. Let M_* be a particular solution (2.12); then, the general solution of (2.12) is given by

$$\mathrm{M}_{1,0} = \mathrm{M}_* + \sum_{i=1}^{4} \mu_i \mathbf{k}_i, \quad \mu_i \in \mathbb{R}. \tag{2.15}$$

Since the rows of A sum up to $2 - a_0$, i.e. $A\,(1, \mathbf{1}, \mathbf{1})^T = (2 - a_0)\,(1, \mathbf{1}, \mathbf{1})^T$, it is immediate to notice that we can take $\mathrm{M}_* = \underline{\mathrm{M}}_{0,0}$. Here we choose to perform an orthogonal projection $(\mathrm{M}_{1,0} - \underline{\mathrm{M}}_{1,0}) \perp \mathrm{Im}(W^T)$, that is

$$\langle \mathrm{M}_{1,0} - \underline{\mathrm{M}}_{1,0} \,|\, \mathbf{k}_i \rangle = 0, \quad i = 1, 2, 3, 4.$$

This leads to the solution obtained by solving

$$K\boldsymbol{\mu} = \widetilde{\mathbf{b}}, \tag{2.16}$$

where $K = \mathrm{diag}(\widehat{K}, \widehat{K})$ and

$$\widehat{K} = \left(\langle \widehat{\mathbf{k}}_i \,|\, \widehat{\mathbf{k}}_j \rangle \right)_{i,j=1,2}$$

is the 2×2 orthogonal projection matrix and

$$\widetilde{\mathbf{b}} = \left(\mathbf{k}_1\ \mathbf{k}_2\ \mathbf{k}_3\ \mathbf{k}_4\right)^T \underline{\mathrm{M}}_{1,0}.$$

We also observe that if we solve (2.15) for $N = 3$, we obtain again the ACC_5 mask $\underline{\mathrm{M}}_{1,0}$.

Example 2.3.3. In the case of an EV with valence $N = 6$, using the construction in Theorem 2.3.2 we obtain $W^T = \mathrm{Span}\{\mathbf{k}_1, \mathbf{k}_2, \mathbf{k}_3, \mathbf{k}_4\}$, where

$$\widehat{\mathbf{k}}_1^T = (1, 0, -1, -1, 0, 1), \quad \widehat{\mathbf{k}}_2^T = (-1, -1, 0, 1, 1, 0).$$

Hence, from (2.15) we compute the coefficients for the orthogonal projection which are

$$\mu_1 = \frac{3}{110}, \quad \mu_2 = -\frac{3}{110}, \quad \mu_3 = \frac{3}{220}, \quad \mu_4 = 0,$$

and substituting them in (2.16) we obtain the weights for the mask $\mathrm{M}_{1,0}$

$$\mathrm{M}_{1,0} = \frac{1}{660}[360, 76, 58, 22, 4, 22, 58, 19, 10, 1, 1, 10, 19]^T.$$

2.3.2 Bézier masks of second order

Relation (2.5), i.e. the equation for the G^1 constraints among Bézier points concerning second order derivatives, can be solved both with respect to $\mathrm{M}_{1,1}$ or $\mathrm{M}_{2,0}$. One approach, that is solving (2.5) w.r.t. the mask $\mathrm{M}_{1,1}$, will lead to a circulant linear system which presents different characteristics depending on the odd or even valence N of the EV to which it is attached. On the other hand, if we solve (2.5) w.r.t. $\mathrm{M}_{2,0}$, we end up with a direct formula for the solution. Let us investigate the two cases.

2.3.2.1 Deriving $\mathrm{M}_{1,1}$ by assigning $\mathrm{M}_{2,0}$: circulant system approach

We start by defining the mask $\mathrm{M}_{2,0}$, which is done differently for odd and even valence. Similarly to (2.12), since $\mathrm{M}_{1,1}^{(0)} = C\,\mathrm{M}_{1,1}$, (2.5) translates into the following linear system:

$$(I + C)\mathrm{M}_{1,1} = \mathbf{d}\,, \quad \text{where} \quad \mathbf{d} = \frac{1}{5}\left(a_0\underline{\mathrm{M}}_{0,0} - 5(a_0 - 2)\mathrm{M}_{1,0} + 4a_0\mathrm{M}_{2,0}\right). \tag{2.17}$$

The Bézier points involved in this relation are highlighted in Figure 2.3-(b). Let $B = I + C$ be the matrix in (2.17). To solve the system we have to distinguish the cases N odd and N even.

Case N odd. In this case, from (2.10), replacing the masks we have we get to

$$\mathrm{M}_{2,0} = [\delta_\bullet, \boldsymbol{\delta}, \boldsymbol{\delta}']^T = \frac{1}{10}\left(-\underline{\mathrm{M}}_{0,0} + 5\mathrm{M}_{1,0} + 10\underline{\mathrm{M}}_{3,0} - 5\underline{\mathrm{M}}_{4,0} + \underline{\mathrm{M}}_{5,0}\right). \tag{2.18}$$

We have the following:

Proposition 2.3.4. *The matrix $B = I + C$ is invertible with $B^{-1} = \mathrm{diag}\left(\frac{1}{2}, \widehat{B}^{-1}, \widehat{B}^{-1}\right)$, $\widehat{B} = \widehat{I} + \widehat{C}$ and*

$$\widehat{B}^{-1} = \frac{1}{2}\sum_{j=1}^{N}(-1)^{j-1}\widehat{C}^{j-1}. \tag{2.19}$$

Proof. It follows from

$$\left(\sum_{j=1}^{N}(-1)^{j-1}\widehat{C}^{j-1}\right)\left(\widehat{I}+\widehat{C}\right)=\sum_{j=1}^{N}(-1)^{j-1}\widehat{C}^{j-1}+\sum_{j=1}^{N}(-1)^{j-1}\widehat{C}^{j}$$
$$=\sum_{j=1}^{N}(-1)^{j-1}\widehat{C}^{j-1}+\sum_{k=2}^{N+1}(-1)^{k-2}\widehat{C}^{k-1}=\widehat{C}^{0}+\widehat{C}^{N}=2\widehat{I}.$$

□

By using (2.19) are now able to compute explicitly $\mathrm{M}_{1,1}$.

Corollary 2.3.5. *The weights for the mask* $\mathrm{M}_{1,1}=[\gamma_\bullet,\boldsymbol{\gamma},\boldsymbol{\gamma'}]^T$ *in the odd case are given in terms of the masks* $\underline{\mathrm{M}}_{0,0}=[\underline{\alpha}_\bullet,\underline{\boldsymbol{\alpha}},\underline{\boldsymbol{\alpha}}']^T$, $\mathrm{M}_{1,0}=[\beta_\bullet,\boldsymbol{\beta},\boldsymbol{\beta'}]^T$ *and* $\mathrm{M}_{2,0}=[\delta_\bullet,\boldsymbol{\delta},\boldsymbol{\delta'}]^T$ *by the following relations:*

$$\gamma_\bullet=\frac{1}{5}(5-2a_0)\underline{\alpha}_\bullet+\frac{2}{5}a_0\delta_\bullet,$$
$$\gamma_i=\frac{1}{10}a_0\sum_{j=1}^{N}(-1)^{j-1}\underline{\alpha}_{i+j-1}-\frac{1}{2}(a_0-2)\sum_{j=1}^{N}(-1)^{j-1}\beta_{i+j-1}+\frac{2}{5}a_0\sum_{j=1}^{N}(-1)^{j-1}\delta_{i+j-1},$$
$$\gamma'_i=\frac{1}{10}a_0\sum_{j=1}^{N}(-1)^{j-1}\underline{\alpha}'_{i+j-1}-\frac{1}{2}(a_0-2)\sum_{j=1}^{N}(-1)^{j-1}\beta'_{i+j-1}+\frac{2}{5}a_0\sum_{j=1}^{N}(-1)^{j-1}\delta'_{i+j-1},$$

with $i=1,\dots,N$ *and where the indices* $i+j-1$ *are understood* mod N *when* $i+j-1>N$.

Proof. Since from (2.17) it results that $\mathrm{M}_{1,1}=B^{-1}\mathbf{d}$, the proof is obtained by plugging in (2.19). □

Case N ***even.*** If we are in presence of an EV with even valence, the matrix B is not invertible and results $\mathrm{corank}(B)=2$, due to $\mathrm{corank}(\widehat{B})=1$. In fact, substituting $\lambda=-1$ in (2.13) it results that $\det(\widehat{B})=0$ and $\widehat{B}_{1,1}=1$, where $\widehat{B}_{1,1}$ denotes the minor of $\widehat{B}$ obtained removing the first row and column and since its cofactor is an upper triangular matrix with only ones on the diagonal. This implies $\mathrm{corank}(B)=2$. Hence, we must add two extra constraints in order to have a solution for the system; the obtained solution, also in this case, has to be meant in the least squares sense. First of all we must check if system (2.17) admits a solution.

Denoting

$$Y=(\mathbf{w}_1\ \mathbf{w}_2),$$

where

$$\mathbf{w}_1=(0,\widehat{\mathbf{w}}^T,\mathbf{0})^T,\quad \mathbf{w}_2=(0,\mathbf{0},\widehat{\mathbf{w}}^T)^T,\quad \widehat{\mathbf{w}}=(1,-1,1,\dots,-1)^T,$$

it follows that

$$\mathrm{Im}(Y^T)=\mathrm{Span}\{\mathbf{w}_1,\mathbf{w}_2\}=\mathrm{Leftker}(B);$$

in fact, it easy to notice that $Y^TB=\mathbf{0}$. Then, the system (2.17) admits a solution if and only if $\mathbf{d}\in\mathrm{Im}(B)$ or, equivalently,

$$Y^T\mathbf{d}=\mathbf{0}. \tag{2.20}$$

Relation (2.20) is known as *Vertex Enclosure Relation* (V.E.R.).
Let now denote

$$\mathrm{M}_{2,0}^{\mathrm{even}} = [\delta_{\bullet}^{\mathrm{even}}, \boldsymbol{\delta}^{\mathrm{even}}, \boldsymbol{\delta}'^{\mathrm{even}}]^T = \frac{1}{10}\left(-\underline{\mathrm{M}}_{0,0} + 5\mathrm{M}_{1,0} + 10\underline{\mathrm{M}}_{3,0} - 5\underline{\mathrm{M}}_{4,0} + \underline{\mathrm{M}}_{5,0}\right).$$

Checking property (2.20) for the masks $\underline{\mathrm{M}}_{0,0}$, $\mathrm{M}_{1,0}$ and $\mathrm{M}_{2,0}^{\mathrm{even}}$ in (2.17), we notice that $Y^T\underline{\mathrm{M}}_{0,0} = \mathbf{0}$, $Y^T\mathrm{M}_{1,0} = \mathbf{0}$ but $Y^T\mathrm{M}_{2,0}^{\mathrm{even}} \neq \mathbf{0}$. Thus, in order to have a solution for system (2.17) we need to compute a new mask $\mathrm{M}_{2,0} = [\delta_{\bullet}, \boldsymbol{\delta}, \boldsymbol{\delta}']^T$ ensuring (2.20). To do this we notice that

$$\mathrm{Im}(B) = \mathrm{Leftker}(\mathrm{Im}(Y));$$

in fact, it follows that $BY = \mathbf{0}$ due to the particular structure of B. Moreover, as for $\mathrm{M}_{1,0}$, we want that the mask $\mathrm{M}_{2,0}$ results close to $\mathrm{M}_{2,0}^{\mathrm{even}}$. Therefore we compute the new mask $\mathrm{M}_{2,0}$ by projecting its difference with $\mathrm{M}_{2,0}^{\mathrm{even}}$ onto the left kernel B and forcing the V.E.R to be verified; since we decided to operate an orthogonal projection, the sought mask $\mathrm{M}_{2,0}$ is obtained imposing

$$\langle \mathrm{M}_{2,0} - \mathrm{M}_{2,0}^{\mathrm{even}} \,|\, \mathbf{r}_i \rangle = 0, \quad i = 1, \ldots, 2N+1,$$

where $\mathbf{r}_i$ are the rows of B i.e. the vectors of one basis of $\mathrm{Leftker}(\mathrm{Im}(Y))$. Using the bilinearity of the scalar product we can rewrite the projection as

$$\langle \mathrm{M}_{2,0} \,|\, \mathbf{r}_i \rangle = \langle \mathrm{M}_{2,0}^{\mathrm{even}} \,|\, \mathbf{r}_i \rangle,$$

which is in fact

$$B\,\mathrm{M}_{2,0} = B\,\mathrm{M}_{2,0}^{\mathrm{even}}.$$

Finally, forcing the V.E.R. to be verified by the desired mask $\mathbf{w}_1^T\mathrm{M}_{2,0} = 0$ and $\mathbf{w}_2^T\mathrm{M}_{2,0} = 0$, we are able to condense the previous relations in the following linear system

$$\widetilde{B}\,\mathrm{M}_{2,0} = \widetilde{\mathbf{f}},$$

where

$$\widetilde{B} = \begin{pmatrix} B \\ \mathbf{w}_1^T \\ \mathbf{w}_2^T \end{pmatrix} \quad \text{and} \quad \widetilde{\mathbf{f}} = \begin{pmatrix} B\,\mathrm{M}_{2,0}^{\mathrm{even}} \\ 0 \\ 0 \end{pmatrix}. \tag{2.21}$$

Reordering the rows of $\widetilde{B}$ we are able to achieve a diagonal block structure for the matrix in (2.21) with blocks $(2, \bar{B}, \bar{B})$ which brings to the following

Proposition 2.3.6. *Let* $\bar{B} = \begin{pmatrix} \widehat{B} \\ \widehat{\mathbf{w}}^T \end{pmatrix}$. *The pseudo-inverse matrix of* $\bar{B}$ *is a block matrix* $\bar{B}^+ = (B_1|B_2)$ *where*

$$B_1 = \frac{1}{2N}\sum_{j=1}^{N}(-1)^{j-1}(N-2j+1)\widehat{C}^{j-1} \quad \textit{and} \quad B_2 = \frac{1}{N}\widehat{\mathbf{w}}, \tag{2.22}$$

with $B_1 \in \mathbb{R}^{N\times N}$ *and* $B_2 \in \mathbb{R}^{N\times 1}$.

Proof. By definition of pseudo-invers matrix, $\bar{B}^+ = (\bar{B}^T\bar{B})^{-1}\bar{B}^T$; therefore it suffices to show that

$$(\bar{B}^T\bar{B})\,(B_1|B_2) = \bar{B}^T.$$

Due to the block structure of $\bar{B}$ we have

$$\bar{B}^T\bar{B} = \widehat{B}^T\widehat{B} + \widehat{\mathbf{w}}\widehat{\mathbf{w}}^T = 2\widehat{I} + \widehat{C} + \widehat{C}^{-1} + \widehat{\mathbf{w}}\widehat{\mathbf{w}}^T.$$

Noticing that

$$\widehat{\mathbf{w}}\widehat{\mathbf{w}}^T = \sum_{j=1}^{N}(-1)^{j-1}j\widehat{C}^{j-1},$$

it results

$$\widehat{\mathbf{w}}\widehat{\mathbf{w}}^T B_1 = \sum_{j=1}^{N} N - 2j + 1 = 0$$

since $\widehat{C}\widehat{\mathbf{w}} = \widehat{C}^{-1}\widehat{\mathbf{w}} = -\widehat{\mathbf{w}}$. This implies that

$$\left(2\widehat{I} + \widehat{C} + \widehat{C}^{-1}\right)\widehat{\mathbf{w}} = 0. \tag{2.23}$$

Let

$$H = \sum_{j=1}^{N}(-1)^{j-1}j\widehat{C}^{j-1}.$$

From (2.22) we deduce that

$$\begin{aligned}(\bar{B}^T\bar{B})B_1 = \left(2\widehat{I} + \widehat{C} + \widehat{C}^{-1} + \widehat{\mathbf{w}}\widehat{\mathbf{w}}^T\right)B_1 &= \left(2\widehat{I} + \widehat{C} + \widehat{C}^{-1}\right)\left(\frac{1}{2}\widehat{\mathbf{w}}\widehat{\mathbf{w}}^T - \frac{1}{N}H + \frac{1}{2N}\widehat{\mathbf{w}}\widehat{\mathbf{w}}^T\right)\\ &= -\frac{1}{N}\left(2H + \widehat{C}H + \widehat{C}^{-1}H\right),\end{aligned} \tag{2.24}$$

and using the notation in (2.11), we have

$$H = \mathrm{Circ}(1,-2,3,\ldots,-N)\,,\quad \widehat{C}H = \mathrm{Circ}(-N,1,-2,\ldots,N-1)\,,\quad \widehat{C}^{-1}H = \mathrm{Circ}(-2,3,-4,\ldots,1)\,,$$

from which

$$2H + \widehat{C}H + \widehat{C}^{-1}H = -N\mathrm{Circ}(1,0,0,\ldots,1) = -N\widehat{B}^T. \tag{2.25}$$

Substituting (2.25) in (2.24) we arrive at

$$(\bar{B}^T\bar{B})B_1 = \widehat{B}^T.$$

In a similar way, using the identity (2.23),

$$(\bar{B}^T\bar{B})B_2 = \left(2\widehat{I} + \widehat{C} + \widehat{C}^{-1} + \widehat{\mathbf{w}}\widehat{\mathbf{w}}^T\right)\frac{1}{N}\widehat{\mathbf{w}} = \frac{1}{N}\widehat{\mathbf{w}}\widehat{\mathbf{w}}^T\widehat{\mathbf{w}} = \widehat{\mathbf{w}}$$

since $\widehat{\mathbf{w}}^T\widehat{\mathbf{w}} = N$. □

From Proposition 2.3.6, we are now able to compute explicitly the new mask $\mathrm{M}_{2,0}$ ensuring the V.E.R as follow:

Corollary 2.3.7. *The mask* $\mathrm{M}_{2,0} = [\delta_\bullet, \boldsymbol{\delta}, \boldsymbol{\delta'}]^T$ *ensuring* (2.20) *is obtained from* $\mathrm{M}_{2,0}^{\text{even}} = [\delta_\bullet^{\text{even}}, \boldsymbol{\delta}^{\text{even}}, \boldsymbol{\delta}'^{\text{even}}]^T$ *as follows:*

$$\begin{aligned}
\delta_\bullet &= \delta_\bullet^{\text{even}},\\
\delta_i &= \frac{1}{2N}\sum_{j=1}^{N}(-1)^{j-1}(N-2j+1)\left(\delta_{i+j-1}^{\text{even}} + \delta_{i+j}^{\text{even}}\right),\\
\delta_i' &= \frac{1}{2N}\sum_{j=1}^{N}(-1)^{j-1}(N-2j+1)\left(\delta_{i+j-1}'^{\text{even}} + \delta_{i+j}'^{\text{even}}\right),
\end{aligned}$$

with $i = 1, \dots, N$ *and where the indices* $i+j-1$ *and* $i+j$ *have to be meant respectively* mod N *when* $i+j-1 > N$ *and* $i+j > N$.

Proof. Since $\mathrm{M}_{2,0} = \widetilde{B}^{+}\widetilde{\mathbf{f}}$ the proof is obtained by the explicit formulas (2.21) and (2.22), using the block structure of $\widetilde{B}$. □

Hence, using mask $\mathrm{M}_{2,0}$ we are sure to have a solution for (2.17). Now, since $\operatorname{corank}(B) = 2$, to compute $\mathrm{M}_{1,1}$ we need to add two extra constraints: once more we want to obtain a mask $\mathrm{M}_{1,1}$ the closest possible to $\underline{\mathrm{M}}_{1,1}$ and it is ensure projecting their difference onto $\operatorname{Im}(Y^T) = \operatorname{Leftker}(B)$. Like in the previous cases, we operate an orthogonal projection i.e.

$$\langle \mathrm{M}_{1,1} - \underline{\mathrm{M}}_{1,1} \,|\, \mathbf{w}_i \rangle = 0, \quad i = 1, 2,$$

which translates into

$$\langle \mathrm{M}_{1,1} \,|\, \mathbf{w}_i \rangle = \langle \underline{\mathrm{M}}_{1,1} \,|\, \mathbf{w}_i \rangle.$$

Joining these two additional relations to system (2.17) and denoting

$$\widetilde{\mathbf{d}} = \begin{pmatrix} \mathbf{d} \\ \langle \underline{\mathrm{M}}_{1,1} \,|\, \mathbf{w}_1 \rangle \\ \langle \underline{\mathrm{M}}_{1,1} \,|\, \mathbf{w}_2 \rangle \end{pmatrix}, \tag{2.26}$$

the mask for $\mathrm{M}_{1,1}$ satisfies the relation

$$\widetilde{B}\,\mathrm{M}_{1,1} = \widetilde{\mathbf{d}}.$$

Similarly to Corollary 2.3.7, we have:

Corollary 2.3.8. *The mask* $\mathrm{M}_{1,1} = [\gamma_\bullet, \boldsymbol{\gamma}, \boldsymbol{\gamma'}]^T$ *in the even case is given in terms of* $\underline{\mathrm{M}}_{0,0} = [\underline{\alpha}_\bullet, \underline{\boldsymbol{\alpha}}, \underline{\boldsymbol{\alpha}'}]^T$, $\mathrm{M}_{1,0} = [\beta_\bullet, \boldsymbol{\beta}, \boldsymbol{\beta'}]^T$, $\underline{\mathrm{M}}_{1,1} = [\underline{\gamma}_\bullet, \underline{\boldsymbol{\gamma}}, \underline{\boldsymbol{\gamma}'}]^T$ *and* $\mathrm{M}_{2,0} = [\delta_\bullet, \boldsymbol{\delta}, \boldsymbol{\delta'}]^T$ *by the following relations:*

$$\gamma_\bullet = \frac{1}{5}(5-2a_0)\underline{\alpha}_\bullet + \frac{2}{5}a_0\delta_\bullet$$

$$\gamma_i = \frac{1}{10N}a_0\sum_{j=1}^{N}(-1)^{j-1}(N-2j+1)\underline{\alpha}_{i+j-1} - \frac{1}{2N}(a_0-2)\sum_{j=1}^{N}(-1)^{j-1}(N-2j+1)\beta_{i+j-1}$$
$$+ \frac{2}{5N}a_0\sum_{j=1}^{N}(-1)^{j-1}(N-2j+1)\delta_{i+j-1} + \frac{1}{N}\widehat{\mathrm{w}}_i\left\langle\underline{\boldsymbol{\gamma}}\,|\,\widehat{\mathbf{w}}\right\rangle,$$

$$\gamma'_i = \frac{1}{10N}a_0\sum_{j=1}^{N}(-1)^{j-1}(N-2j+1)\underline{\alpha}'_{i+j-1} - \frac{1}{2N}(a_0-2)\sum_{j=1}^{N}(-1)^{j-1}(N-2j+1)\beta'_{i+j-1}$$
$$+ \frac{2}{5N}a_0\sum_{j=1}^{N}(-1)^{j-1}(N-2j+1)\delta'_{i+j-1} + \frac{1}{N}\widehat{\mathrm{w}}_i\left\langle\underline{\boldsymbol{\gamma}'}\,|\,\widehat{\mathbf{w}}\right\rangle,$$

with $i = 1, \dots, N$ *and where the indices* $i+j-1$ *have to be meant* $\bmod N$ *when* $i+j-1 > N$.

Proof. Because $\mathrm{M}_{1,1} = \widetilde{B}^{+}\widetilde{\mathbf{d}}$, the proof is obtained using (2.22), (2.26) and the block shape of $\widetilde{B}$. □

Example 2.3.9. For an EV with valence $N = 6$, and using the formulas in Corollary 2.3.7, the mask $\mathrm{M}_{2,0}$ satisfying (2.20) is given by

$$\mathrm{M}_{2,0} = \frac{1}{1320}[592, 294, 130, -8, 18, -8, 130, 79, 8, -1, -1, 8, 79]^T. \tag{2.27}$$

Thus, using (2.27) in Corollary 2.3.8, the cross derivative mask is

$$\mathrm{M}_{1,1} = \frac{1}{9900}[5016, 1512, 1512, 258, 144, 144, 258, 604, 185, 43, -4, 43, 185]^T.$$

We notice that negative weights may occur.

2.3.2.2 Deriving $\mathrm{M}_{2,0}$ by assigning $\mathrm{M}_{1,1}$: direct approach based on ACC_5

If we assume the cross derivatives to be fixed, substituting the known masks in (2.5), we directly obtain

$$\mathrm{M}_{2,0} = \frac{1}{4a_0}\left(-a_0\underline{\mathrm{M}}_{0,0} + 5(a_0-2)\mathrm{M}_{1,0} + 5(\underline{\mathrm{M}}_{1,1} + \underline{\mathrm{M}}^{(0)}_{1,1})\right), \tag{2.28}$$

that is well defined since $a_0 \neq 0$ for $N \neq 4$, which is the case for the EVs we are analyzing. In Figure 2.3-(c) is figured the location of the treated control points.

2.3.3 Third and fourth order Bézier masks

First of all, depending on the previous solving strategies, from (2.10) we define the new mask

$$\mathrm{M}_{3,0} = \frac{1}{10}\left(\underline{\mathrm{M}}_{0,0} - 5\mathrm{M}_{1,0} + 10\mathrm{M}_{2,0} + 5\underline{\mathrm{M}}_{4,0} - \underline{\mathrm{M}}_{5,0}\right).$$

When we modify the control point related to the mask $\mathrm{M}_{3,0}$ the regularity relation for C^2 smoothness in Bézier surfaces across a vertex does not hold anymore. The resulting surface will be C^1 smooth around regular vertices linked to EVs instead of the initial C^2 regularity. From (2.6), it follows that

$$\mathrm{M}_{2,1} + \mathrm{M}^{(0)}_{1,2} = \mathbf{r}, \quad \text{where} \quad \mathbf{r} = \frac{1}{10}\left(-a_0\underline{\mathrm{M}}_{0,0} + 5a_0\mathrm{M}_{1,0} - 10(a_0-2)\mathrm{M}_{2,0} + 6a_0\mathrm{M}_{3,0}\right), \tag{2.29}$$

where Figure 2.3-(d) presents the location of the involved Bézier points. By using the reparametrization

$$\mathrm{M}^{(0)}_{1,2} - \underline{\mathrm{M}}^{(0)}_{1,2} = \theta\left(\mathrm{M}_{2,1} - \underline{\mathrm{M}}_{2,1}\right)$$

we obtain the one parameter family of solutions of (2.29)

$$\begin{cases} \mathrm{M}_{2,1} = \dfrac{1}{\theta+1}\left(\mathbf{r} + \theta\underline{\mathrm{M}}_{2,1} - \underline{\mathrm{M}}^{(0)}_{1,2}\right), \\ \mathrm{M}^{(0)}_{1,2} = \mathbf{r}\dfrac{\theta}{\theta+1} - \dfrac{1}{\theta+1}\left(\theta\underline{\mathrm{M}}_{2,1} - \underline{\mathrm{M}}^{(0)}_{1,2}\right), \end{cases} \tag{2.30}$$

where the parameter $\theta \in \mathbb{R} \setminus \{-1\}$ has to be fixed. If we take $\theta = 1$, (2.30) will return a symmetric solution; indeed, this choice reflects the fact that we are looking for a solution by projecting orthogonally the ACC_5 masks $(\underline{\mathrm{M}}_{2,1}, \underline{\mathrm{M}}^{(0)}_{1,2})$ onto the G^1 solution space. For all the other values of $\theta \setminus \{-1, 1\}$ we will obtain a non symmetric solution for (2.30); in particular, for $\theta = 0$, we obtain the asymmetric solution by projecting along the direction $\underline{\mathrm{M}}^{(0)}_{1,2}$. Moreover, with $\theta = \pm\infty$ we mean the projection along the direction of $\underline{\mathrm{M}}_{2,1}$; as expected from (2.30), we don't have any solution for the value $\theta = -1$ because it translates into a projection along a parallel direction to the G^1 solution space. For the fourth order Bézier masks, from (2.7) we have

$$\mathrm{M}_{3,1} + \mathrm{M}^{(0)}_{1,3} = \widetilde{\mathbf{r}}, \quad \text{where} \quad \widetilde{\mathbf{r}} = \frac{1}{10}\left(a_0\underline{\mathrm{M}}_{0,0} - 5a_0\mathrm{M}_{1,0} + 10a_0\mathrm{M}_{2,0} - 10(a_0-2)\mathrm{M}_{3,0} + 4a_0\underline{\mathrm{M}}_{4,0}\right), \tag{2.31}$$

which can be solved as above, replacing $\mathbf{r}$ in (2.30) with $\widetilde{\mathbf{r}}$ in (2.31); the positioning of the considered control points is shown in Figure 2.3-(e).

2.3.4 Treatment of boundaries

Meshes are not always closed and, therefore, they can possess boundaries in which EVs can eventually lay. Bézier control points returning G^1 smoothness around boundary regions can be obtained by applying (2.4) to (2.10) starting from the boundary masks presented in [LS08], after degree elevation. Since these masks ensure the interpolation of boundary edges, we are not going to modify them; we will recover the G^1 regularity by imposing G^1 relations only in the inner regions and using the ACC_5 boundary masks as initial and final value (see Figure 2.4). Let κ be the numbers of patches sharing the boundary vertex: for the Bézier masks of order

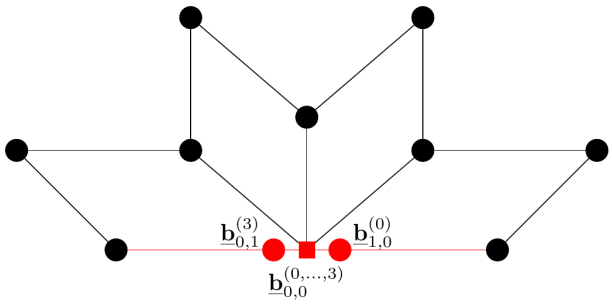


Figure 2.4. Example of a boundary EV with four incident faces. Red square: Bézier point for the border EV $\underline{\mathbf{b}}_{1,0}^{(0,\dots,3)}$. Red circles: initial ($\underline{\mathbf{b}}_{1,0}^{(0)}$) and final ($\underline{\mathbf{b}}_{0,1}^{(3)}$) Bézier points for the G^1 relations around a boundary EV.

one, imposing (2.4) $(\kappa-1)$ times across the inner extraordinary edges we obtain, for $\kappa \geq 4$, the following linear system:

$$\widehat{A}\,\mathbf{M}_{1,0} = \mathbf{h}, \quad \text{where} \quad \mathbf{h} = \begin{pmatrix} (2-a_0)\underline{\mathrm{F}}_{0,0}^{(1)} - \underline{\mathrm{F}}_{1,0}^{(0)} \\ (2-a_0)\underline{\mathrm{F}}_{0,0}^{(2)} \\ \vdots \\ (2-a_0)\underline{\mathrm{F}}_{0,0}^{(\kappa-2)} \\ (2-a_0)\underline{\mathrm{F}}_{0,0}^{(\kappa-1)} - \underline{\mathrm{F}}_{0,1}^{(\kappa-1)} \end{pmatrix}, \tag{2.32}$$

$\mathbf{M}_{1,0}=\left(\mathrm{M}_{1,0}^{(1)},\dots,\mathrm{M}_{1,0}^{(\kappa-1)}\right)$, $\underline{\mathrm{F}}_{0,0}^{(1,\dots,\kappa-1)}$, $\underline{\mathrm{F}}_{1,0}^{(0)}$ and $\underline{\mathrm{F}}_{0,1}^{(\kappa-1)}$ are, respectively, the ACC_5 masks for the boundary (or frontier) control points $\underline{\mathbf{b}}_{0,0}^{(1,\dots,\kappa-1)}$, $\underline{\mathbf{b}}_{1,0}^{(0)}$ and $\underline{\mathbf{b}}_{0,1}^{(\kappa-1)}$, and $\widehat{A} = \mathrm{Tridiag}(-\boldsymbol{a_0}, \mathbf{1}, \mathbf{1}) \in \mathbb{R}^{(\kappa-1)\times(\kappa-1)}$, that is a tridiagonal matrix with entries $-a_0$ along the main diagonal and entries 1 on the first upper and lower diagonal. For $\kappa = 3$ it results that $\widehat{A} = -a_0\widehat{I} + \widehat{C}$ and the vector $\mathbf{h}$ has entries $(2-a_0)\underline{\mathrm{F}}_{0,0}^{(1)} - \underline{\mathrm{F}}_{1,0}^{(0)}$ and $(2-a_0)\underline{\mathrm{F}}_{0,0}^{(\kappa-1)} - \underline{\mathrm{F}}_{0,1}^{(\kappa-1)}$. System (2.32) can be solved with a similar strategy like in Section 2.3.1. Regarding the second order masks, even in this case, from (2.10) and as in (2.18), we have

$$\mathrm{M}_{2,0}^{(i)} = \frac{1}{10}\left(-\underline{\mathrm{F}}_{0,0}^{(i)} + 5\mathrm{M}_{1,0}^{(i)} + 10\underline{\mathrm{M}}_{3,0}^{(i)} - 5\underline{\mathrm{M}}_{4,0}^{(i)} + \underline{\mathrm{M}}_{5,0}^{(i)}\right), \quad i = 1,\dots,\kappa-1.$$

Now, for the cross derivatives, imposing (2.5) $(\kappa-1)$ times we get the system:

$$\begin{pmatrix} 1 & 1 & 0 & \dots & 0 \\ & 1 & 1 & & \\ & & & \ddots & \\ 0 & \dots & 0 & 1 & 1 \end{pmatrix} \mathbf{M}_{1,1} = \mathbf{g}, \tag{2.33}$$

where

$$\mathbf{g} = \frac{1}{5}\left(a_0\underline{\mathrm{F}}_{0,0}^{(i)} + 5(2-a_0)\mathrm{M}_{1,0}^{(i)} + 4a_0\mathrm{M}_{2,0}^{(i)}\right), \quad i = 1,\dots,\kappa-1,$$

and $\mathbf{M}_{1,1}=\left(\mathrm{M}_{1,1}^{(0)},\dots,\mathrm{M}_{1,1}^{(\kappa-1)}\right)$. Let $R \in \mathbb{R}^{(\kappa-1)\times\kappa}$ be the rectangular matrix in (2.33); since $\mathrm{corank}(R) = 1$ we have a degree of freedom for the solution. If we want the masks to be close

to ACC_5, we can fix this free degree, for example, projecting orthogonally the solution of (2.33) onto the ACC_5 space; this translates into the solution

$$\mathbf{M}_{1,1} = \underline{\mathbf{M}}_{1,1} + R^T \boldsymbol{\rho},$$

where $\boldsymbol{\rho}$ is the vector containing the coefficients for the orthogonal projection given by

$$R\,R^T \boldsymbol{\rho} = -R\,\underline{\mathbf{M}}_{1,1} + \mathbf{g}.$$

The third and fourth order Bézier points are obtained in the same way as the case for an inner EV using (2.29) and (2.31) along the inner extraordinary edges.

2.4 Analysis of the solutions and numerical results

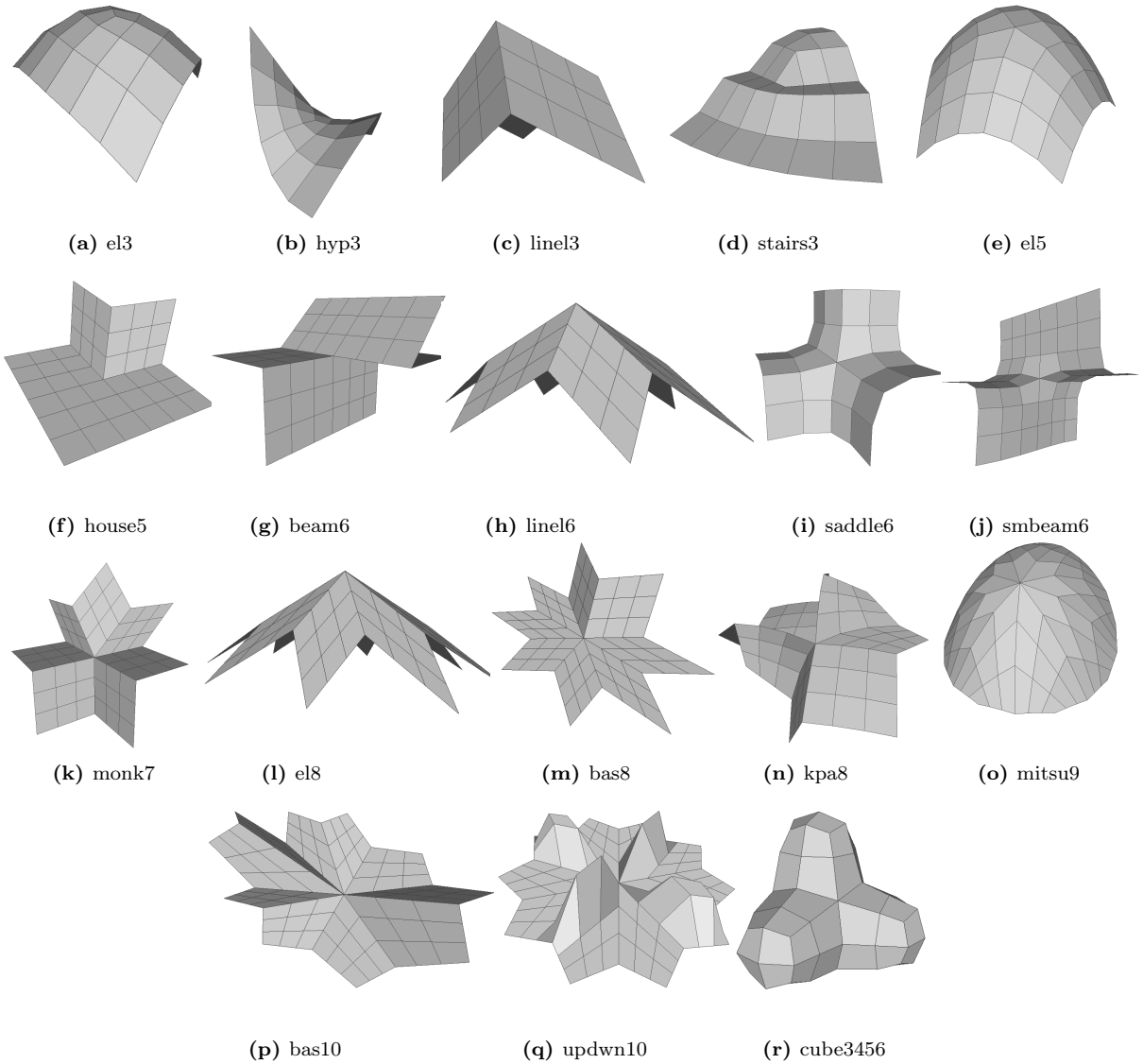
From the constructions presented in Section 2.3 it turns out that we can solve system (2.4)-(2.10) in multiple ways, obtaining a family of G^1 schemes. We are now going to analyze this family and identify the procedure returning the most regular surface. First, we will describe the results obtained using a set of test meshes [Pet] (cf. Figure 2.5) which contains shapes with EVs of different valence and particular geometric structure; from the results obtained by this procedure we will find out the element of the family returning the smoothest surface. Then, we will check the strength of the previous-found best strategy by applying it to the more complex meshes in Figure 2.12 and showing numerical results for the G^1 continuity.

2.4.1 Comparing the different schemes

Let us call CS (circulant system) the solving strategy in Section 2.3.2.1, NCS (no circulant system) the strategy in 2.3.2.2 and S (symmetric) and AS (asymmetric) the solutions for the higher derivatives obtained in 2.3.3 respectively with $\theta = 1$ and $\theta = 0$; then, we can define four global solving strategies as: NCS-S, NCS-AS, CS-AS and CS-S. To identify the smoothest surface we carried out an isophote and curvature analysis together with numerical evaluation of the G^2 errors on the surfaces obtained by each strategy. In Figures 2.6 and 2.7 the results for two meshes with EV of valence $N = 5$ and $N = 8$ are shown. In strategy NCS-S we notice that for both valences, isophotes are deviated away from the EV creating an unwanted white spot around it, as also underlined by the curvatures. For the asymmetric strategies NCS-AS and CS-AS we easily realize that the asymmetry of the Bézier points translates into a rotation of the patches around the EVs, which implies irregularities in the isophotes and unpleasant curvatures. Instead, in strategy CS-S, we see that both isophotes and curvatures related to the two valences are smooth and regular in a neighborhood of the EVs. To evaluate numerically the results we use the quantities

$$[\![\widehat{\mathbf{n}}]\!] := \sqrt{\int_{\mathcal{E}} \|\widehat{\mathbf{n}}^L - \widehat{\mathbf{n}}^R\|_2^2}, \quad (\!|c_G|\!) := \max_{\mathbf{e}\in\mathcal{E}} \left(\max_{u\in\mathbf{e}} \left| c_G^L - c_G^R \right| \right), \quad (\!|c_M|\!) := \max_{\mathbf{e}\in\mathcal{E}} \left(\max_{u\in\mathbf{e}} \left| c_M^L - c_M^R \right| \right),$$

where $\int_{\mathcal{E}} := \sum_{\mathbf{e}\in\mathcal{E}} \int_{\mathbf{e}}$ with $\mathcal{E}$ the set of all edges $\mathbf{e}$ of the mesh $\mathcal{M}$ and $\widehat{\mathbf{n}}^L$, c_G^L, c_M^L, $\widehat{\mathbf{n}}^R$, c_G^R, c_M^R are respectively the unit normal vector, the Gauss curvature and the mean curvature of, respectively, the left and right patch sharing the common edge e. The G^1 quality is evaluated by $[\![\widehat{\mathbf{n}}]\!]$, which is a measure of the distance of our surface from the set of purely C^1 parametric



(a) el3 **(b)** hyp3 **(c)** linel3 **(d)** stairs3 **(e)** el5

(f) house5 **(g)** beam6 **(h)** linel6 **(i)** saddle6 **(j)** smbeam6

(k) monk7 **(l)** el8 **(m)** bas8 **(n)** kpa8 **(o)** mitsu9

(p) bas10 **(q)** updwn10 **(r)** cube3456

Figure 2.5. Benchmark meshes in [Pet]: the numbers in the names refers to the EVs valences of the meshes.

surfaces. Indeed, this measure is zero on any C^1 interface, while it expresses the discrepancy or "jump" of the (non-unit) normal vector when there is a discontinuity across an interface. Moreover, the quantities $(\!|c_G|\!)$ and $(\!|c_M|\!)$ return, respectively, the maximum difference between the Gauss curvatures and mean curvatures on the patch interfaces: Table 3.25 shows the values for these quantities of each solving strategy carried out on the test meshes [Pet] in Figure 2.5. At this point, using both the numerical and visual approach, we can state that in any case the asymmetric approach AS is never returning very regular surfaces; on the other hand we see that in most cases the best approach is given by the strategy CS-S. From the numerical results presented in Table 2.1, we notice that for meshes with very quick and sudden change of slope of patches around EVs, strategy NCS-S returns slightly better surfaces in terms of curvature compared to CS-S.

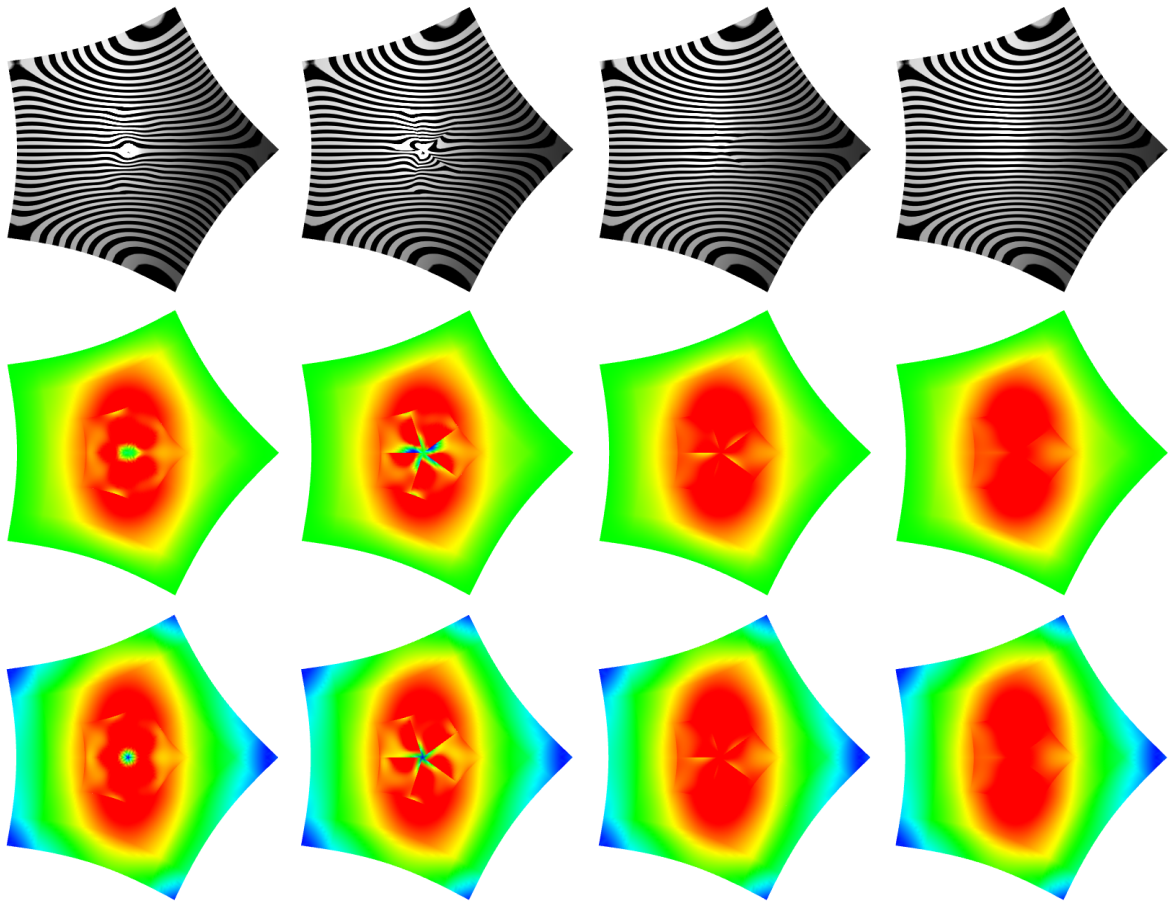


Figure 2.6. Mesh in Figure 2.5-(e). From left to right: solving strategies NCS-S, NCS-AS, CS-AS, CS-S. Upper row: isophotes. Middle row: Gauss curvature. Lower row: mean curvature.

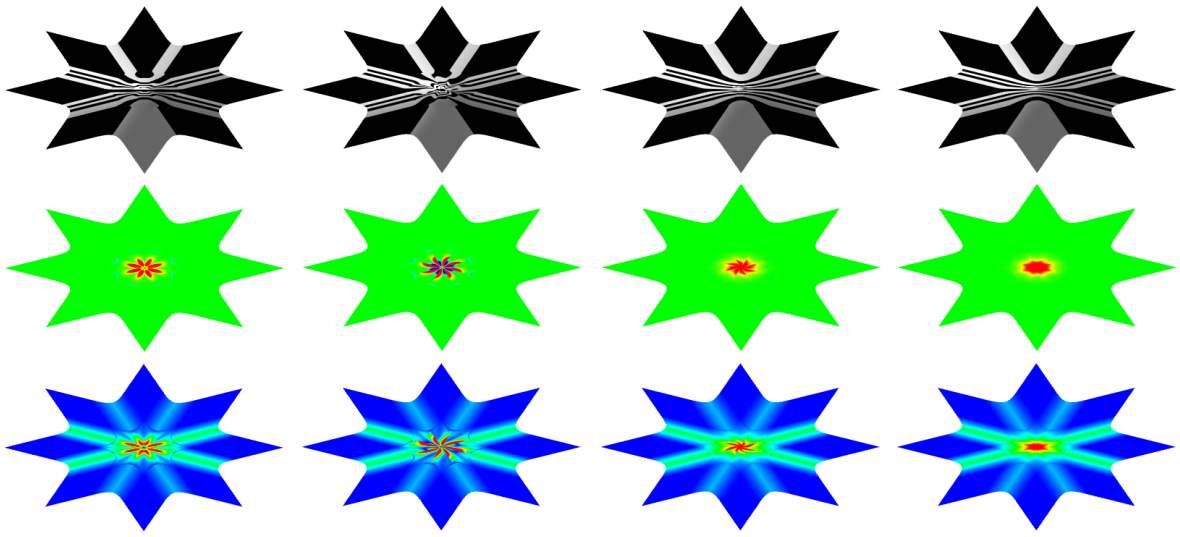


Figure 2.7. Mesh in Figure 2.5-(l). From left to right: solving strategies NCS-S, NCS-AS, CS-AS, CS-S. Upper row: isophotes. Middle row: Gauss curvature. Lower row: mean curvature.

		el3	hyp3	linel3	stairs3	el5	house5	beam6	linel6	saddle6
NCS-S	$[\![\hat{\mathbf{n}}]\!]$	4.4e-15	3.4e-15	8.5e-15	3.3e-15	9.9e-15	7.0e-16	2.2e-15	1.5e-14	3.9e-15
	$(\!\|c_G\|\!)$	1.9e-00	3.1e-00	3.4e-00	2.7e-00	1.2e-00	11.3e-00	84.7e-00	8.1e-00	1.3e-00
	$(\!\|c_M\|\!)$	0.9e-00	2.5e-00	0.3e-00	3.5e-00	0.7e-00	1.4e-00	2.4e-00	2.8e-00	0.5e-00
NCS-AS	$[\![\hat{\mathbf{n}}]\!]$	4.4e-15	3.4e-15	8.9e-15	3.3e-15	9.9e-15	7.5e-16	2.2e-15	1.5e-14	3.9e-15
	$(\!\|c_G\|\!)$	13.6e-00	3.4e-00	132.8e-00	11.0e-00	7.9e-00	16.5e-00	92.1e-00	75.2e-00	1.6e-00
	$(\!\|c_M\|\!)$	4.2e-00	2.7e-00	23.7e-00	7.0e-00	4.1e-00	4.9e-00	7.3e-00	23.7e-00	1.4e-00
CS-AS	$[\![\hat{\mathbf{n}}]\!]$	4.4e-15	3.4e-15	7.0e-15	3.3e-15	9.9e-15	1.0e-15	2.2e-15	1.5e-14	4.0e-15
	$(\!\|c_G\|\!)$	5.0e-00	1.3e-00	33.0e-00	4.0e-00	1.3e-00	3.7e-00	43.3e-00	16.0e-00	4.1e-00
	$(\!\|c_M\|\!)$	2.0e-00	1.4e-00	8.2e-00	4.2e-00	0.9e-00	1.5e-00	16.7e-00	5.7e-00	3.5e-00
CS-S	$[\![\hat{\mathbf{n}}]\!]$	4.4e-15	3.4e-15	6.9e-15	3.3e-15	9.9e-15	9.5e-16	2.6e-15	1.5e-14	4.4e-15
	$(\!\|c_G\|\!)$	0.8e-00	1.3e-00	6.8e-00	3.9e-00	0.1e-00	3.4e-00	17.4e-00	0.9e-00	1.5e-00
	$(\!\|c_M\|\!)$	0.4e-00	1.4e-00	0.6e-00	4.0e-00	0.1e-00	0.8e-00	0.9e-00	0.3e-00	0.6e-00

		smbeam6	monk7	el8	bas8	kpa8	mitsu9	bas10	updwn10	cube3456
NCS-S	$[\![\hat{\mathbf{n}}]\!]$	2.9e-15	6.2e-15	3.0e-14	5.6e-15	1.0e-14	1.9e-14	1.8e-14	1.0e-14	3.6e-14
	$(\!\|c_G\|\!)$	45.6e-00	8.3e-00	7.5e-00	7.7e-00	56.6e-00	1.6e-00	50.0e-00	70.4e-00	0.7e-00
	$(\!\|c_M\|\!)$	4.3e-00	6.4e-00	2.9e-00	5.5e-00	6.5e-00	1.4e-00	11.8e-00	19.1e-00	0.5e-00
NCS-AS	$[\![\hat{\mathbf{n}}]\!]$	2.9e-15	6.2e-15	3.0e-14	5.6e-15	1.0e-14	1.9e-14	1.8e-14	1.0e-14	3.2e-14
	$(\!\|c_G\|\!)$	49.5e-00	38.3e-00	107.5e-00	12.1e-00	63.0e-00	23.4e-00	91.8e-00	106.2e-00	3.3e-00
	$(\!\|c_M\|\!)$	5.1e-00	10.6e-00	40.8e-00	13.6e-00	17.6e-00	12.8e-00	27.6e-00	29.7e-00	1.9e-00
CS-AS	$[\![\hat{\mathbf{n}}]\!]$	2.9e-15	6.1e-15	2.6e-14	5.7e-15	1.1e-14	1.9e-14	1.8e-14	1.0e-14	4.0e-14
	$(\!\|c_G\|\!)$	35.1e-00	92.0e-00	43.6e-00	16.1e-00	27.6e-00	14.6e-00	87.9e-00	25.3e-00	1.3e-00
	$(\!\|c_M\|\!)$	7.8e-00	39.0e-00	16.6e-00	16.8e-00	10.7e-00	8.3e-00	49.5e-00	87.4e-00	1.9e-00
CS-S	$[\![\hat{\mathbf{n}}]\!]$	2.9e-15	6.1e-15	2.6e-14	5.7e-15	1.1e-14	1.9e-14	1.8e-14	1.1e-14	4.5e-14
	$(\!\|c_G\|\!)$	6.0e-00	91.4e-00	0.9e-00	15.1e-00	27.0e-00	1.0e-00	83.3e-00	24.5e-00	0.7e-00
	$(\!\|c_M\|\!)$	1.2e-00	38.8e-00	0.3e-00	15.8e-00	6.0e-00	1.0e-00	47.5e-00	84.4e-00	0.6e-00

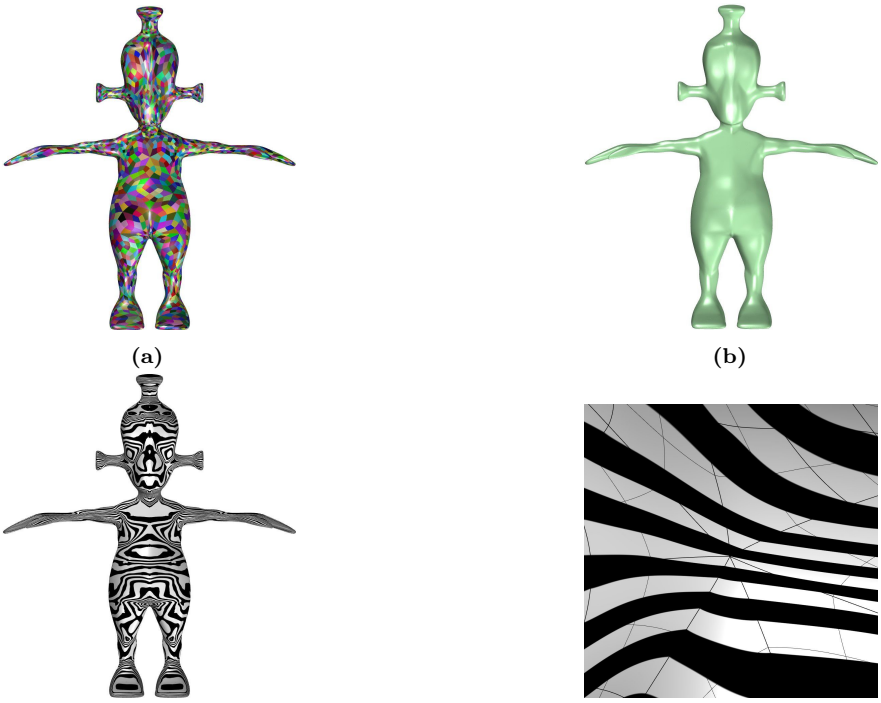
Table 2.1. G^1 smoothness $[\![\hat{\mathbf{n}}]\!]$, Gauss curvature difference $(\!|c_G|\!)$ and mean curvature difference $(\!|c_M|\!)$ obtained from strategies NCS-S, NCS-AS, CS-AS, CS-S for every test mesh in Figure 2.5.

2.4.2 Complex meshes

Using the best strategy CS-S obtained in Section 2.4.1, we developed a G^1 continuity analysis on the complex meshes in Figures 2.8 to 2.12; these meshes present a very complicate geometry together with a very high number of faces, vertices and, obviously, extraordinary vertices of several valence. In Table 2.2 we present a detailed description of each mesh: we show the number of EVs of any valence, the total number of EVs, the number of vertices and faces of the mesh and the numerical evaluation of the G^1 regularity. From the results in Table 3.25 and Table 2.2, we can affirm that this construction returns really smooth surfaces, with a G^1 error in the range of 10^{-11}-10^{-15}, i.e. numerically identical to zero.

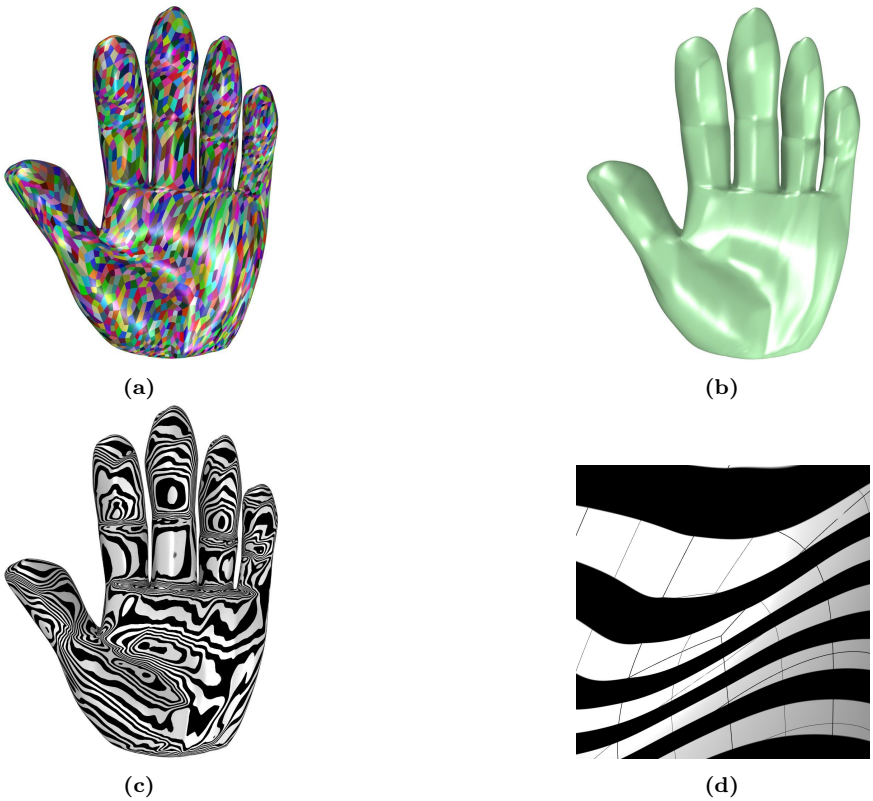
	N=**3,**	**5,**	**6,**	**7,**	**8,**	**9,**	**10,**	**11,**	**12,**	**16,**	**45**	**Total EVs**	**Vertices**	**Faces**	$[\![\hat{\mathbf{n}}]\!]$
alien	858,	82,	156,	102,	30,	6,	0,	0,	0,	0,	0	1234	5126	5124	9.0e-12
bird	1135,	147,	179,	84,	45,	14,	7,	10,	1,	0,	0	1622	6782	6780	2.7e-11
dinosaur	501,	57,	134,	34,	9,	6,	0,	0,	0,	0,	0	741	3002	3000	7.2e-12
disk	270,	90,	45,	0,	0,	0,	0,	0,	0,	0,	2	407	2214	1620	7.8e-12
dolphin	840,	89,	203,	74,	15,	7,	1,	2,	0,	0,	0	1231	5018	2016	2.7e-11
gear	1007,	147,	160,	64,	37,	9,	8,	9,	4,	1,	0	1446	6000	6000	6.4e-12
hammer	502,	73,	108,	34,	10,	6,	3,	1,	0,	0,	0	737	3004	3000	8.6e-12
hand	768,	36,	302,	27,	7,	1,	10,	0,	0,	0,	0	1142	4610	4608	5.7e-12
head	1383,	184,	241,	157,	38,	10,	4,	0,	0,	1,	0	2018	8270	8268	3.7e-11
rabbit	904,	134,	154,	88,	24,	9,	7,	1,	0,	0,	0	1321	5414	5412	6.4e-12
skull	1252,	151,	255,	124,	39,	7,	1,	2,	1,	0,	0	1832	7474	7476	7.8e-12
venus	502,	65,	113,	48,	11,	3,	0,	0,	0,	0,	0	742	3002	3000	2.8e-12

Table 2.2. Detailed features of the meshes in Figures 2.8 to 2.12.



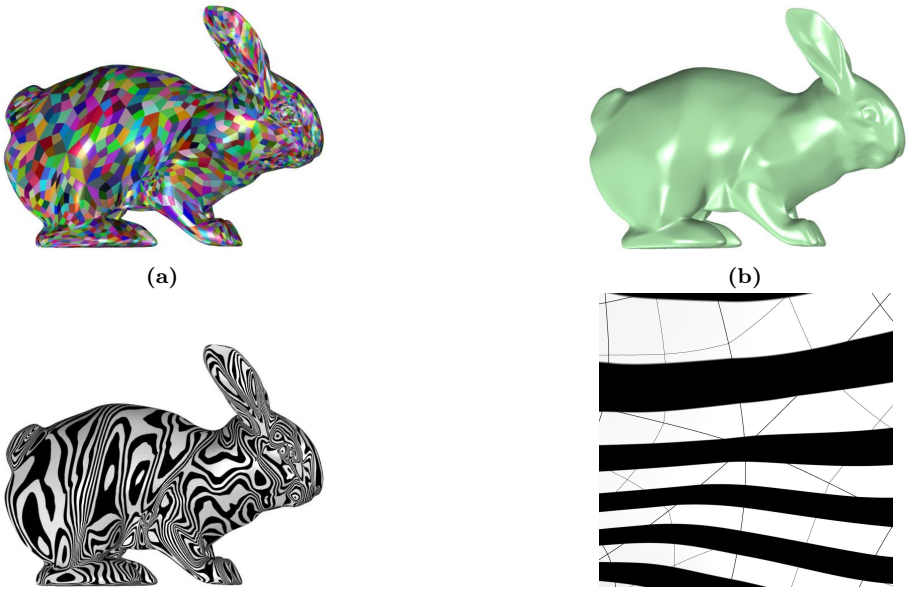
(a) **(b)** **(c)** **(d)**

Figure 2.8. Alien. (a): multipatch color. (b): solid color. (c): isophotes. (d): zoom on an EV of valence 7. The surface was obtained using the solving strategy CS-S.



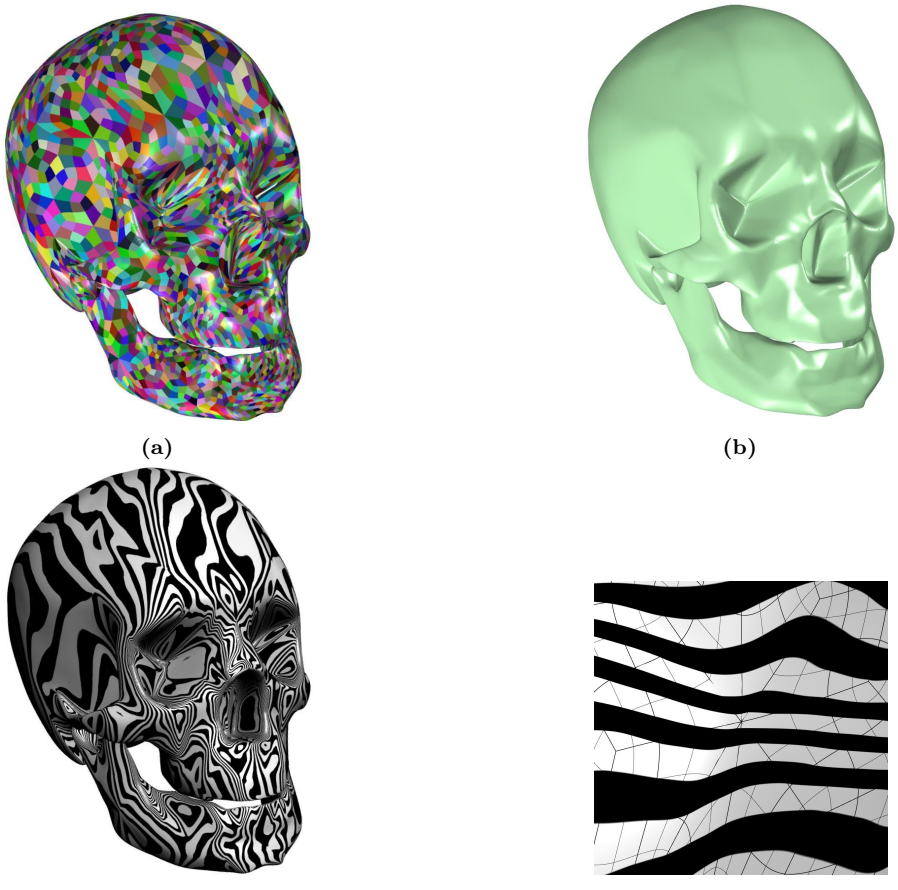
(a) **(b)** **(c)** **(d)**

Figure 2.9. Hand. (a): multipatch color. (b): solid color. (c): isophotes. (d): zoom on an EV of valence 3. The surface was obtained using the solving strategy CS-S.



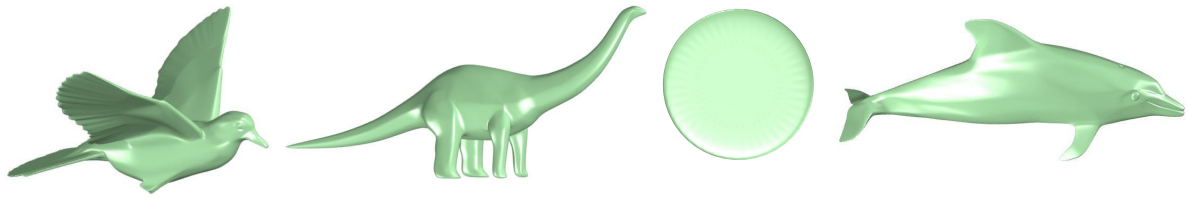
(a) **(b)** **(c)** **(d)**

Figure 2.10. Rabbit. (a): multipatch color. (b): solid color. (c): isophotes. (d): zoom on an EV of valence 6. The surface was obtained using the solving strategy CS-S.



(a) **(b)** **(c)** **(d)**

Figure 2.11. Skull. (a): multipatch color. (b): solid color. (c): isophotes. (d): zoom on an EV of valence 8. The surface was obtained using the solving strategy CS-S.



(a) bird **(b)** dinosaur **(c)** disk **(d)** dolphin

(e) gear **(f)** hammer **(g)** head **(h)** venus

Figure 2.12. Other surfaces obtained by complex meshes using the solving strategy CS-S.

2.4.3 Comparison with ACC_3 surface and Catmull-Clark limit surface

In order to investigate the accuracy of our G^1 approximation of the Catmull-Clark limit surface we analyse on the distance between the two surfaces; the two quantities we take into account are the geometry error and the normal error. For each point laying on the Catmull-Clark limit surface, to compute the geometry error we look for their closest points on the G^1 surface: formally, if P_{CC} is a point on the Catmull-Clark surface, the corresponding geometry error geom_{err} is obtaining by minimizing the distance

$$\text{dist}(P_{CC}, \mathcal{G}^1) = \min_{P \in \mathcal{G}^1} \|P_{CC} - P\|_2 \,,$$

where we define with $\mathcal{G}^1$ to be our CS-S surface. To evaluate the normal error, we first compute the orthogonal projection $P_\perp$ of each point P_{CC} onto $\mathcal{G}^1$. Then the normal error norm_{err} is obtained evaluating the 2-norm of the corresponding unit normals attached to them

$$\text{norm}_{err} = \|\mathbf{n}(P_{CC}) - \mathbf{n}(P_\perp)\|_2 \,.$$

The same study is conducted for the ACC_3 surface in order to provide a comparison with our improved construction and the obtained result are shown in Figure 2.13. We also estimate the convergence rates of geom_{err} and norm_{err} for both the surfaces ACC_3 and $\mathcal{G}^1$. Starting from different Catmull-Clark subdivision levels of the same input mesh, we construct on each of them the two surfaces and we evaluate the maximum of the two errors over a dense sampling of the surface.

In accordance with the quadratic convergence of the Catmull-Clark subdivision scheme, the geometric error geom_{err} of our G^1 construction converges quadratically, while norm_{err} converges with a rate of 0.7.

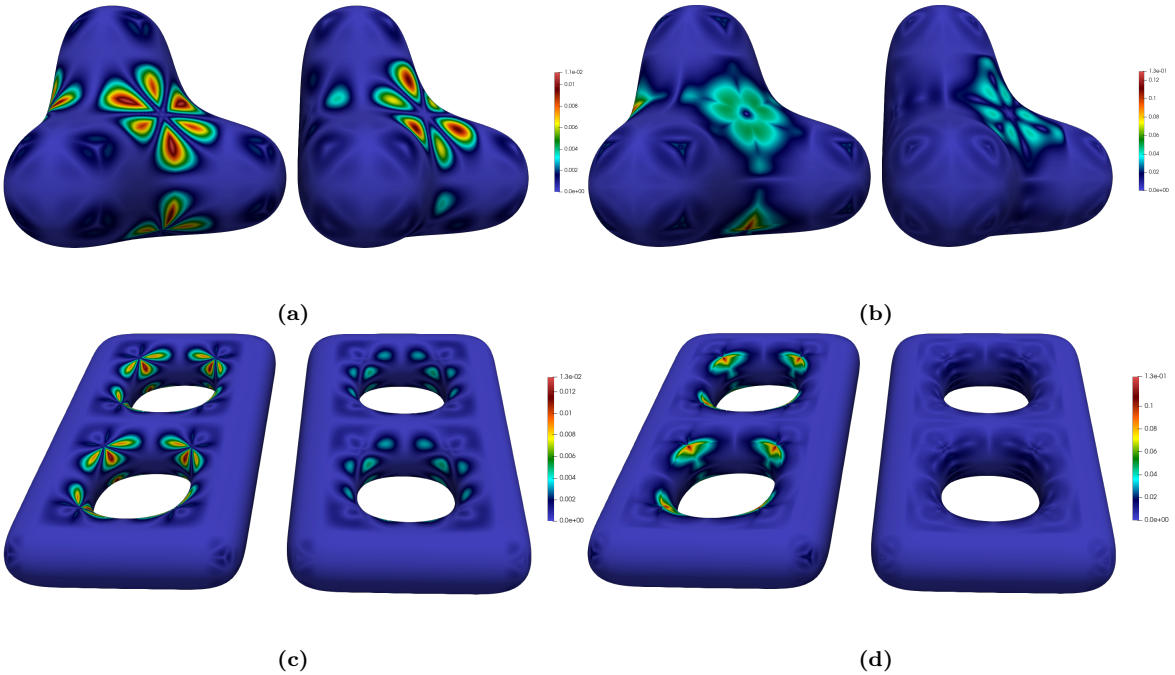


Figure 2.13. Catmull-Clark limit surfaces with error colormap representing the geometry error (a)-(c) and normal error (b)-(d) between the ACC_3 surface (left objects) and our G^1 surface (right objects) using the data in [LS08, p. 8].

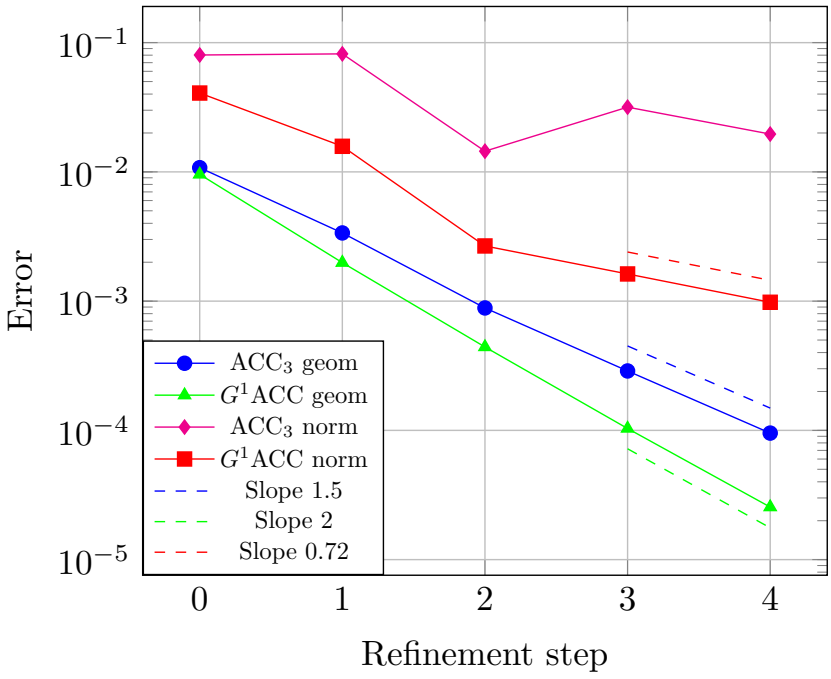


Figure 2.14. Experimental convergence rates of the geometry and normal errors for ACC_3 and G^1CS-S under successive Catmull-Clark subdivisions.

We also observed a convergence rate of 1.5 for the normal vector obtained as cross product between the coordinate derivatives, which is in general a non-unit vector. On the other side, in the ACC_3 construction we observe that geom_{err} converges with a rate of 1.5 while norm_{err} presents an oscillatory non-convergent behavior. Figure 2.14 summarizes these results.

Moreover, from the numerical results we also noticed that this construction returns unit normals at the vertices which are close to the real normal of the Catmull-Clark surface. Let $\mathrm{T}_{\mathbf{u}}, \mathrm{T}_{\mathbf{v}} = C\,\mathrm{T}_{\mathbf{u}}$ be the masks returning the tangents of the limit Catmull-Clark surface with weights described in [LS08, pag. 3, 6] and $\partial_{\mathbf{u}} = \mathrm{M}_{1,0} - \underline{\mathrm{M}}_{0,0}, \partial_{\mathbf{v}} = \mathrm{M}_{0,1} - \underline{\mathrm{M}}_{0,0} = C\,\partial_{\mathbf{u}}$ the masks returning the derivative directions of our surface obtained from (2.15). Denoting D the matrix with columns $\mathrm{T}_{\mathbf{u}}, \mathrm{T}_{\mathbf{v}}, \partial_{\mathbf{u}}, \partial_{\mathbf{v}}$, we observe that two trailing singular values $\sigma_i(D)$ of D are small, which means that the four vectors are almost coplanar i.e. the tangent planes generated by $\mathrm{Span}\{\mathrm{T}_{\mathbf{u}}, \mathrm{T}_{\mathbf{v}}\}$ and $\mathrm{Span}\{\partial_{\mathbf{u}}, \partial_{\mathbf{v}}\}$ are approaching each other, also verified by the converge of the unit normal (cf. Figure 2.14).

Summary

In this chapter we presented a novel construction of G^1 surfaces defined through masks over quad meshes which approximate the well-known Catmull-Clark limit surface obtained from the same initial geometry. Starting from the C^0 scheme in [LS08], by degree elevating the extraordinary patches and using of quadratic gluing data functions we achieved explicit solutions for the weights identifying the different masks, after having adequately fixed the degrees of freedom that occurred during the construction; these masks would return the control points forming the net of a biquintic Bézier patch. Several numerical tests on a benchmark of challenging meshes ensured the quality of the latter construction.

Chapter 3

Geometrically smooth functions for point cloud fitting

This chapter is devoted to the construction of a G^1 set of basis function generating the space of geometrically smooth splines defined over a quad mesh with the aim of applying them in point cloud fitting problem. Point clouds arise in many different fields, from earth surveying to medical imaging, and being able to have a smooth representation of it is a crucial task. Here, we investigate this topic by using G^1 Bézier functions defined over a quadrilateral mesh.

After having recalled the equations generating a G^1 smooth spline space we will go further in the construction of a set of basis function spanning the targeted space; moreover, we also provide a combinatorial formula for the space dimension depending only on the main features of the given mesh. The numerical experiments we provide to show the quality of the resulting fitting are mainly obtained by using parametrized point cloud, i.e. points already equipped with a parametrization. Moreover, we present a full pipeline to deal with general point clouds in order to compute a quad mesh and a parametrization. This chapter is based on [MMM24].

3.1 G^1 spline space on a mesh $\mathcal{M}$

Here we will recall the equation defining a G^1 joint between two adjacent Bézier patches nearby EVs on a quad mesh $\mathcal{M}$: similarly to relations (2.4) to (2.10), if $\mathrm{b}_{i,j}^{(k)}$ are the control points laying on the k-th patch (see Figure 2.2 for their labelling), the geometric continuity constraints across an edge linked to an EV of valence N are:

$$\begin{aligned}
\mathrm{b}_{0,1}^{(1)} + \mathrm{b}_{1,0}^{(0)} &= \bar{a}_0\mathrm{b}_{0,0}^{(1)} + a_0\mathrm{b}_{1,0}^{(1)}, && (3.1)\\
5(\mathrm{b}_{1,1}^{(1)} + \mathrm{b}_{1,1}^{(0)}) &= a_0\mathrm{b}_{0,0}^{(1)} + 5\bar{a}_0\mathrm{b}_{1,0}^{(1)} + 4a_0\mathrm{b}_{2,0}^{(1)}, && (3.2)\\
10(\mathrm{b}_{2,1}^{(1)} + \mathrm{b}_{1,2}^{(0)}) &= -a_0\mathrm{b}_{0,0}^{(1)} + 5a_0\mathrm{b}_{1,0}^{(1)} + 10\bar{a}_0\mathrm{b}_{2,0}^{(1)} + 6a_0\mathrm{b}_{3,0}^{(1)}, && (3.3)\\
10(\mathrm{b}_{3,1}^{(1)} + \mathrm{b}_{1,3}^{(0)}) &= a_0\mathrm{b}_{0,0}^{(1)} - 5a_0\mathrm{b}_{1,0}^{(1)} + 10a_0\mathrm{b}_{2,0}^{(1)} + 10\bar{a}_0\mathrm{b}_{3,0}^{(1)} + 4a_0\mathrm{b}_{4,0}^{(1)}, && (3.4)\\
\mathrm{b}_{4,1}^{(1)} + \mathrm{b}_{1,4}^{(0)} &= 2\mathrm{b}_{4,0}^{(1)}, && (3.5)\\
\mathrm{b}_{5,1}^{(1)} + \mathrm{b}_{1,5}^{(0)} &= 2\mathrm{b}_{5,0}^{(1)}, && (3.6)\\
10(\mathrm{b}_{3,0}^{(1)} - \mathrm{b}_{2,0}^{(1)}) &= \mathrm{b}_{0,0}^{(1)} - 5\mathrm{b}_{1,0}^{(1)} + 5\mathrm{b}_{4,0}^{(1)} - \mathrm{b}_{5,0}^{(1)}, && (3.7)
\end{aligned}$$

with $\bar{a}_0 = 2 - a_0$. The gluing data function we use is the same as in (2.2).

Starting from the above equations (3.1) to (3.7), we shall present an explicit construction of a set of basis functions generating the G^1 spline space that will follow the structure of

the input mesh $\mathcal{M}$. For this purpose, it is interesting to count its main features: we will refer to n_V as the number of vertices of the mesh, which can be subdivided in n_{IEV} inner EVs, n_{BEV} boundary EVs and n_{RV} regular vertices. The total number of edges is denoted by n_E and is composed by the number of extraordinary and inner regular edges, n_{EE} and n_{IRE} respectively, that is, edges attached to extraordinary and regular vertices, as well as the number of boundary edges n_{BE}. With n_F we refer to the number of faces and lastly n_C is the number of corners of the mesh.

3.2 Basis extraction

In this section we present an explicit construction of a set of basis functions generating the G^1 space over a quad mesh $\mathcal{M}$, which we refer to as $\mathcal{B}$. Following the topology of the input mesh $\mathcal{M}$, we can distinguish the set of functions attached to the vertices $\mathcal{B}_V$ (i.e. spanned by functions whose support lies on all the patch(es) sharing the vertex), set of basis functions attached to the edges $\mathcal{B}_E$ (i.e. spanned by functions whose support lies on the patch(es) sharing the edge) and set of basis functions attached to the faces $\mathcal{B}_F$ (i.e. spanned by functions whose support lies only on the interior of a single patch). This decomposition will be exploited in the proofs of Section 3.3. Thus, we arrive at a set of basis functions which can be decomposed as

$$\mathcal{B} = \left(\bigcup_{i=1}^{n_V} \mathcal{B}_{V_i}\right) \cup \left(\bigcup_{i=1}^{n_E} \mathcal{B}_{E_i}\right) \cup \left(\bigcup_{i=1}^{n_F} \mathcal{B}_{F_i}\right). \tag{3.8}$$

Also in this construction we assume that all the EVs in $\mathcal{M}$ are isolated, that is, their one-ring neighborhood contains only RVs. Having EVs attached to regular vertices only, which translates into (3.5) and (3.6), implies that vertex, edge and face functions have value and derivative equal to zero on the boundary of their support. This is because the G^1 constraints, defined making use of the gluing functions (2.2), do not affect the final six points across the edge (i.e. the points responsible of the value and derivative of the basis functions at the endpoint of the edge) and they can therefore be set to zero. Thus, basis functions with local support on a collection of faces of $\mathcal{M}$ do not influence functions in the region surrounding its support. By combining (3.1) to (3.7) circularly around all the edges attached to an EV we can reformulate the G^1 constraints using a block staircase matrix, which is useful to better understand the basis extraction and their analysis which follows. Let $\mathbf{b}_{i,j} = \left(\mathrm{b}_{i,j}^{(k)}\right), k = 1, \ldots, N$, be the vector containing all the points $\mathrm{b}_{i,j}^{(k)}$ attached to the neighborhood of the EV we are considering, $\mathbf{u} = (\underbrace{1, \ldots, 1}_{N})^T$, $\widehat{C} \in \mathbb{R}^{N \times N}$ and the circulant matrix $\widehat{C}$ introduced in (2.11)

$$\widehat{C} = \begin{pmatrix} 0 & 1 & 0 & \cdots & 0 \\ & & 1 & & \\ & & & \ddots & \\ & & & & 1 \\ 1 & 0 & & \cdots & 0 \end{pmatrix} = \mathrm{Circ}(0, 1, 0, \ldots, 0).$$

Note that $\mathbf{b}_{4,0} = \mathbf{b}_{5,0} = 0$, as a consequence of the locality of the G^1 constraints around the EV. The full system of G^1 relations around an EV can be written as

$$\begin{pmatrix} -\bar{a}_0\mathbf{u} & C_1 & & & & & & & \\ -a_0\mathbf{u} & -5\bar{a}_0 I & -4a_0 I & 5C_2 & & & & & \\ -\mathbf{u} & 5I & -10I & & 10I & & & & \\ a_0\mathbf{u} & -5a_0 I & -10\bar{a}_0 I & & -6a_0 I & 10I & 10I & & \\ -a_0\mathbf{u} & 5a_0 I & -10a_0 I & & 10\bar{a}_0 I & & & 10I & 10I \end{pmatrix} \begin{pmatrix} \mathbf{b}_{0,0} \\ \mathbf{b}_{1,0} \\ \mathbf{b}_{2,0} \\ \mathbf{b}_{1,1} \\ \mathbf{b}_{3,0} \\ \mathbf{b}_{2,1} \\ \widehat{\mathbf{b}}_{1,2} \\ \mathbf{b}_{3,1} \\ \widehat{\mathbf{b}}_{1,3} \end{pmatrix} = \mathbf{0}, \tag{3.9}$$

where $a_0 = 2\cos(2\pi/N)$ is the first Bernstein coefficient of the gluing data function (2.2), $\bar{a}_0 = 2 - a_0$, $I \in \mathbb{R}^{N\times N}$ is the identity matrix,

$$C_1 = -a_0 I + \widehat{C} + \widehat{C}^{N-1}, \quad C_2 = I + \widehat{C}, \tag{3.10}$$

and $\widehat{\mathbf{b}}_{1,2} = \widehat{C}\,\mathbf{b}_{1,2}$, $\widehat{\mathbf{b}}_{1,3} = \widehat{C}\,\mathbf{b}_{1,3}$, with $\widehat{C}$ in (2.11). The strategy we exploit to obtain the Bézier coefficients defining the basis functions is the following: starting from (3.1) to (3.7) or eq. (3.9) we impose, one by one, the value one to each "free" coefficient involved in the G^1 constraints. Then, with this initial value we solve the G^1 relations (3.1) to (3.7) (or (3.9)), while we gradually set the value of any unconstrained coefficients that we encounter (i.e. coefficients which lead to an overdetermined equation) to zero.

3.2.1 The set $\mathcal{B}_V$ of vertex basis functions

Here belong basis functions connected to inner and boundary vertices (extraordinary or regular) and corner vertices. We only provide the explicit construction for functions attached to inner extraordinary and inner regular vertices, since the construction for the remaining cases is analogous.

3.2.1.1 Construction of basis functions associated to an inner EV

Given an extraordinary vertex of valence N, we associate to it a basis function responsible of the value (of elements in the G^1 spline space) at the vertex and basis functions related to first and cross derivatives. We deduce during the construction that these basis functions are $N + 3$ in total.

Basis function related to the vertex value. To extract the basis function associated to the value at the vertex, we start solving the system (3.1) to (3.7) with initial value $\mathbf{b}_{0,0} = 1$ and continuing the construction fixing zero values for all the control points which are not constrained by any relation we will encounter during the construction. With this assumption, we can rewrite (3.1) in the form

$$C_1\mathbf{b}_{1,0} = (2 - a_0)\mathbf{u}, \tag{3.11}$$

with $C_1 \in \mathbb{R}^{N\times N}$ defined in (3.10). The solution of (3.11) returns the control points involved in the definition of the first derivative of the basis with unit value at the vertex. From Section 2.3.1, in particular from Proposition 2.3.1 and Theorem 2.3.2, we can deduce that the matrix

$C_1 = -a_0 I + \widehat{C} + \widehat{C}^{N-1}$ is singular and corank$(C_1) = 2$; for this reason, in order to obtain a unique solution we need to insert two extra constraints to the system (3.11). Let

$$\mathrm{Ker}(C_1) = K = \mathrm{Span}\{\mathbf{k}_1, \mathbf{k}_2\}$$

be the kernel of the matrix C_1 generated by the two vectors $\mathbf{k}_1$ and $\mathbf{k}_2$; we can choose our solution to be orthogonal to the space K, i.e.

$$\langle \mathbf{b}_{1,0} | \mathbf{k}_1 \rangle = \langle \mathbf{b}_{1,0} | \mathbf{k}_2 \rangle = 0.$$

To compute the kernel K we can use formula (2.14). This procedure allows us to achieve a unique solution for this set of control points. Going further in the resolution of the system, using the solution of (3.1) and the circulant matrix C, and taking into account the constraint along the edge in (3.7) which becomes

$$\mathbf{b}_{2,0} = \frac{1}{2}\mathbf{b}_{1,0} - \frac{1}{10}\mathbf{u}, \tag{3.12}$$

we can reorder (3.2) as

$$C_2 \mathbf{b}_{1,1} = -\frac{1}{5} a_0 \mathbf{u} + 5(2 + a_0)\mathbf{b}_{1,0}, \tag{3.13}$$

with C_2 as in (3.10). Following the analysis in previous Section 2.3.2.1, for odd values of the valence N the matrix C_2 is invertible, while for even occurrences we have corank$(C_2) = 1$. To obtain a unique solution in the singular case we need to fix an extra constraint which we choose to be the orthogonality of the expected solution $\mathbf{b}_{1,1}$ to $\mathrm{Ker}(C_2)$. Regarding the control points for the higher derivatives $\mathbf{b}_{2,1}$ and $\mathbf{b}_{3,1}$, from (3.3) and (3.4) and using again (3.12) we come up with the relations

$$\mathbf{b}_{2,1} + \widehat{\mathbf{b}}_{1,2} = -\frac{1}{5}\mathbf{u} + \mathbf{b}_{1,0} \quad \text{and} \quad \mathbf{b}_{3,1} + \widehat{\mathbf{b}}_{1,3} = \mathbf{0}, \tag{3.14}$$

which can be solved, for example (and as we do), imposing the extra relations

$$\mathbf{b}_{2,1} = \widehat{\mathbf{b}}_{1,2},$$
$$\mathbf{b}_{3,1} = \widehat{\mathbf{b}}_{1,3}.$$

This procedure, as depicted from the construction, returns a unique basis function. In Figure 3.1-(a) is presented an example of the coefficients obtained with the previous construction in case of an EV of valence $N = 3$, while in Figure 3.2-(a) is shown the plot of the basis function for $N = 5$.

Basis functions attached to the first derivatives at the vertex. Proceeding with the construction of the second subset of basis functions we start again from (3.1) but imposing this time the value $\mathbf{b}_{0,0} = 0$; this choice leads to the following homogeneous linear system

$$C_1 \mathbf{b}_{1,0} = \mathbf{0}, \tag{3.15}$$

where the matrix C_1 is the same as the previous paragraph and introduced in (3.10). Similarly to (3.11), a solution of (3.15) is easily given by

$$\mathbf{b}_{1,0} \in \mathrm{Ker}(C_1) = \mathrm{Span}\{\mathbf{k}_1, \mathbf{k}_2\};$$

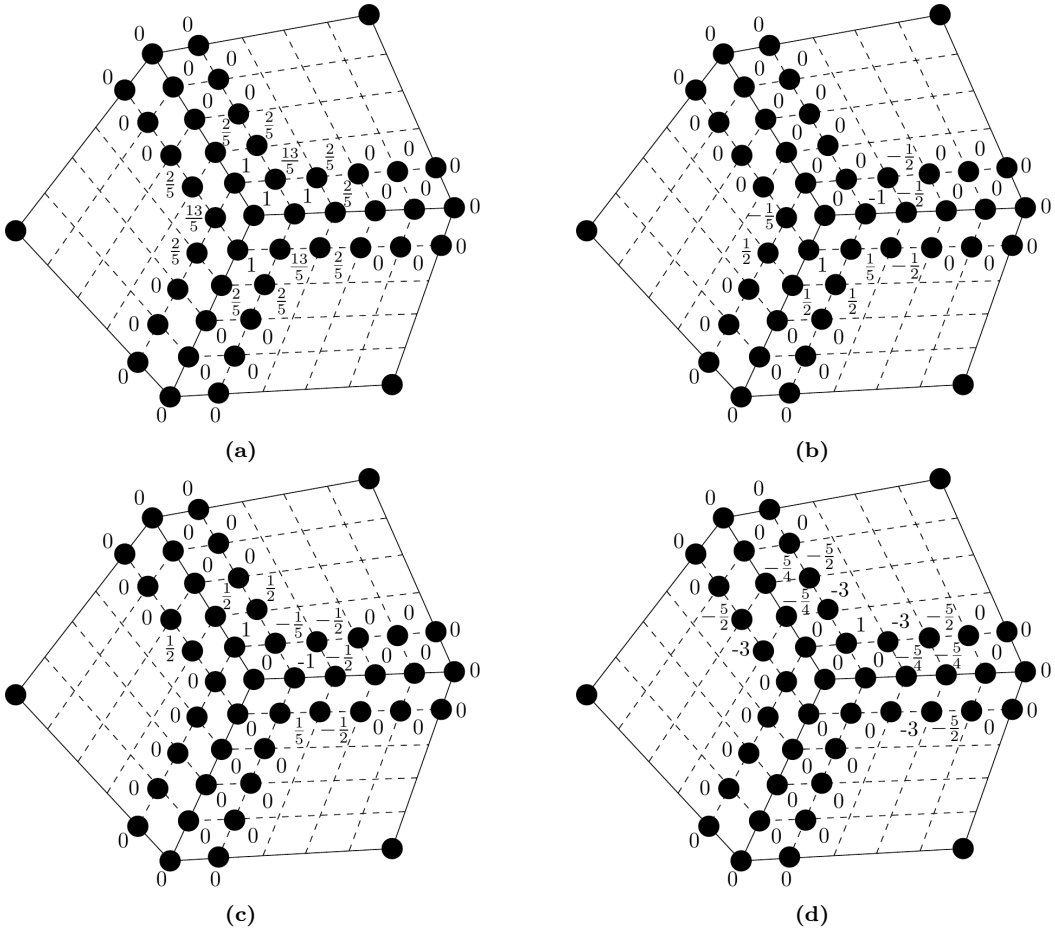


Figure 3.1. Coefficients for a basis function attached to the value at the vertex (a), attached to the first derivatives at the vertex (b)-(c) and attached to the cross derivatives at the vertex (d) for an EV of valence $N = 3$.

since the kernel of the matrix C_1 is a 2-dimensional space generated by the vectors $\mathbf{k}_1$ and $\mathbf{k}_2$ we will have two admissible solutions for the system (3.15) which, in fact, leads to two different basis function attached to value of the first derivative at the vertex obtained by solving the other G^1 relations starting with $\mathbf{b}_{1,0} = \mathbf{k}_1$ and $\mathbf{b}_{1,0} = \mathbf{k}_2$. The remaining constraints relating high order derivatives, taking into account the edge constraint obtained from (3.7)

$$\mathbf{b}_{2,0} = \frac{1}{2}\mathbf{b}_{1,0},$$

are given by the equations

$$\begin{aligned} C_2\mathbf{b}_{1,1} &= \frac{1}{5}(2 + a_0)\mathbf{b}_{1,0}, \\ \mathbf{b}_{2,1} + \widehat{\mathbf{b}}_{1,2} &= \mathbf{b}_{1,0}, \\ \mathbf{b}_{3,1} + \widehat{\mathbf{b}}_{1,3} &= \mathbf{0}, \end{aligned}$$

which can be solved in the same way as (3.13) and (3.14). Figure 3.1-(b) and (c) show the graph of the resulting functions for an EV with $N = 3$ while Figure 3.2-(b) and (c) presents their plot when in presence of an EV with valence $N = 5$.

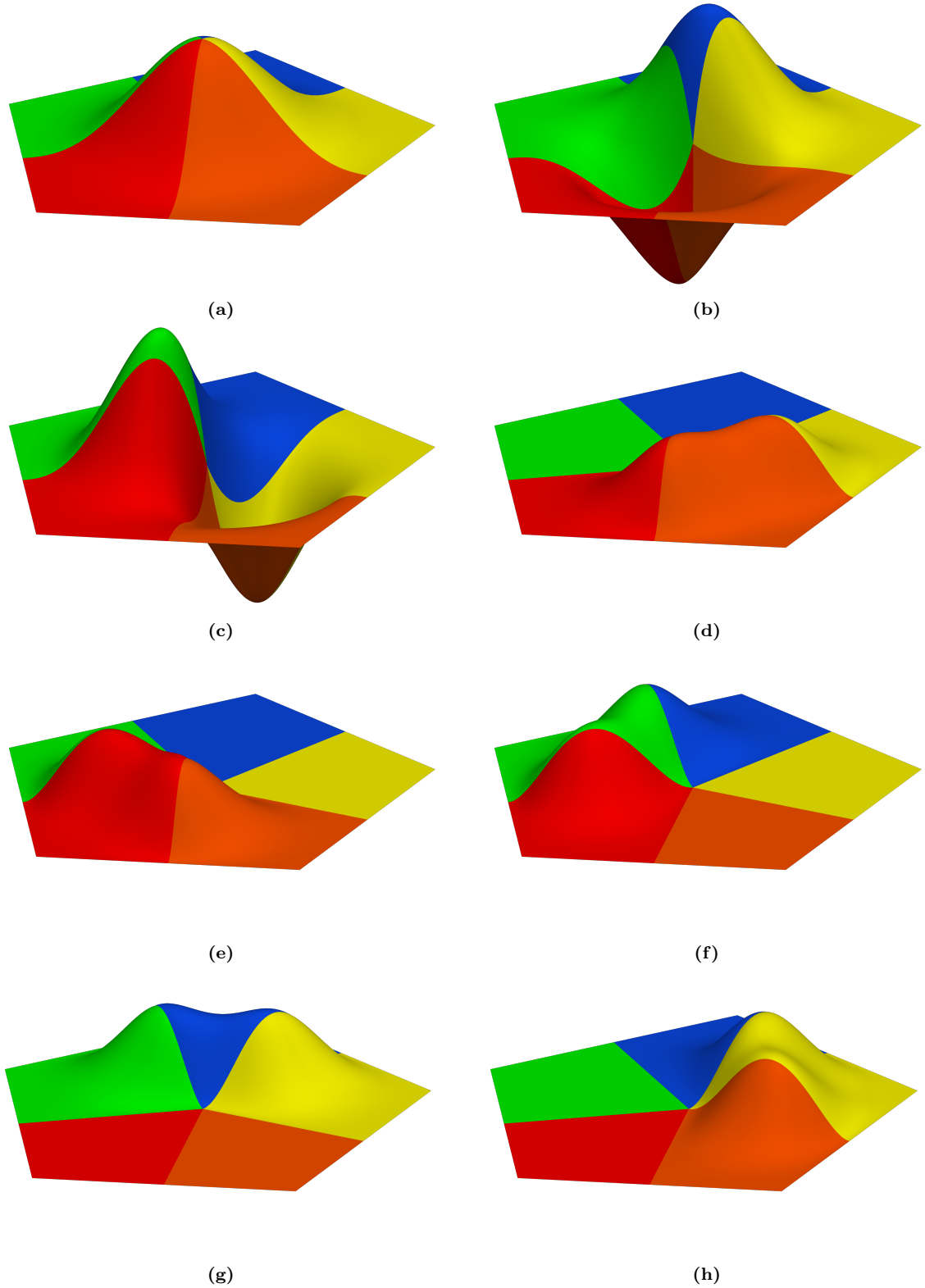


Figure 3.2. Basis function attached to the value of the vertex (a), value of first derivatives (b)-(c) and value of cross derivatives (d)-(e)-(f)-(g)-(h) for an EV of valence $N = 5$.

Basis responsible for the cross derivatives at the vertex. To a vertex of valence N correspond N cross derivatives attached to it; this means that we need to compute N basis functions related to the value of the cross derivative at the vertex. Let's consider the k-th patch belonging to the vertex ring. By setting the value $\mathrm{b}_{1,1}^{(k)} = 1$, from (2.5) we realize that this point has only influence on values laying in patches $k-1$, k and $k+1$; more precisely, the points affected by this choice are only $\mathrm{b}_{2,0}^{(k)}$ and $\mathrm{b}_{2,0}^{(k+1)}$ regarding the second derivatives, which will have values equal to

$$\mathrm{b}_{2,0}^{(k)} = \mathrm{b}_{2,0}^{(k+1)} = \frac{5}{4a_0},$$

being well defined since $a_0 \neq 0 \iff N \neq 4$, that is the case of an EV we are in fact investigating. Regarding higher order derivatives, we have the points $\mathrm{b}_{3,0}^{(k)}, \mathrm{b}_{3,0}^{(k+1)}, \mathrm{b}_{1,2}^{(k-1)}, \mathrm{b}_{2,1}^{(k)}, \mathrm{b}_{1,2}^{(k)}, \mathrm{b}_{2,1}^{(k+1)}$ and $\mathrm{b}_{1,3}^{(k-1)}, \mathrm{b}_{3,1}^{(k)}, \mathrm{b}_{1,3}^{(k)}, \mathrm{b}_{3,1}^{(k+1)}$ defined by the relations

$$\begin{aligned}
\mathrm{b}_{3,0}^{(k)} = \mathrm{b}_{3,0}^{(k+1)} &= \frac{5}{4a_0}, \\
\mathrm{b}_{1,2}^{(k-1)} + \mathrm{b}_{2,1}^{(k)} = \mathrm{b}_{2,1}^{(k)} + \mathrm{b}_{1,2}^{(k+1)} &= \frac{1}{2}\left(\frac{5}{a_0} - 1\right), \\
\mathrm{b}_{1,3}^{(k-1)} + \mathrm{b}_{3,1}^{(k)} = \mathrm{b}_{3,1}^{(k)} + \mathrm{b}_{1,3}^{(k+1)} &= \frac{5}{2a_0},
\end{aligned}$$

which are obtained making use of (3.3), (3.4) and (3.7). Previous equations can be solved as in (3.14). Repeating the same construction for all the patches in the ring we come up with the N basis functions attached to the cross derivatives. The result in the case $N = 3$ is shown in Figure 3.1-(d), while Figure 3.2-(d) to (h) presents the plot of the set of cross derivative basis functions for an EV of valence $N = 5$.

3.2.1.2 Basis functions at an inner regular vertex

In presence of a regular vertex (RV), i.e. a vertex with valence $N = 4$, we expect to have by construction C^1-smooth basis functions; this is in fact what (3.5) and (3.6) state. First we need to investigate how many RV basis function we have. To do that we first need to expand cyclically (3.5) and (3.6) to all the control points around the vertex; this procedure, using the notation in Figure 3.3-(a) leads to the following system:

$$\begin{pmatrix}
1 & 0 & 0 & -2 & 0 & 0 & 1 & 0 & 0 \\
0 & 1 & 0 & 0 & -2 & 0 & 0 & 1 & 0 \\
1 & -2 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 & -2 & 0 & 0 & 1 & 0 \\
0 & 0 & 1 & 0 & 0 & -2 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & -2 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & -2 & 1
\end{pmatrix}
\begin{pmatrix}
\mathrm{P} \\ \mathrm{Q} \\ \mathrm{R} \\ \mathrm{S} \\ \mathrm{T} \\ \mathrm{U} \\ \mathrm{V} \\ \mathrm{W} \\ \mathrm{X}
\end{pmatrix} = \mathbf{0}. \tag{3.16}$$

Let D be the coefficient matrix in (3.16). The number of basis functions attached to the RV is given by $\operatorname{corank}(D) = 9 - \operatorname{rank}(D) = 9 - 5 = 4$. Hence, a choice for the Bézier points returning linearly independent functions (by construction) verifying (3.16) is given by the

coefficients in Figure 3.3-(b). In presence of RV connected to an EV it is also necessary to modify more control points for each extraordinary edge, according with (3.3) and (3.4).

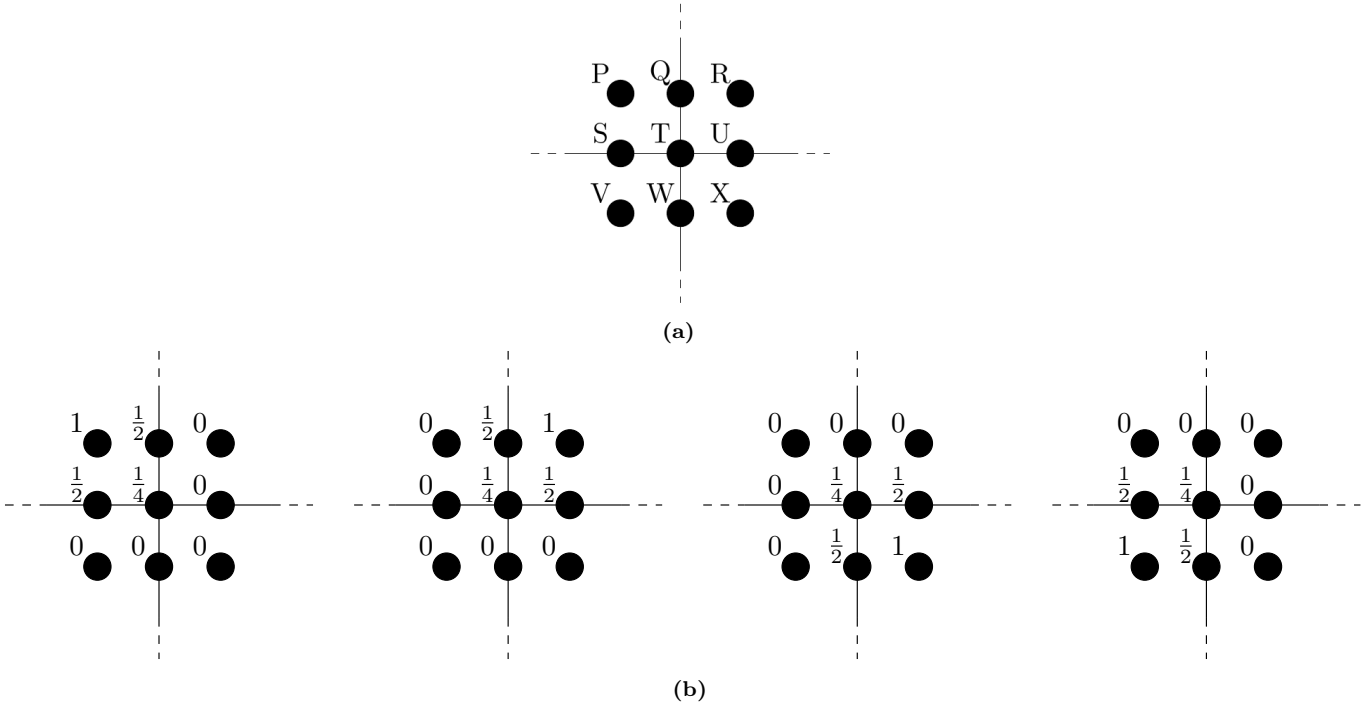


Figure 3.3. (a): labelling of the control points around an inner regular vertex. (b): values for the Bézier points of the four basis functions.

3.2.1.3 Basis functions linked to extraordinary and regular boundary vertices and corners

The extraction of this type of functions is analogous to the constructions developed in the previous sections for the inner cases. Naming with κ the valence of an extraordinary boundary vertex, i.e. the number of patches attached to it, imitating the process in Section 3.2.1.1 we come up with $\kappa + 3$ basis functions, which is equivalent to $N + 2$ since $\kappa = N - 1$. On the other hand, to obtain the functions connected to a regular boundary vertex and corner we need to copy the procedure shown in Section 3.2.1.2: in both cases, following the same strategy leading to (3.16), we come up again with four basis functions.

3.2.2 The set $\mathcal{B}_E$ of edge basis functions

In this second set of functions, we have basis functions attached to inner and boundary edges, either extraordinary and regular ones. In the same way as the vertex functions we will present the explicit construction in the case of inner extraordinary and regular edge functions, whereas the construction for the remaining case is analogous.

3.2.2.1 Construction of basis functions connected to an extraordinary edge

These functions are obtained starting from (3.3),(3.4) and (3.7). Similarly to the construction in Section 3.2.1.1, to extract the Bézier coefficients for the functions, we need to set zero values at the free points appearing in the equations. In the construction of basis functions connected to extraordinary edges, the control points we need to nullify are all the points laying on the edge, i.e. $\mathbf{b}_{0,0}$, $\mathbf{b}_{1,0}$, $\mathbf{b}_{2,0}$, $\mathbf{b}_{3,0}$, $\mathbf{b}_{4,0}$ and $\mathbf{b}_{5,0}$. This assumption transforms (3.3) and (3.4) into

$$\begin{aligned} \mathbf{b}_{2,1} + \widehat{\mathbf{b}}_{1,2} &= \mathbf{0}, \\ \mathbf{b}_{3,1} + \widehat{\mathbf{b}}_{1,3} &= \mathbf{0}, \end{aligned} \tag{3.17}$$

which define the two basis functions living on an extraordinary edge. The simplest solution satisfying (3.17) is to take

$$\begin{aligned} \mathbf{b}_{2,1} = \mathbf{b}_{3,1} &= 1, \\ \widehat{\mathbf{b}}_{1,2} = \widehat{\mathbf{b}}_{1,3} &= -1, \end{aligned}$$

or vice-versa. Figure 3.4 presents their plot.

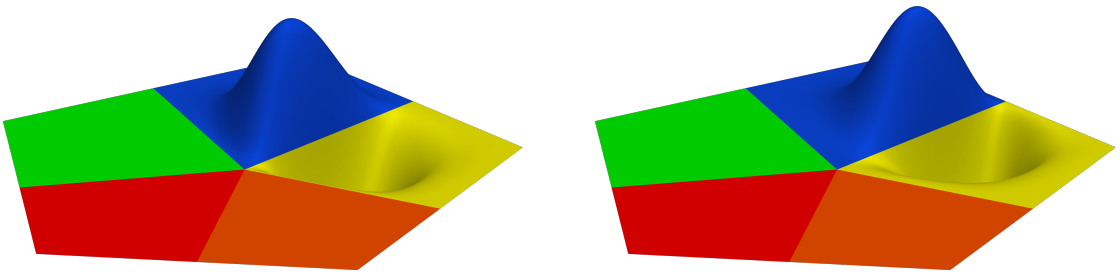


Figure 3.4. The two edge basis functions defined on an extraordinary edge.

3.2.2.2 Basis functions belonging to an inner regular edge

These basis functions are obtained with a similar approach as in Section 3.2.1.2 for an inner regular vertex; are involved in this construction the six control points in the inner part of the edge, i.e. away from the influence of the vertices' equations. In order to determine the number of these functions, applying the C^1 constraints in (3.5) and (3.6) to the two layers of control points implicated in this analysis with the notation given by Figure 3.5-(a) we have:

$$\begin{pmatrix} 1 & -2 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & -2 & 1 \end{pmatrix} \begin{pmatrix} \bar{\mathrm{P}} \\ \bar{\mathrm{Q}} \\ \bar{\mathrm{R}} \\ \bar{\mathrm{S}} \\ \bar{\mathrm{T}} \\ \bar{\mathrm{U}} \end{pmatrix} = \mathbf{0}. \tag{3.18}$$

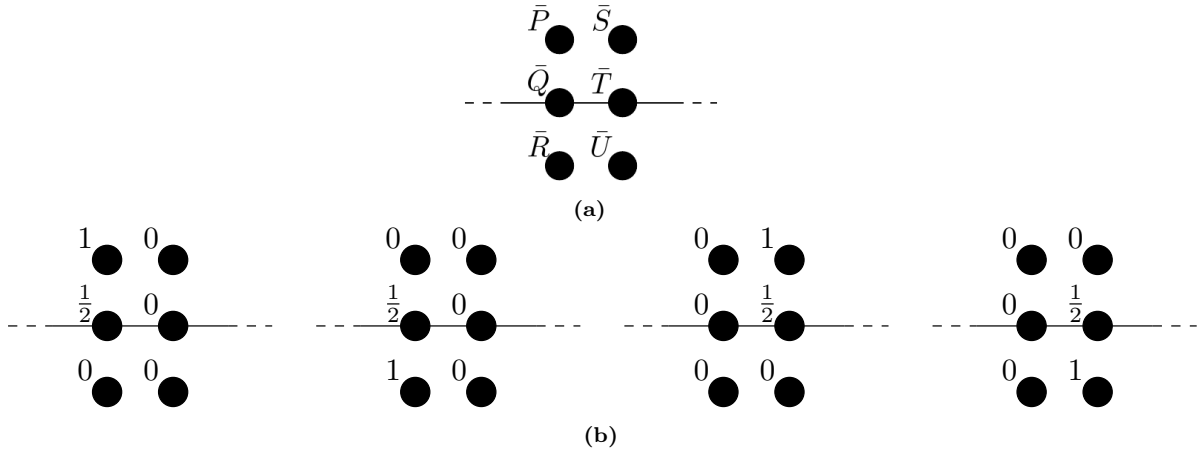


Figure 3.5. (a): labelling of the control points across a regular inner edge. (b): values for the Bézier points of the four basis functions.

If $\bar{D}$ is the matrix in (3.18), the number of basis functions attached to an inner regular edge is given by $\text{corank}(\bar{D}) = 4$ and a set of possible solutions verifying these constraints returning linearly independent functions is given by the configurations in Figure 3.5-(b).

3.2.2.3 Boundary edge basis functions

In presence of a boundary edge we have no smoothness constraints to impose. Hence, the basis functions in this case are the classical bivariate Bézier polynomials obtained assigning the unit value to the four control points involved in this setting, one at the time, to obtain the four basis functions we were looking for.

3.2.3 The set $\mathcal{B}_F$ of face basis functions

To conclude the construction of our basis we miss the definition of functions belonging uniquely to a single patch. As for the case of boundary edge functions, here we have no regularity conditions to impose; thus the construction is the same as in Section. 3.2.2.3 returning four Bézier polynomial on each face.

3.3 Analysis of the basis and space dimension

The functions we built in Section 3.2 form a basis of the G^1 spline space over the mesh $\mathcal{M}$. Denoting by $\mathbb{G}^1(\mathcal{M})$ such space, it results that $\mathbb{G}^1(\mathcal{M}) = \langle \mathcal{B} \rangle$. In this section we will provide a proof for this statement as well as a dimension formula for the underlined space. The first part of the proof, obtained by contradiction, focuses on the linear independence of the basis functions and it is carried out exploiting the property (3.8) and analysing their supports; the second half, instead, uses the equations defining the smoothness constraints of the basis functions in order to prove that they generate the desired spline space.

Theorem 3.3.1. *The functions introduced in Section 3.2 form a basis $\mathcal{B}$ for the space $\mathbb{G}^1(\mathcal{M})$ over a quad mesh $\mathcal{M}$.*

Proof. In order to prove that the set $\mathcal{B}$ is a basis of our space we need to prove that it is linear independent and the property of being a generating set. For the linear independence, we proceed by contradiction. Since we assume to have isolated EVs only, we have by construction that the supports of vertex basis functions attached to different vertices are disjoint; the same property holds for edge basis functions defined over different edges as well as face basis functions attached to different faces. Hence, we can restrict our proof on basis functions attached on elements sharing a common vertex. Therefore we consider an EV of valence N with its N extraordinary edges and N faces (the case with regular vertices and regular edges is analogous) and we assume that there exists a linear dependence between the basis functions supported on these elements, i.e. there exist non zero coefficients $\boldsymbol{\alpha}, \boldsymbol{\beta}, \boldsymbol{\gamma}$ such that

$$\sum_{i=1}^{N+3} \alpha_i B_i^{EV} + \sum_{j=1}^{N} \sum_{i=1}^{2} \beta_i^{EE_j} B_i^{EE_j} + \sum_{j=1}^{N} \sum_{i=1}^{4} \gamma_i^{F_j} B_i^{F_j} = 0. \tag{3.19}$$

From (3.19) it holds that

$$\operatorname{supp}\left(\sum_{i=1}^{N+3} \alpha_i B_i^{EV} + \sum_{j=1}^{N} \sum_{i=1}^{2} \beta_i^{EE_j} B_i^{EE_j}\right) = \operatorname{supp}\left(-\sum_{j=1}^{N} \sum_{i=1}^{4} \gamma_i^{F_j} B_i F_j\right),$$

and in particular

$$\operatorname{supp}\left(\sum_{i=1}^{N+3} \alpha_i B_i^{EV} + \sum_{j=1}^{N} \sum_{i=1}^{2} \beta_i^{EE_j} B_i^{EE_j}\right) \subseteq \left(\bigcup_{i=1}^{N+3} \operatorname{supp}\left(B_i^{EV}\right)\right) \cup \left(\bigcup_{i=1}^{N}\left(\bigcup_{j=1}^{2} \operatorname{supp}\left(B_i^{EE_j}\right)\right)\right)$$

$$\operatorname{supp}\left(\sum_{i=1}^{N+3} \alpha_i B_i^{EV} + \sum_{j=1}^{N} \sum_{i=1}^{2} \beta_i^{EE_j} B_i^{EE_j}\right) \subseteq \left(\bigcup_{i=1}^{N}\left(\bigcup_{j=1}^{4} \operatorname{supp}\left(B_i^{F_j}\right)\right)\right).$$

It follows that

$$\begin{aligned}
&\operatorname{supp}\left(\sum_{i=1}^{N+3} \alpha_i B_i^{EV} + \sum_{j=1}^{N} \sum_{i=1}^{2} \beta_i^{EE_j} B_i^{EE_j}\right) \\
&\subseteq \left(\left(\bigcup_{i=1}^{N+3} \operatorname{supp}\left(B_i^{EV}\right)\right) \cup \left(\bigcup_{i=1}^{N}\left(\bigcup_{j=1}^{2} \operatorname{supp}\left(B_i^{EE_j}\right)\right)\right)\right) \cap \left(\bigcup_{i=1}^{N}\left(\bigcup_{j=1}^{4} \operatorname{supp}\left(B_i^{F_j}\right)\right)\right) = \emptyset.
\end{aligned} \tag{3.20}$$

Hence, (3.20) implies

$$\sum_{i=1}^{N+3} \alpha_i B_i^{EV} + \sum_{j=1}^{N} \sum_{i=1}^{2} \beta_i^{EE_j} B_i^{EE_j} = 0 = \sum_{j=1}^{N} \sum_{i=1}^{4} \gamma_i^{F_j} B_i^{F_j},$$

that is $\gamma_i^{F_j} = 0 \ \forall\, i, j$ since $B_i^{F_j}$ are linearly independent by construction. Now we have

$$\sum_{i=1}^{N+3} \alpha_i B_i^{EV} = -\sum_{j=1}^{N} \sum_{i=1}^{2} \beta_i^{EE_j} B_i^{EE_j}.$$

Let us fix two adjacent patches k and $k-1$ sharing the EV we are considering and let us focus on the control points $\mathrm{b}_{0,0}^{(k)}, \mathrm{b}_{1,0}^{(k)}, \mathrm{b}_{0,1}^{(k)}, \mathrm{b}_{1,1}^{(k)}, \mathrm{b}_{1,0}^{(k-1)}, \mathrm{b}_{1,1}^{(k-1)}$. By definition, these control points are always equal to zero for all the basis functions $B_i^{EE_j}$, while this is not the case for the B_i^{EV}; hence, if we now look at the submatrices corresponding to the points we are considering for both B_i^{EV} and $B_i^{EE_j}$, we deduce that in the first case the selected submatrix is invertible, while the second submatrix is the null matrix. Therefore it follows that $\forall i$, $\alpha_i = 0$ since the B_i^{EV} are linearly independent by construction and consequently, using the same argument, $\beta_i^{EE_j} = 0 \ \forall i, j$.

Regarding the generating property, let $f \in \mathbb{G}^1(\mathcal{M})$. Since the basis functions $B_i^{F_j}$ attached to a face F_j are C^1 smooth we can define the function

$$f' := f - \sum_{j=1}^{n_F} \sum_{i=1}^{4} c_i^{F_j} B_i^{F_j}, \tag{3.21}$$

which is still G^1 and is such that the inner face coefficients vanish. The same procedure can be applied to the C^1 basis function attached to corners $B_i^{C_j}$, boundary edges $B_i^{BE_j}$ and inner regular edges $B_i^{IRE_j}$ which can be used to define a new function starting from (3.21) as

$$f'' := f' - \sum_{j=1}^{n_C} \sum_{i=1}^{4} c_i^{C_j} B_i^{C_j} - \sum_{j=1}^{n_{BE}} \sum_{i=1}^{4} c_i^{BE_j} B_i^{BE_j} - \sum_{j=1}^{n_{IRE}} \sum_{i=1}^{4} c_i^{IRE_j} B_i^{IRE_j} \in G^1,$$

so that it has null coefficients in the above mentioned sectors.

Let $\mathbf{s}$ be the indices of the 9 control points in Figure 3.3 related to the basis functions $B_i^{RV_j}$ attached to a regular vertex RV_j and let denote by $f[\mathbf{s}]$ the corresponding control coefficients of the function f. Since from (3.16)

$$Af[s] = 0,$$

it follows that

$$\mathrm{Ker}(A) = \mathrm{Span}\left\{ B_i^{RV_j}[\mathbf{s}] \right\}.$$

Hence

$$f''[\mathbf{s}] = \sum_{i=1}^{4} c_i^{RV_j} B_i^{RV_j}[\mathbf{s}];$$

we proceed similarly with boundary EVs $B_i^{BEV_j}$ and boundary regular vertices $B_i^{BRV_j}$ to define respectively coefficients $c_i^{BEV_j}, c_i^{BRV_j}$ so that the function

$$f''' := f'' - \sum_{j=1}^{n_{RV}} \sum_{i=1}^{4} c_i^{RV_j} B_i^{RV_j} - \sum_{j=1}^{n_{BRV}} \sum_{i=1}^{4} c_i^{BRV_j} B_i^{BRV_j} - \sum_{j=1}^{n_{BEV}} \sum_{i=1}^{N_{BEV_j}+2} c_i^{BEV_j} B_i^{BEV_j}$$

is still a G^1 smooth function and it has zero coefficients in the previously treated regions.

Now, the function f''' has only non-zero coefficients around extraordinary vertices and their incident edges. We define $\mathbf{t}$ as the indices of the non-zero coefficients for an extraordinary vertex EV_j (black points in Figure 3.1). Similarly to the case of regular vertices, from (3.9),

$$\bar{A} f'''[\mathbf{t}] = 0,$$

where $\bar{A}$ is the matrix in (3.9) containing the G^1 constraints. Then,

$$\mathrm{Ker}(\bar{A}) = \mathrm{Span}\left\{B_i^{EV_j}[\mathbf{t}]\right\}$$

which leads to

$$f'''[\mathbf{t}] = \sum_{i=1}^{3N_{EV_j}+3} c_i^{EV_j} B_i^{EV_j}[\mathbf{t}].$$

Finally, we have that

$$f'''' := f''' - \sum_{j=1}^{n_{EV}} \sum_{i=1}^{3N_{EV_j}+3} c_i^{EV_j} B_i^{EV_j} = 0,$$

which concludes the proof. □

As consequence of the particular structure of the set $\mathcal{B}$ we have the following formula for the dimension of our spline space.

Corollary 3.3.2. *The space $\mathbb{G}^1(\mathcal{M})$ has dimension given by:*

$$\begin{aligned}\dim\left(\mathbb{G}^1(\mathcal{M})\right) &= \sum_{i=1}^{n_V} |\mathcal{B}_{V_i}| + \sum_{i=1}^{n_E} |\mathcal{B}_{E_i}| + \sum_{i=1}^{n_F} |\mathcal{B}_{F_i}| \\ &= \sum_{i=1}^{n_{EV}} N_{EV_i} + 3n_{IEV} + 2(n_{BEV} + n_{EE}) + 4(n_{IRE} + n_{BE} + n_{RV} + n_C + n_F).\end{aligned}$$

Proof. The proof is obtained using the decomposition in (3.8) and summing up all the members of the basis functions for each feature of the mesh shown in Sections 3.2.1, 3.2.2 and 3.2.3. □

3.4 Numerical experiments

We conclude this chapter presenting numerical experiments in which we assess the basis functions constructed in Section 3.2 in point cloud fitting problems. The setup of our investigation is the classical least squares fitting problem: given a point cloud $\mathcal{P}$, i.e. a set of points $\boldsymbol{P}_i \in \mathbb{R}^3$, $i = 1, \dots, n_P$, with associated parameters $(u_i, v_i) \in [0,1]^2$ on the patch ℓ_i, we want to find the coefficients $\boldsymbol{c}_i \in \mathbb{R}^3$ of a G^1 surface $\boldsymbol{G} = \sum_k \boldsymbol{c}_k B_k$ such that the quantity

$$F = \sum_{i=1}^{n_P} \left\| \boldsymbol{G}^{(\ell_i)}(u_i, v_i) - \boldsymbol{P}_i \right\|_2^2 + \lambda E_{thin}, \quad \lambda \geq 0, \tag{3.22}$$

is minimal, where $\boldsymbol{G}^{(\ell)}$, $\ell = 1, \dots, n_F$, is the geometry map

$$\boldsymbol{G}^{(\ell)} : [0,1]^2 \longrightarrow \mathbb{R}^3, \quad (u_i^{(\ell)}, v_i^{(\ell)}) \longmapsto \boldsymbol{G}^{(\ell)}(u_i^{(\ell)}, v_i^{(\ell)}) = \sum_{i,j=1}^{36} \mathrm{b}_{i,j}^{(\ell)} B_{i,j}^{(\ell)}(u_i^{(\ell)}, v_i^{(\ell)}), \tag{3.23}$$

defining the $\mathbb{G}^1(\mathcal{M})$ space, and $\mathrm{b}_{i,j}^{(\ell)}$ the control points associated with the bivariate Bernstein polynomials $B_{i,j}^{(\ell)}$ defining the ℓ-th patch $\Omega^{(\ell)} = \boldsymbol{G}^{(\ell)}([0,1]^2)$.

We also take into account in our minimization problem (3.22) an energy term given by the standard thin-plate energy

$$E_{thin} = \sum_{\ell=1}^{n_F} \iint_{[0,1]^2} \left\| \boldsymbol{G}_{uu}^{(\ell)} \right\|_2^2 + 2 \left\| \boldsymbol{G}_{uv}^{(\ell)} \right\|_2^2 + \left\| \boldsymbol{G}_{vv}^{(\ell)} \right\|_2^2 \, du\, dv, \tag{3.24}$$

which is controlled by the parameter λ. The minimization of the functional in (3.24) is responsible for a regularisation effect on the final surface $\boldsymbol{G}$ (i.e. oscillations). Moreover, following the construction presented in [Hos88], we perform few iterations of parameter correction to further reduce the approximation error, if needed.

Since our focus is on the quality of the G^1 basis for fitting, the parametrized data that we use in the following numerical experiments are obtained by evaluating certain input functions or surfaces to obtain parameters. In particular, starting from a planar mesh, in order to build point cloud data from a given analytic function f, we compute parameters (u_i, v_i) by randomly sampling a certain number of points in the unit square $[0,1]^2$, for each patch. Therefore, the point cloud is obtained by the triple

$$(x_i, y_i, f(x_i, y_i)), \quad \text{with} \quad (x_i, y_i) = \boldsymbol{G}^{(\ell_i)}(u_i, v_i).$$

A similar procedure is performed to obtain a point cloud from a quad mesh yielding an AC_3 surface [LS08]: considering the geometry map $\boldsymbol{G}_{ACC_3}$, we obtain sample points

$$(x_i, y_i, z_i) = \boldsymbol{G}_{ACC_3}^{(\ell_i)}(u_i, v_i)$$

randomly on the surface, with $(u_i, v_i) \in [0,1]^2$. Furthermore, we present two examples of data fitting starting from point clouds without any given structure where both the quad mesh and a parametrization are computed just starting from the points data.

After having computed a least squares surface $\boldsymbol{G}$ from a point cloud $\mathcal{P}$, let us define the array of errors $\mathbf{err} = \{err_i\}_{i=1}^{nP}$ whose entries are the quantities

$$err_i = \left\| \boldsymbol{G}^{(\ell_i)}(u_i, v_i) - \boldsymbol{P}_i \right\|_2,$$

i.e. the Euclidean distance (ℓ_2 norm) from each point of the cloud and the corresponding value on the surface evaluated at its parameter. From this we define two quantities which will be used to asses the accuracy of the fitting, namely the maximum error and the root mean squared error (RMSE in short):

$$L^\infty = \max_{i=1,\dots,n_P} err_i, \quad \text{RMSE} = \sqrt{\frac{1}{n_P} \sum_{i=1}^{n_P} err_i^2}. \tag{3.25}$$

3.4.1 Point cloud by analytic function evaluation

The experiments we propose here use point cloud data obtained by sampling an input function $f(x, y)$ over a domain identified by a quad mesh $\mathcal{M}$. In Example 3.4.1 we focus on the quality of the fitting showing how it improves when the number of basis functions increases, without the need to use extra smoothness constraints. On the other hand, Example 3.4.2 exhibit the power of our construction when thin plate energy is used in order to obtain an optimal result,

but without increasing the dimension of the spline space. Finally, Example 3.4.3 provide an example of fitting from a point cloud derived by a trivariate function defined over the unit sphere.

Example 3.4.1. For this test, the point cloud is obtained evaluating the function

$$f_T(x,y) = \frac{y}{2\left(\cos(4(x^2+y-1))\right)}, \quad (x,y) \in \mathcal{M}_T \tag{3.26}$$

over the mesh $\mathcal{M}_T$, defining a triangle, formed by 3 patches which present an EV of valence $N = 3$; our sampling produced a point cloud formed by 1536000 points. This example uses no smoothing, which means we fix $\lambda = 0$. From the result in Figure 3.6-(a) can be noticed that constructing our spline space over a coarse mesh as $\mathcal{M}_T$ leads to a fitting surface which is not approximating in a proper way our point set, due to its very oscillatory behavior. This issue can be solved by increasing the number of functions generating the spline space, i.e. increasing the number of patches defining the polygonal domain; in this example the mesh has been refined via Catmull-Clark subdivision.

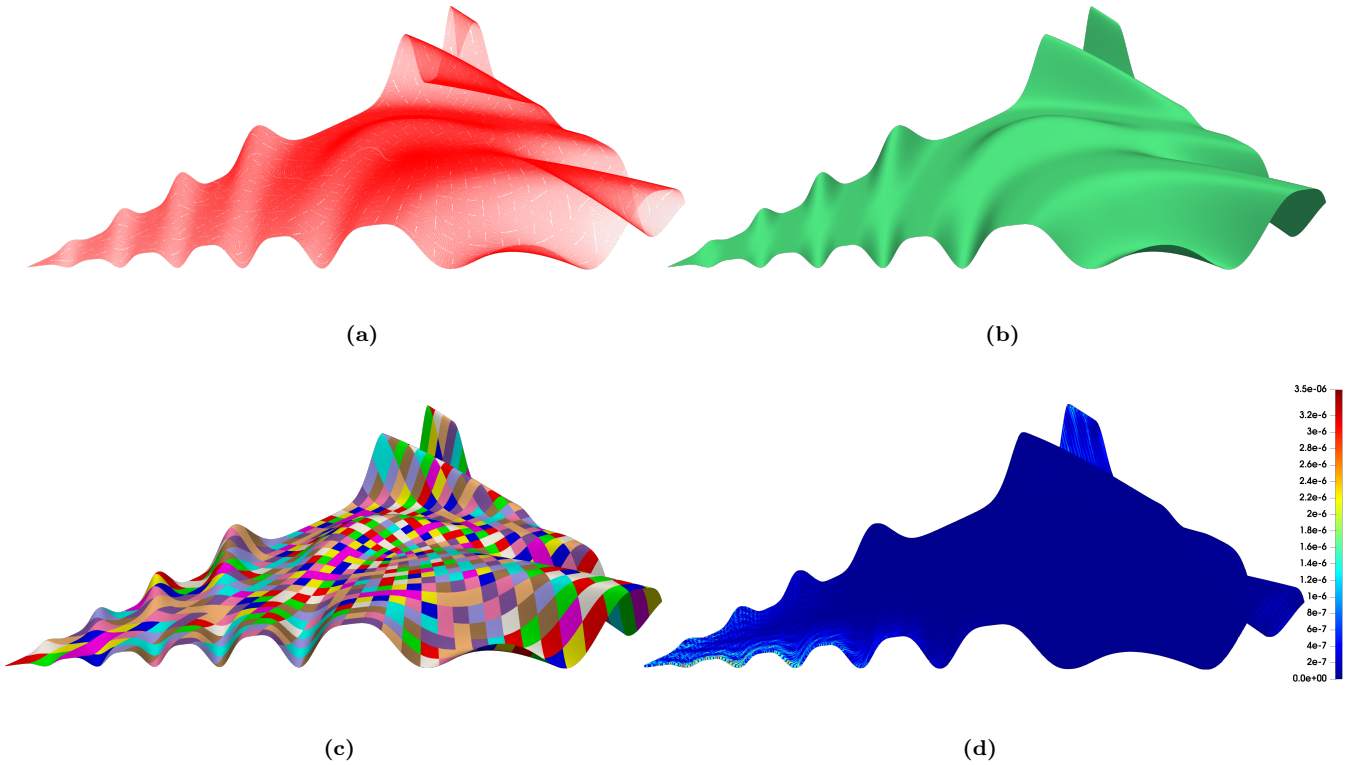


Figure 3.6. (a): final point cloud obtained sampling (3.26) at the last refinement step. (b): resulting surface. (c): multipatch representation of the surface. (d): error color plot of the ℓ_2 distances. The cloud is contained in a bounding box whose longest length is 2.

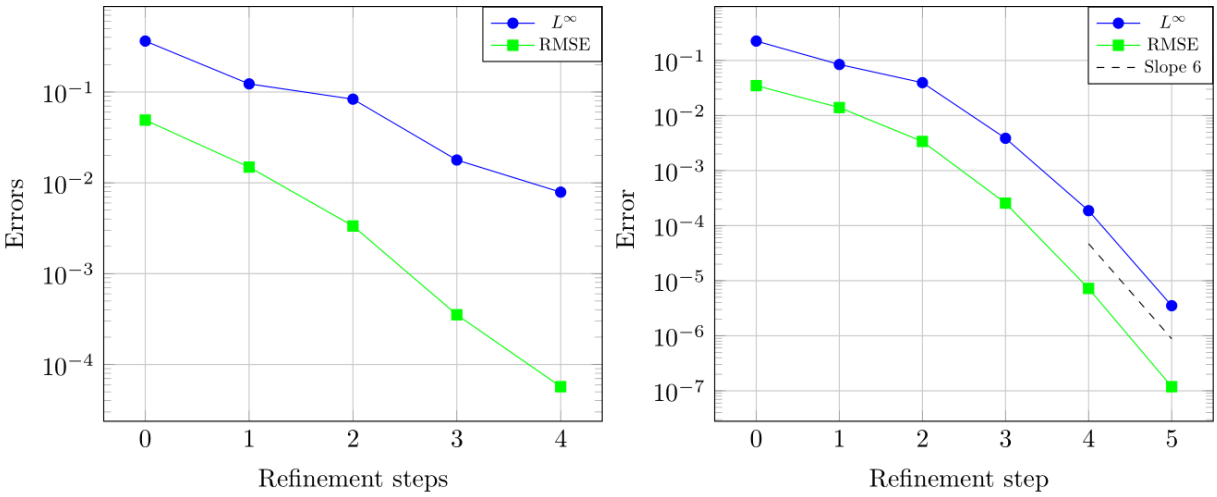
In Table 3.1 and Figure 3.7 are respectively listed and plotted the errors computed from (3.25) for 5 refinement levels. Figure 3.6 shows the input point cloud (a) and the results at the last step of subdivision (b)-(c) together with its error color plot (d), while Table 3.2 and Figure 3.7 present the resulting errors, when the number of points increases along with the dimension

$\dim\left(\mathbb{G}^1(\mathcal{M}_T)\right)$	72	240	864	3264	12672
L^∞ **error**	0.364e-00	0.122e-00	0.834e-01	0.178e-01	0.790e-02
RMSE	0.491e-01	0.149e-01	0.335e-02	0.353e-03	0.570e-04

Table 3.1. Maximal error L^∞ and RMSE for the surfaces in Example 3.4.1 obtained under 5 Catmull-Clark subdivision steps from a point cloud of 150528 elements.

$\dim\left(\mathbb{G}^1(\mathcal{M}_T)\right)$	72	240	864	3264	12672	49920
L^∞ **error**	0.225e-00	0.844e-01	0.340e-01	0.385e-02	0.187e-03	0.351e-05
RMSE	0.349e-01	0.140e-01	0.338e-02	0.256e-03	0.723e-05	0.119e-06

Table 3.2. Maximal error L^∞ and RMSE for the surfaces in Example 3.4.1 obtained under 6 Catmull-Clark subdivision steps with increasing number of elements in the point cloud to estimate the rate of convergence. The point clouds have respectively 1500, 6000, 24000, 96000, 384000 and 1536000 points.



(a) **(b)**

Figure 3.7. Experimental behaviors of errors for Example 3.4.1. (a): errors obtained when increasing the number of basis functions but keeping fix the size of the point cloud. (b): errors obtained when increasing both the number of basis functions but and the size of the point cloud at every refinement step.

of the G^1 spline space. Moreover, we investigate the convergence of the fitting error and, as shown in Figure 3.7, we recovered the optimal convergence rate 6 using biquintic splines.

Example 3.4.2. Here, the cloud data is derived sampling the function

$$f_E(x,y) = \sum_{\nu\in\{-3,0,3\}} \frac{2}{3e^{\sqrt{(10x+\nu)^2+(10y+\nu)^2}}}, \quad (x,y)\in\mathcal{M}_E, \tag{3.27}$$

where $\mathcal{M}_E$ is a quad mesh tracing out an hexagon composed of 96 patches identifying an EV of valence $N=6$ in its middle. The point cloud we obtain is formed of 153600 points. Here we show the power of the smoothing property of this construction: differently from Example 3.4.1, fixing the number of patches i.e. the number of basis function, which is equal to 1725 in our

case, we compute the fitting surface increasing the smoothing parameter λ from 10^{-3} to 10^{-1} with a step of 10^{-1}.

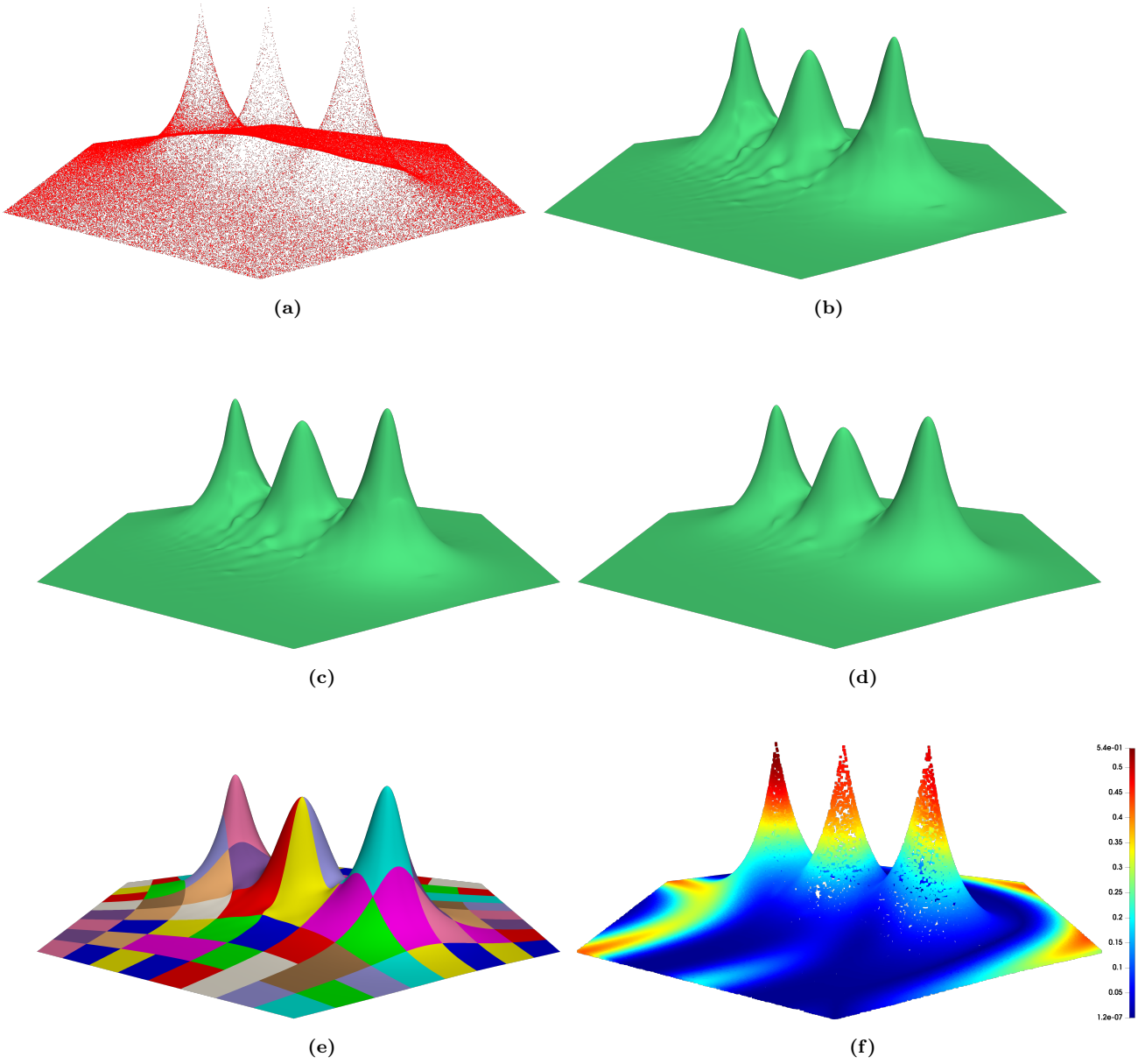


Figure 3.8. (a): point cloud obtained sampling the function in (3.27). (b) to (d): approximating surfaces obtained by using the same number of basis function and point cloud with smoothing parameter $\lambda = 0$, $\lambda = 10^{-2}$ and $\lambda = 10^{-1}$, respectively. (e): multipatch coloring of the surface in (d). (f): error color plot of the ℓ_2 distances for the surface in (d). The cloud is contained in a bounding box whose longest length is 2. Notice how the wrinkles nearby the middle peak diminish when the smoothing parameter increases from (b) to (d).

From Figure 3.8-(b) we notice that the fitting surface presents several wrinkles around the middle peak; by increasing the smoothing factor λ we recover regularity in the output function

which presents much less irregularities, as can be noticed in Figure 3.8-(d). As expected, this procedure will produce at every iteration a smoother function than the previous, but on the other side this forced regularity constraint is reflected in an increase of the three errors in (3.25); this phenomenon is shown in Table 3.3 and graphically in Figure 3.9.

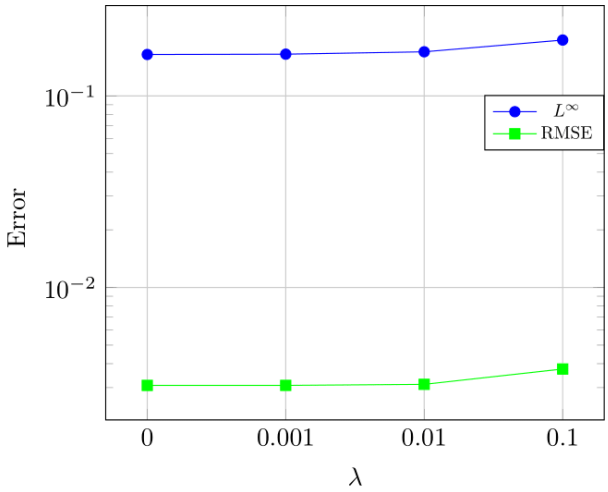


Figure 3.9. Experimental behavior obtained from the errors for Example 3.4.2 when increasing the smoothing parameter λ.

$\boldsymbol{\lambda}$	0	10^{-3}	10^{-2}	10^{-1}
$\boldsymbol{L^\infty}$ **error**	0.164e-00	0.165e-00	0.170e-00	0.196e-00
RMSE	0.308e-02	0.310e-02	0.312e-02	0.375e-02

Table 3.3. Maximal L^∞ error and RMSE for the surfaces in Example 3.4.1 for the surfaces in Example 3.4.2 computed making use of 1725 basis functions and progressively bigger smoothing parameter λ. The maximum length of the box containing the model in Figure 3.8 is equal to two.

Example 3.4.3. In this example, the point cloud is obtained by evaluating a trivariate function defined over the unit sphere $\mathcal{S}^2$. More precisely, the points are sampled from the function

$$f_{\mathcal{S}^2}(x,y,z) = \max\{0, \sin(2\pi x)\sin(2\pi y)\sin(2\pi z)\} + 1, \quad (x,y,z) \in \mathcal{S}^2. \tag{3.28}$$

The data we get is composed of 540000 points, while the set of basis functions has been built over the quad mesh $\mathcal{M}_{\mathcal{S}^2}$ approximating the unit sphere with 96 faces. We notice from the color plot in Figure 3.10-(d) that, understandably, the error is concentrated in the regions where the point cloud has peaks. This is due to the noticeable slopes created by the presence of the max function in (3.28). Table 3.4 shows the numerical results for the errors.

	$\boldsymbol{\dim\left(\mathbb{G}^1(\mathcal{M}_{\mathcal{S}^2})\right)}$	$\boldsymbol{L^\infty}$ **error**	**RMSE**
$f_{\mathcal{S}^2}$	1512	0.539e-01	0.107e-01

Table 3.4. Spline space dimension, maximal error L^∞ and RMSE for the fitting presented in Example 3.4.3. The point cloud in Figure 3.10 is surrounded by a box with longest size 3.

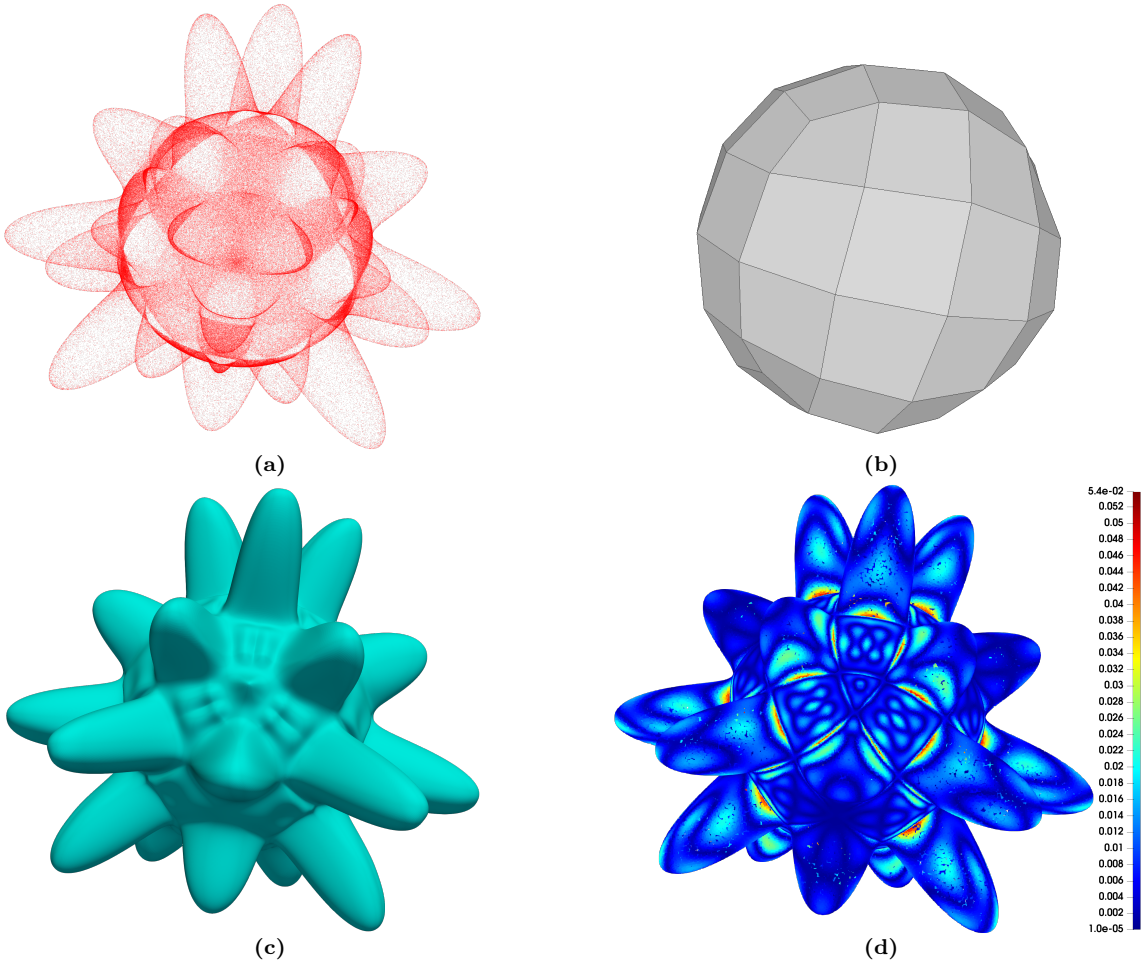
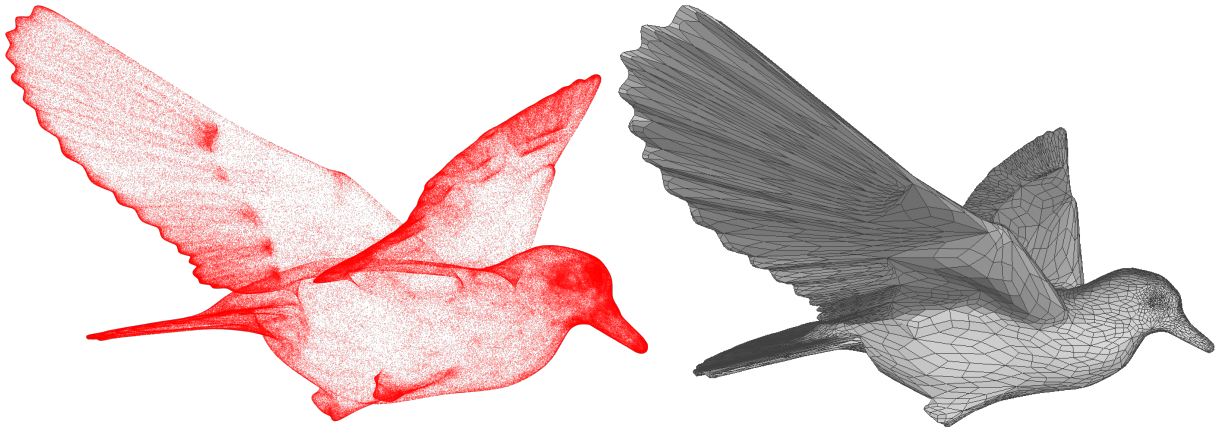


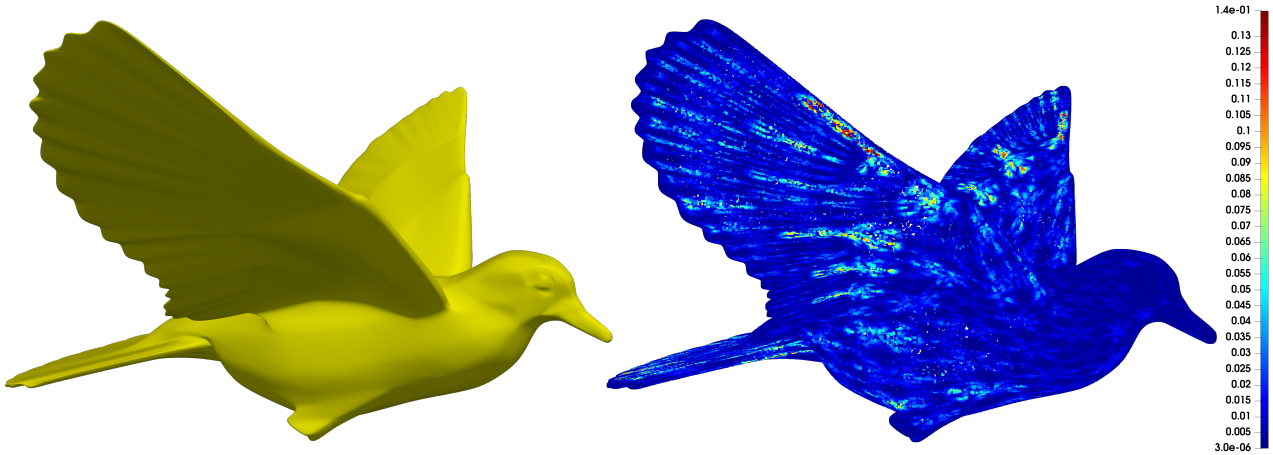
Figure 3.10. (a): point cloud obtained sampling the trivariate function defined in (3.28) over the unit sphere $\mathcal{S}^2$. (b): quad mesh used to define the G^1 space. (c): final least squares surface. (d): error color plot of the ℓ_2 distances. The cloud is contained in a bounding box whose longest length is 3.

3.4.2 Point cloud from ACC_3 surfaces

We now present fitting examples obtained from big data sets. The point clouds utilized in this section are provided by randomly sampling the Approximate Catmull-Clark surfaces (ACC_3) obtained from the construction in [LS08]. Figures 3.11 to 3.16 show the data we use for our investigation obtained from the ACC_3 surfaces, starting from parameters sampling: the dimension of the clouds goes from a minimum of 549180 to a maximum of 968704 points. In all the experiments presented here we do not consider any smoothing parameter, i.e. $\lambda = 0$, but few iterations of parameter correction [Hos88] are performed to optimize the fitting error; the errors in Table 3.5 are obtained using the formulas in (3.25). Since ACC_3 surfaces are not G^1 at the EVs, it is not surprising to see that the maximal fitting error is located around the extraordinary vertices.



(a) (b)

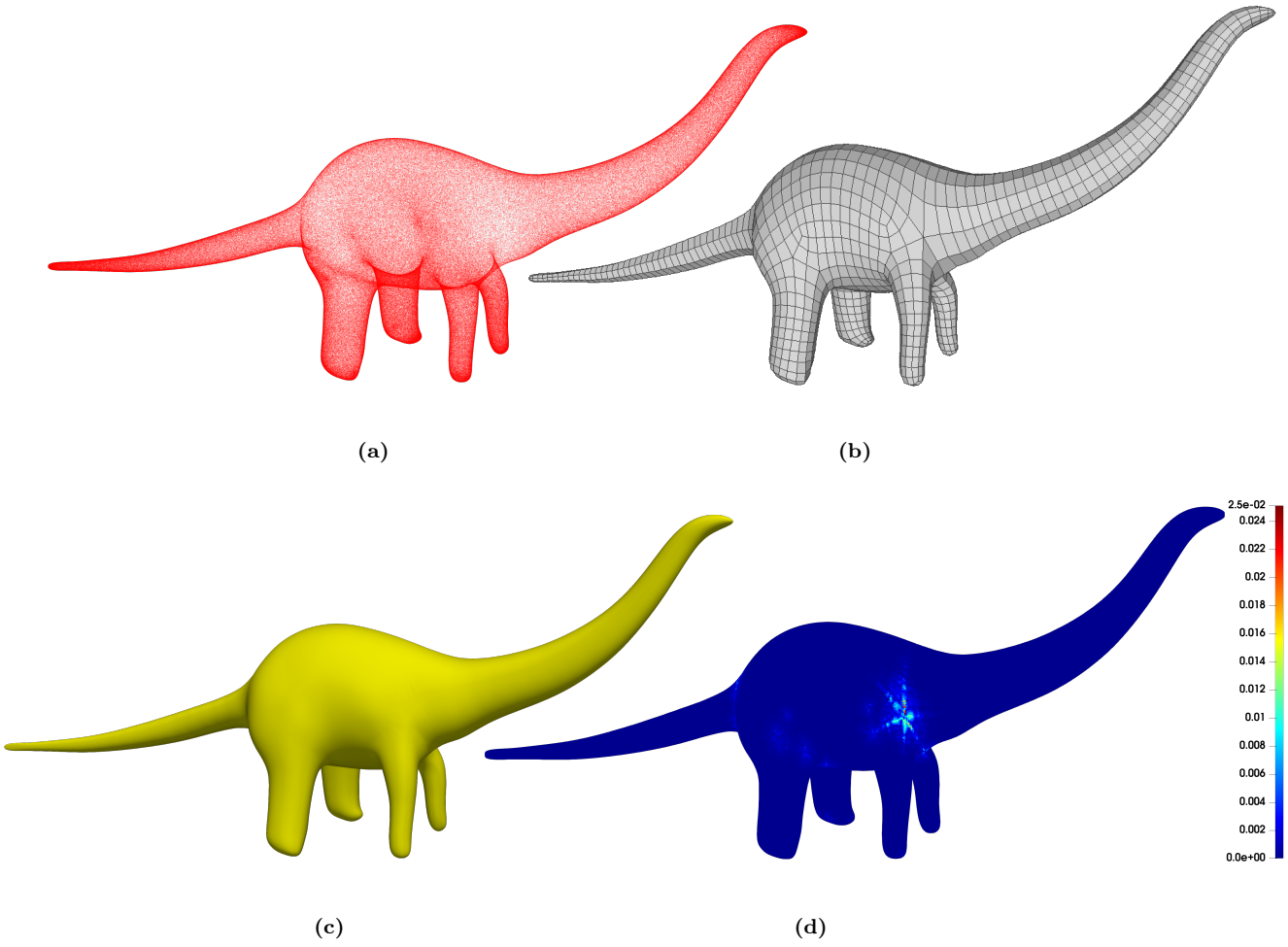


(c) (d)

Figure 3.11. Bird. (a): point cloud obtained sampling an ACC_3 surface. (b): quad mesh used to define the G^1 space. (c): final least squares surface. (d): error color plot of the ℓ_2 distances. The cloud is contained in a bounding box whose longest length is 100.

	L^∞ error		RMSE			
	no p.c.	3 × p.c.	no p.c.	3 × p.c.	$\dim\left(\mathbb{G}^1(\mathcal{M})\right)$	n_P
bird	0.138e-00	0.754e-01	0.325e-02	0.648e-03	100386	549180
dinosaur	0.251e-01	0.162e-01	0.348e-03	0.205e-03	29885	680124
hammer	0.667e-02	0.555e-02	0.138e-03	0.596e-04	48580	685800
hand	0.417e-01	0.320e-01	0.616e-03	0.341e-03	28227	575424
rabbit	0.120e-01	0.114e-01	0.302e-03	0.145e-03	60220	968704
venus	0.131e-01	0.611e-02	0.210e-03	0.610e-04	55018	676592

Table 3.5. Maximal error L^∞ and RMSE obtained both without and with 3 iterations of parameter correction (p.c.) for the fitting examples presented in Figures 3.11 to 3.16 with a description of the spline space and point cloud features. All the above mentioned models are contained in a box whose longest length is 100.



(a) **(b)**

(c) **(d)**

Figure 3.12. Dinosaur. (a): point cloud obtained sampling an ACC_3 surface. (b): quad mesh used to define the G^1 space. (c): final least squares surface. (d): error color plot of the ℓ_2 distances. The cloud is contained in a bounding box whose longest length is 100.

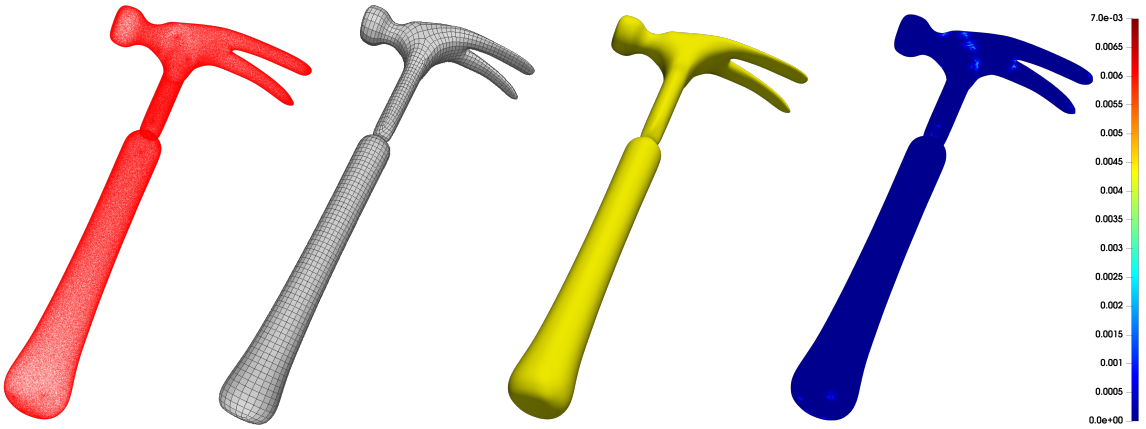


Figure 3.13. Hammer. (a): point cloud obtained sampling an ACC_3 surface. (b): quad mesh used to define the G^1 space. (c): final least squares surface. (d): error color plot of the ℓ_2 distances. The cloud is contained in a bounding box whose longest length is 100.

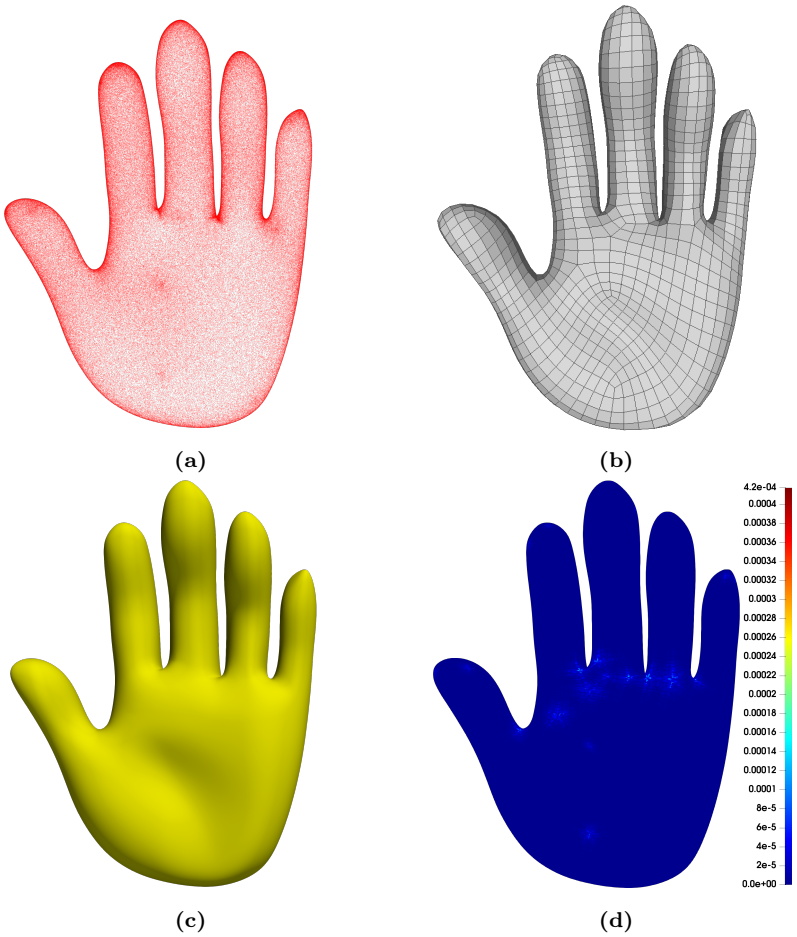


Figure 3.14. Hand. (a): point cloud obtained sampling an ACC_3 surface. (b): quad mesh used to define the G^1 space. (c): final least squares surface. (d): error color plot of the ℓ_2 distances. The cloud is contained in a bounding box whose longest length is 100.

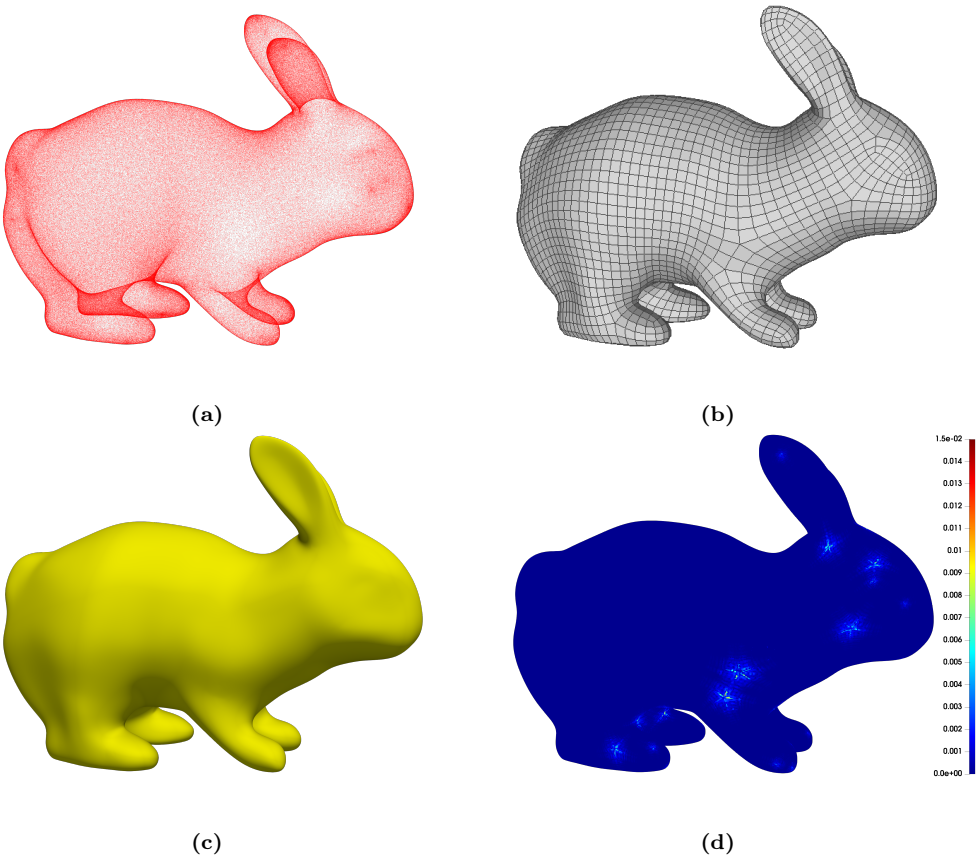


Figure 3.15. Rabbit. (a): point cloud obtained sampling an ACC_3 surface. (b): quad mesh used to define the G^1 space. (c): final least squares surface. (d): error color plot of the ℓ_2 distances. The cloud is contained in a bounding box whose longest length is 100.

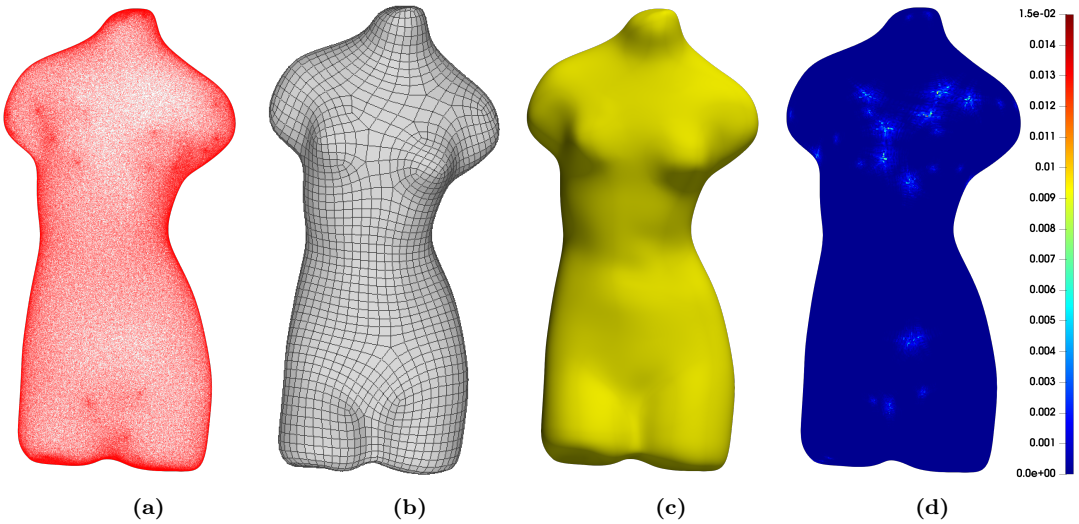


Figure 3.16. Venus. (a): point cloud obtained sampling an ACC_3 surface. (b): quad mesh used to define the G^1 space. (c): final least squares surface. (d): error color plot of the ℓ_2 distances. The cloud is contained in a bounding box whose longest length is 100.

3.4.3 Quadrilateral mesh generation, parametrization and fitting

In the previous experiments, all the point clouds were already equipped with a parametrization. This is because the goal of this chapter is to show the quality of the fitting with the basis functions we propose, rather than improving upon parameter computation. Nevertheless, we now present few examples in which both the parametrization of an unorganised point cloud and the construction of a coarse quadrilateral mesh are derived. Starting from an input point cloud, we first reconstruct a triangular mesh by using the algorithm in [Ber99]; then we apply the so-called 4-8 subdivision scheme [VZ01] to obtain a quadrangular representation of the previous triangular mesh. This procedure leads to a mesh, which is much more refined than we would need for our computation. Hence, we coarsen it using RHINO3D command *QuadRemesh* (see [McN10]), which reduces the number of faces of the quad mesh without modifying its topology. This quad mesh $\mathcal{M}$ is the mesh on which we define the G^1 space and basis functions.

In order to associate parameters to a point $P_i = (x_i, y_i, z_i)$ of the point cloud, we compute its orthogonal projection $P_i^\perp = (x_i^\perp, y_i^\perp, z_i^\perp)$ onto the G^1 spline surface $\boldsymbol{G}$ obtained from the construction in Chapter 2. Assume that the point $P_i^\perp$ is laying on the ℓ-th patch of $\boldsymbol{G}$. Let $\boldsymbol{G}^{(\ell)}$ be the geometry map associated with the patch ℓ defined in (3.23). We define the parameters (u_i, v_i) associated with the point P_i as

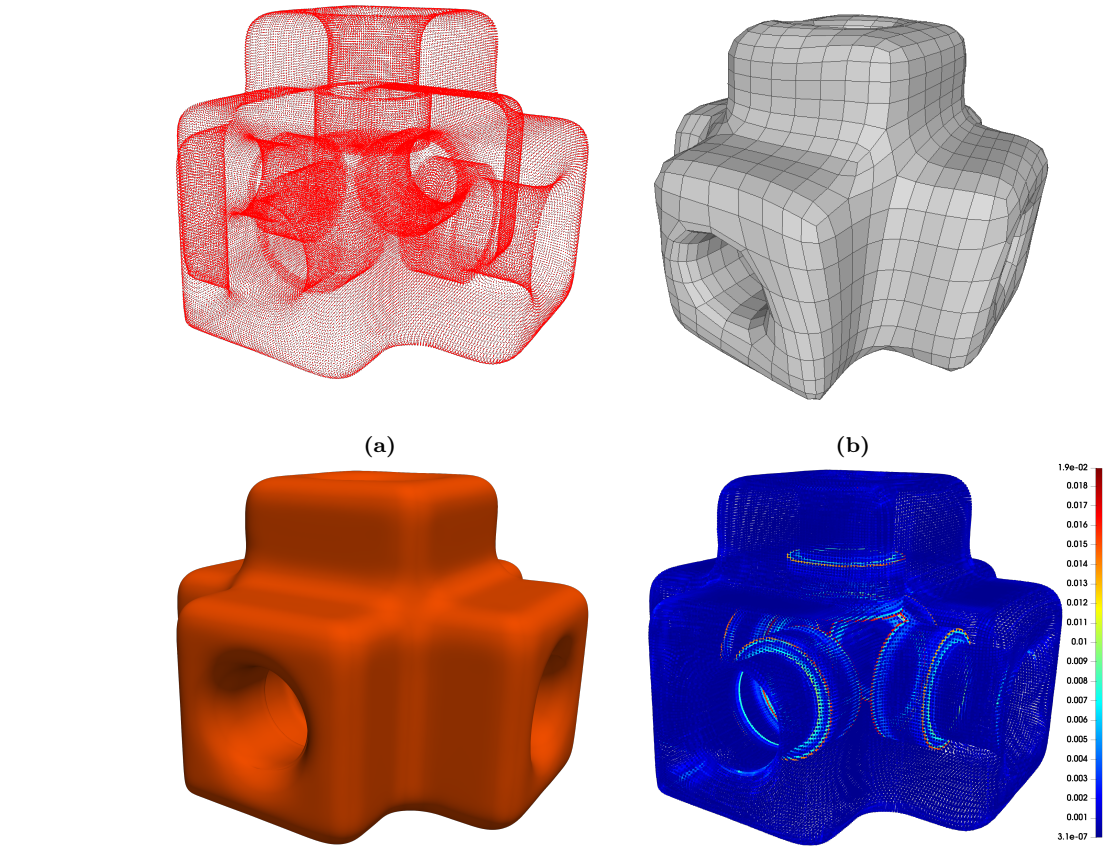
$$(u_i, v_i) = \left(\boldsymbol{G}^{(\ell)}\right)^{-1} (P_i^\perp);$$

This parameter computation is repeated for all the points of the point cloud.

Figures 3.17, 3.19 and 3.21 present three examples of point cloud fitting obtained using the pipeline presented in this section. Particular interest goes for the example in Figure 3.19, whose initial point cloud presents a very uneven distribution of points. Moreover, in all the numerical examples a regularization parameter $\lambda = 10^{-1}$ has been used in order to have a well-posed problem. The compression we obtain in these examples is smaller than the one in the experiments presented in Section 3.4.2; it could be increased doing more coarsening on the quad mesh but we didn't go further in this direction since the aim of the subsection is the general parametrization and mesh generation. Table 3.6 summarizes the key points of these experiments.

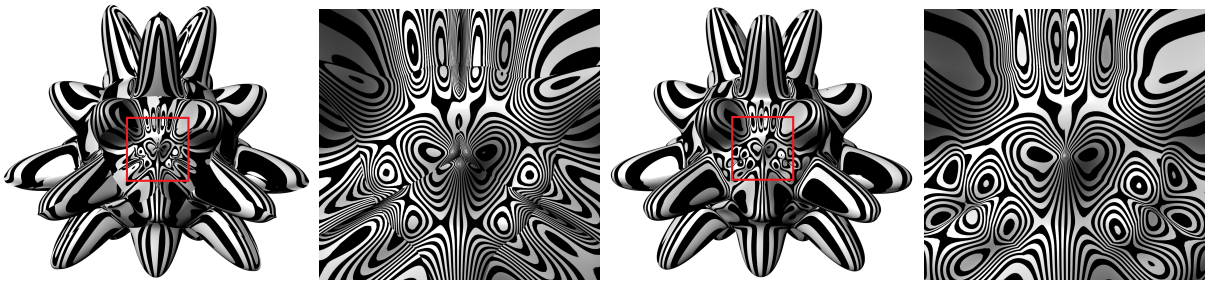
	n_P	n_F coarse quad mesh	$\dim\left(\mathbb{G}^1(\mathcal{M})\right)$	L^∞ error	RMSE
uniform toy	113846	2104	33280	0.190e-01	0.248e-02
nonuniform toy	30908	1860	29391	0.250e-01	0.277e-02
icosahedron	208708	7892	125205	0.189e-01	0.142e-02

Table 3.6. Points in the input cloud, quad mesh and corresponding spline space details, maximal error L^∞ and RMSE for the fitting examples in Figures 3.17, 3.19 and 3.21. All the models are contained in a bounding box whose longest length is 2.



(a) **(b)** **(c)** **(d)**

Figure 3.17. Uniform toy. (a): input point cloud. (b): reconstructed quad mesh used to define the G^1 space. (c): final least squares surface. (d): error color plot of the ℓ_2 distances. The cloud is contained in a bounding box whose longest length is 2.



(a) **(b)** **(c)** **(d)**

Figure 3.18. Reflection lines on the C^0 (a) and G^1 (c) surfaces obtained from the point cloud in Figure 3.10-(a) and relative zooms, (b) and (d), around an EV.

3.4.4 Comparison with C^0 fitting

Here we present a comparison between surfaces obtained by fitting a point cloud using our proposed G^1 basis functions and using the standard C^0 Bernstein basis on (the faces of) a quad mesh $\mathcal{M}$, whose space is denoted by $\mathbb{C}^0(\mathcal{M})$. In Table 3.7 we can notice that, despite the higher dimension of the continuous basis functions, the errors we get are comparable with the ones obtained from the G^1 construction using fewer basis functions; Figures 3.18 and 3.20 present the reflection lines for the treated examples.

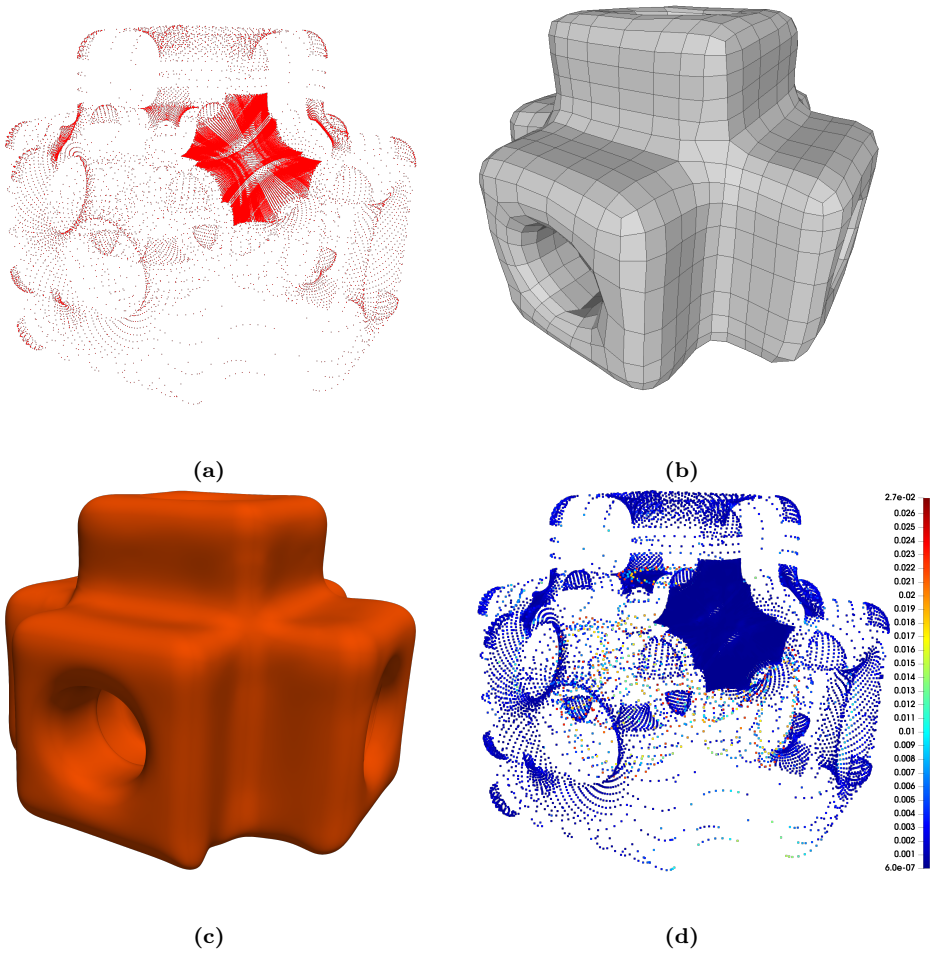
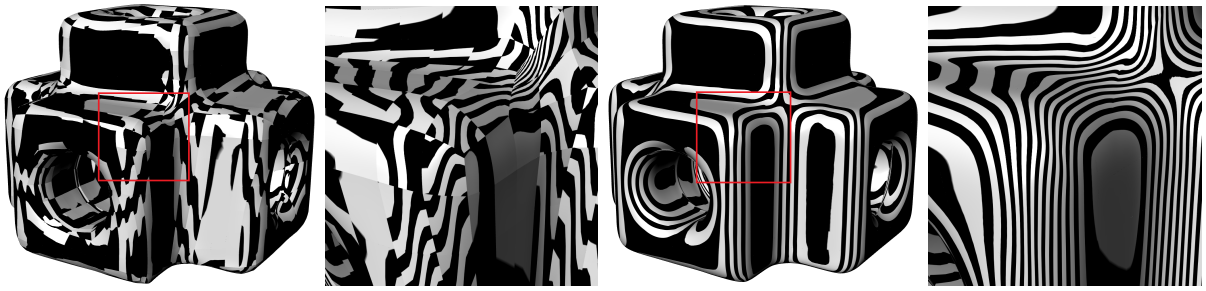


Figure 3.19. Nonuniform toy. (a): input point cloud. (b): reconstructed quad mesh used to define the G^1 space. (c): final least squares surface. (d): error color plot of the ℓ_2 distances. The cloud is contained in a bounding box whose longest length is 2.

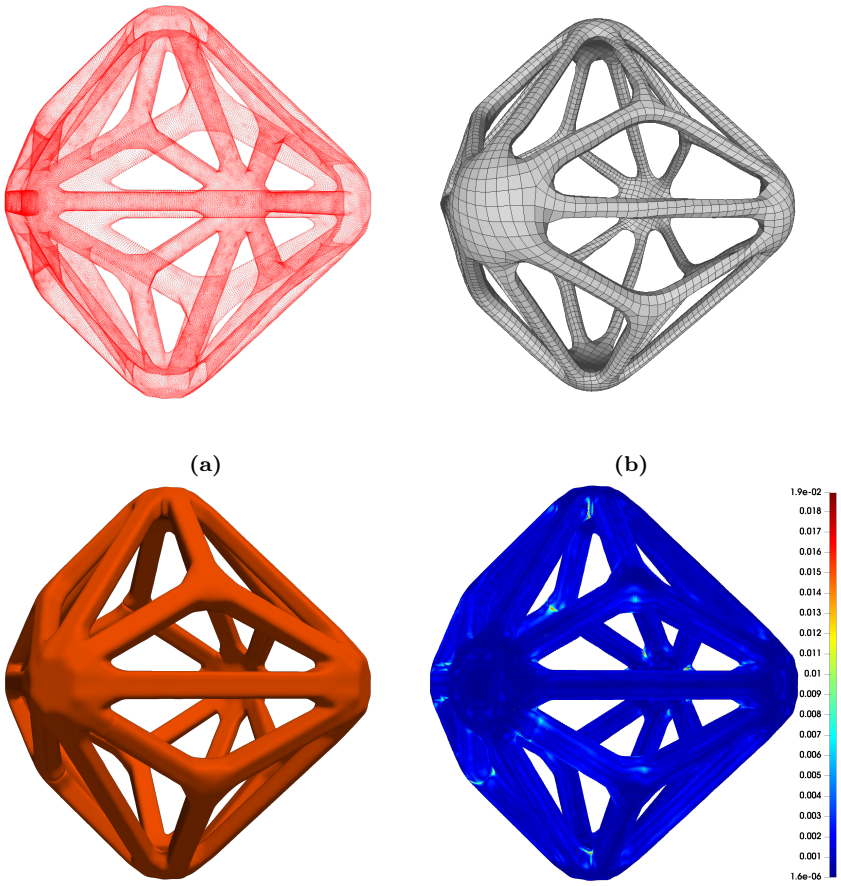
	$\dim\left(\mathbb{C}^0(\mathcal{M})\right)$	$\dim\left(\mathbb{G}^1(\mathcal{M})\right)$	$L^\infty_{C^0}$ error	$L^\infty_{G^1}$ error	RMSE_{C^0}	RMSE_{G^1}
$f_{\mathcal{S}^2}$	2402	1512	0.455e-01	0.539e-01	0.728e-02	0.107e-01
uniform toy	52602	33280	0.165e-01	0.190e-01	0.224e-02	0.248e-02

Table 3.7. Comparison between C^0 and G^1 dimensions and errors for the examples in Figure 3.10 and 3.17.



(a) **(b)** **(c)** **(d)**

Figure 3.20. Reflection lines on the C^0 (a) and G^1 (c) surfaces obtained from the point cloud in Fig. 3.17-(a) and relative zooms, (b) and (d), around an EV.



(a) **(b)**

(c) **(d)**

Figure 3.21. Icosahedron. (a): input point cloud. (b): reconstructed quad mesh used to define the G^1 space. (c): final least squares surface. (d): error color plot of the ℓ_2 distances. The cloud is contained in a bounding box whose longest length is 2.

Summary

In the present chapter we described an easy procedure to build biquintic G^1 smooth Bézier basis functions over quadrangular meshes to be utilized in point cloud fitting problems. Starting from the equations expressing the geometric continuity constraints obtained making use of quadratic gluing data, we performed an extraction approach, based on the different degrees of freedom present in the system, leading to linear systems returning the control point defining the sought functions. Moreover, we provided an analysis of the latter basis functions as well as a dimension formula for the spline space they generate. We concluded providing numerical experiments establishing their quality in point data fitting.

Chapter 4

Analysis-suitable G^1 bases for isogeometric analysis

Complex geometries are commonplace when dealing with PDEs simulations arising from real-life applications; from this there is a need to have robust and analysis-suitable constructions to deal with such shapes. In this chapter we propose a new set of analysis-suitable G^1 spline basis functions for isogeometric analysis simulations on arbitrary domains $\Omega \subset \mathbb{R}^3$.

After recalling the generic constraints ensuring G^1 continuity between Bézier patches, we present the construction of vertex, edge and face functions and we provide a dimension formula for the spline space generated by such basis. Furthermore, we produce numerical experiments to ensure their suitability in IGA simulations on various PDEs. This chapter is based on [MMM23].

Motivation

The basis functions introduced in Chapter 3 do not behave optimally when applied in the numerical resolution of PDEs on successive approximation of Catmull-Clark surfaces. This fact can be noticed from the plot in Figure 4.1, which reports the H^1 errors we obtain when solving the Poisson's equation (cf. (1.12) or following Section 4.4.2) over the domains obtained applying the G^1 construction introduced in Chapter 2 to the meshes in Figure 4.9. Indeed, except for the case of valence $N = 4$ that corresponds to the regular case, the other errors do not decrease, as expected in the optimal case, by a rate of 5. The reason behind this suboptimal result can be found in the refinement procedure we use during the simulations. By definition, the basis functions in Chapter 3 only allow the construction of a sequence of isogeometric spaces by iteratively splitting each quadrilateral mesh face regularly into four quadrilaterals (we opted for the Catmull-Clark split) and consequently reparametrizing via the G^1 framework in Chapter 2. It is easy to realize that the spaces obtained during this procedure are not nested. Moreover, this approach modifies the geometry, which changes at every refinement step. A rigorous explanation of why this suboptimal behaviour arises can be found in [Tak23].

The goal of this chapter is to define a spline space and a basis over quad meshes allowing refinement techniques that don't change the geometry of the desired domain nor require a reparametrization at every step. This can be achieved, for example, and it is what we actually do, by equipping the construction with the possibility of using knot insertion. With this approach the resulting spline space will turn out to be nested and the domain's geometry is fixed.

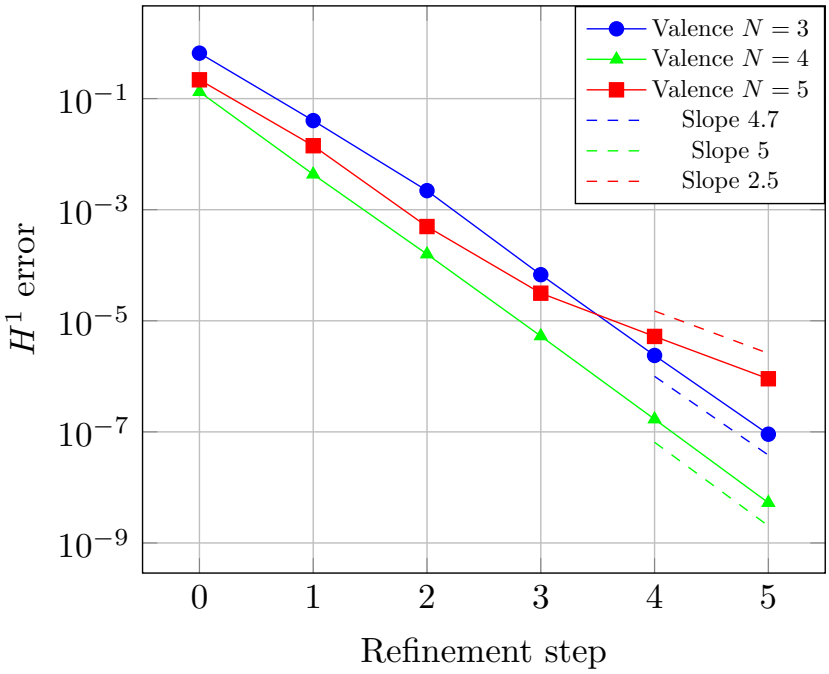


Figure 4.1. Convergence plot of the error for the Poisson's problem when using the basis functions introduced in Chapter 3. Except for the regular case $N = 4$, we obtain a suboptimal rate.

4.1 General formulation of G^1 conditions

In order to describe our construction, we need to deal with G^1 constraints across Bézier patches defined over quad faces where, eventually, two linked EVs may occur. Actually, as will be explained in the next section, we will deal with EVs attached to valence 4 vertices whose edges are not all orthogonal to their adjacent ones; this vertex takes the name of *non-crossing regular vertex* (non-crossing RV in short) and, for our purposes, it will be treated like an EV. We assume again that the input quad mesh $\mathcal{M}$ presents isolated EVs. Similarly to the constructions in Chapter 2 and Chapter 3 we use here quadratic symmetric gluing data of the form

$$\begin{aligned}\mathfrak{a}_{N,N'}(u) &= a_0 B_2^0(u) + a_1 B_2^1(u) + a_2 B_2^2(u),\\ \mathfrak{b}_{N,N'}(u) &= -1,\end{aligned}$$

where the B_d^i are the univariate Bernstein polynomials in (1.3) and the $a_i \in \mathbb{R}$, $i = 1, 2, 3$, are coefficients that we will define later on. As is well known, a Bézier patch is a particular case of tensor-product B-spline whose knot vectors are formed by zeros and ones only; for the biquintic case we are dealing with, the knot vector $\mathbf{t}_{\mathrm{B}}$ defining the grid $\mathbf{t}_{\mathrm{B}} \times \mathbf{t}_{\mathrm{B}}$ on which our patch lives is

$$\mathbf{t}_{\mathrm{B}} = \{0,0,0,0,0,0,1,1,1,1,1,1\} = \{\mathbf{0}^6, \mathbf{1}^6\}. \tag{4.1}$$

In this setting, specializing (1.10) for two Bézier functions with $\mathrm{b}_{i,j}^{(k)}$ their control points defining the k-th patch labelled as in Figure 2.2, the system defining the G^1 constraints is defined by

$$\begin{aligned}
\mathrm{b}_{0,1}^{(1)} + \mathrm{b}_{1,0}^{(0)} &= (2-a_0)\mathrm{b}_{0,0}^{(1)} + a_0\mathrm{b}_{1,0}^{(1)}, && (4.2)\\
5(\mathrm{b}_{1,1}^{(1)} + \mathrm{b}_{1,1}^{(0)}) &= (a_0-2a_1)\mathrm{b}_{0,0}^{(1)} + (10-5a_0+2a_1)\mathrm{b}_{1,0}^{(1)} + 4a_0\mathrm{b}_{2,0}^{(1)}, && (4.3)\\
10(\mathrm{b}_{2,1}^{(1)} + \mathrm{b}_{1,2}^{(0)}) &= (-a_0+2a_1-a_2)\mathrm{b}_{0,0}^{(1)} + (5a_0-10a_1+a_2)\mathrm{b}_{1,0}^{(1)} + 2(-5a_0+4a_1+10)\mathrm{b}_{2,0}^{(1)}\\
&\quad + 3a_0\mathrm{b}_{3,0}^{(1)}, && (4.4)\\
10(\mathrm{b}_{3,1}^{(1)} + \mathrm{b}_{1,3}^{(0)}) &= (a_0-2a_1+a_2)\mathrm{b}_{0,0}^{(1)} + 5(-a_0+2a_1-a_2)\mathrm{b}_{1,0}^{(1)} + 2(5a_0-10a_1+2a_2)\mathrm{b}_{2,0}^{(1)}\\
&\quad + 2(-5a_0+6a_1+10)\mathrm{b}_{3,0}^{(1)} + 4a_0\mathrm{b}_{4,0}^{(1)}, && (4.5)\\
5(\mathrm{b}_{4,1}^{(1)} + \mathrm{b}_{1,4}^{(0)}) &= (2a_1-a_2)\mathrm{b}_{0,0}^{(1)} - (10a_1-a_2)\mathrm{b}_{1,0}^{(1)} + 4(2a_1-a_2)\mathrm{b}_{2,0}^{(1)} - 2(10a_1-3a_2)\mathrm{b}_{3,0}^{(1)}\\
&\quad + 2(4a_1+5)\mathrm{b}_{4,0}^{(1)}, && (4.6)\\
\mathrm{b}_{5,1}^{(1)} + \mathrm{b}_{1,5}^{(0)} &= (2+a_2)\mathrm{b}_{5,0}^{(1)} - a_2\mathrm{b}_{4,0}^{(1)}, && (4.7)\\
10(\mathrm{b}_{3,0}^{(1)} - \mathrm{b}_{2,0}^{(1)}) &= \mathrm{b}_{0,0}^{(1)} - 5\mathrm{b}_{1,0}^{(1)} + 5\mathrm{b}_{4,0}^{(1)} - \mathrm{b}_{5,0}^{(1)}. && (4.8)
\end{aligned}$$

These conditions, similarly to (3.9), can be reformulated in matrix form following the notation introduced in Section 3.2 as

$$M_{G^1}\mathbf{b} = \mathbf{0}, \tag{4.9}$$

where $M_{G^1} = (M_1|M_2)$ is a block matrix with

$$M_1 = \begin{pmatrix}
-(2-a_0)\mathbf{u} & C_1 & & & \\
-(a_0-2a_1)\mathbf{u} & (10-5a_0+2a_1)I & -4a_0I & 5C_2 & \\
-\mathbf{u} & 5I & 10I & & -10I\\
(a_0-2a_1+a_2)\mathbf{u} & -(5a_0-10a_1+a2)I & 2(5a_0-4a_1-10)I & & -3a_0I\\
-(a_0-2a_1+a_2)\mathbf{u} & 5(a_0-2a_1+a_2)I & -2(5a_0-10a_1+2a_2)I & & 2(5a_0-6a_1-10)I\\
-(2a_1-a_2)\mathbf{u} & (10a_1-a_2)I & -4(2a_1-a_2)I & & 2(10a_1-3a_2)I
\end{pmatrix}$$

$$M_2 = \begin{pmatrix}
10I & 10I & & & & & & & \\
 & & -4a_0I & 10I & 10I & & & & \\
 & & -2(4a_1+5)I & & & 5I & 5I & & \\
 & & a_2I & & & & & -(2+a_2)I & I & I
\end{pmatrix}$$

and $\mathbf{b}$ the vector

$$\mathbf{b} = \left(\mathbf{b}_{0,0}\,\mathbf{b}_{1,0}\,\mathbf{b}_{2,0}\,\mathbf{b}_{1,1}\,\mathbf{b}_{3,0}\,\mathbf{b}_{2,1}\,\widehat{\mathbf{b}}_{1,2}\,\mathbf{b}_{4,0}\,\mathbf{b}_{3,1}\,\widehat{\mathbf{b}}_{1,3}\,\mathbf{b}_{4,1}\,\widehat{\mathbf{b}}_{1,4}\,\mathbf{b}_{5,0}\,\mathbf{b}_{5,1}\,\widehat{\mathbf{b}}_{1,5}\right)^T,$$

with $\mathbf{b}_{i,j} = \left(\mathrm{b}_{i,j}^{(k)}\right), k = 1,\ldots,N.$

We notice that when $a_1 = a_2 = 0$, system (4.2) to (4.8) reduces to system (2.4) to (2.10).

Given two biquintic patches defined over a generic tensor-product grid $\mathbf{t} \times \mathbf{t}$, with $\mathbf{t} = \{\mathbf{0}^6,\ t_1 \leq t_2 \leq \ldots \leq t_s,\ \mathbf{1}^6\}$, the equations defining the G^1 joins will depend also on the values of the knots (and their combinations), making them harder to be solved. In fact, calling with m the number of control points laying on the common edge shared by two patches, the G^1 equations on a general knot vector can be sketched as

$$\mathrm{b}^{(1)}_{0,1} + \mathrm{b}^{(0)}_{1,0} = (2 - a_0)\mathrm{b}^{(1)}_{0,0} + a_0\mathrm{b}^{(1)}_{1,0}, \tag{4.10}$$
$$\mathrm{b}^{(1)}_{1,1} + \mathrm{b}^{(0)}_{1,1} = \mathcal{L}_{1,1}(\mathrm{b}^{(1)}_{0,0}, \ldots, \mathrm{b}^{(1)}_{m-1,0}\,; t_i, a_0, a_1, a_2), \tag{4.11}$$
$$\mathrm{b}^{(1)}_{2,1} + \mathrm{b}^{(0)}_{1,2} = \mathcal{L}_{2,1}(\mathrm{b}^{(1)}_{0,0}, \ldots, \mathrm{b}^{(1)}_{m-1,0}\,; t_i, a_0, a_1, a_2), \tag{4.12}$$
$$\vdots$$
$$\mathrm{b}^{(1)}_{m-3,1} + \mathrm{b}^{(0)}_{1,m-3} = \mathcal{L}_{m-3,1}(\mathrm{b}^{(1)}_{0,0}, \ldots, \mathrm{b}^{(1)}_{m-1,0}\,; t_i, a_0, a_1, a_2), \tag{4.13}$$
$$\mathrm{b}^{(1)}_{m-2,1} + \mathrm{b}^{(0)}_{1,m-2} = \mathcal{L}_{m-2,1}(\mathrm{b}^{(1)}_{0,0}, \ldots, \mathrm{b}^{(1)}_{m-1,0}\,; t_i, a_0, a_1, a_2), \tag{4.14}$$
$$\mathrm{b}^{(1)}_{m-1,1} + \mathrm{b}^{(0)}_{1,m-1} = \mathcal{L}_{m-1,1}(\mathrm{b}^{(1)}_{m-2,0}, \mathrm{b}^{(1)}_{m-1,0}\,; t_i, a_2), \tag{4.15}$$

with $t_i \in \mathbf{t}$, $i = 1, \ldots, s$, and where we indicate with $\mathcal{L}_{j,1}(\cdot\,;\cdot)$ the linear combination of the variables present in the first argument by the coefficients present in the second argument concerning the j-th pair of control points. In addiction to (4.10) to (4.15) we also need to consider edge constraints, i.e. linear combinations of control points on the edge, which are as many as the knot spans present in $\mathbf{t}$.

Let now assume that our knot vector is only formed by s uniform inner knots with multiplicity 5 (e.g. equal to the degree of the patch), that is

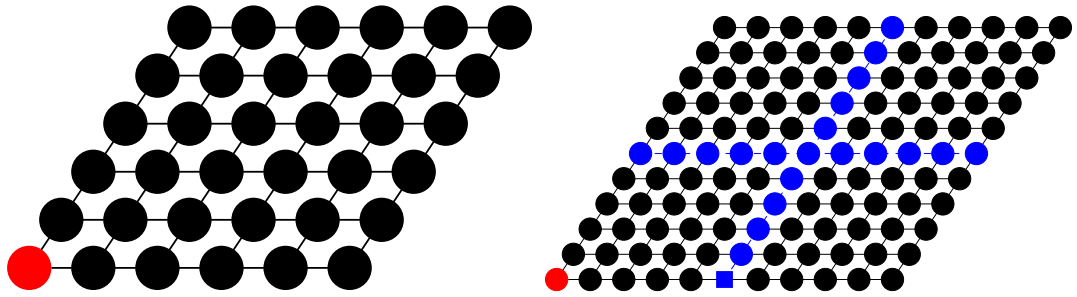
$$\mathbf{t} = \{\mathbf{0}^6, \mathbf{t}_1^5 < \mathbf{t}_2^5 < \cdots < \mathbf{t}_s^5, \mathbf{1}^6\}, \tag{4.16}$$

where we followed the notation in (4.1). Under this hypothesis, the spline spaces associated to the knot vectors will be nested and, as consequence of the knot insertion algorithm, the patch will be split in several biquintic Bézier components which joins C^0 between them; moreover this split procedure gives rise to valence 4 points on the edge which can be seen as extraordinary vertices that are the so called non-crossing RV, in which the gluing data functions are nonzero. We will refer to them as virtual EVs. Finally, in order to have enough regularity in our space, we will impose extra C^1 continuity across the split lines of each patch. Figure 4.2 shows an example of patch with no split and with one split.

Using (4.2) to (4.8) (e.g. system (4.9)) and the construction presented in Section 3.2, we provide in the next section the derivation of a set of basis function spanning the G^1 spline space which result to be analysis-suitable for IGA simulations.

4.2 Construction of the basis

We present here the construction of a set of basis functions generating the G^1 spline space over a quad mesh $\mathcal{M}$ associated to the knot vector $\mathbf{t}$ as in (4.16). We will refer to this set as $\mathcal{B}^{\mathbf{t}}$, to stress its dependency on the knot vector. Similarly to what we did in Section 3.2, we will follow the topology of the input mesh $\mathcal{M}$ in order to separate the set of functions attached to



(a) (b)

Figure 4.2. (a): control points in a Bézier patch with knots $\mathbf{t} = \{\mathbf{0}^6, \mathbf{1}^6\}$. The red dot identifies the EV. (b): control points in a patch with a 4-split given by the knots $\mathbf{t} = \{\mathbf{0}^6, \mathbf{1/2}^5, \mathbf{1}^6\}$. The red dot identifies the EV, the blue square the virtual EVs while the blue dots outline the C^0 split lines.

the vertices $\mathcal{B}_V^{\mathbf{t}}$, set of basis functions attached to the edges $\mathcal{B}_E^{\mathbf{t}}$ and set of functions attached to the faces $\mathcal{B}_F^{\mathbf{t}}$. Hence, similarly to (3.8) we can decompose the set of basis functions as

$$\mathcal{B}^{\mathbf{t}} = \left(\bigcup_{i=1}^{n_V} \mathcal{B}_{V_i}^{\mathbf{t}}\right) \cup \left(\bigcup_{i=1}^{n_E} \mathcal{B}_{E_i}^{\mathbf{t}}\right) \cup \left(\bigcup_{i=1}^{n_F} \mathcal{B}_{F_i}^{\mathbf{t}}\right). \tag{4.17}$$

Depending on the basis functions we want to compute, the strategy we exploit is different. Regarding the vertex functions, their control points are obtained via knot insertion starting from the vertex basis functions defined on a Bézier knot vector, while the edge basis is computed concatenating around a virtual EV the relations (4.2) to (4.8), thanks to the particular shape of the patch we obtained under the assumption (4.16). The functions related to the faces are again the standard C^1 Bézier basis. Let investigate these constructions more in details.

4.2.1 Vertex basis functions: the set $\mathcal{B}_V^{\mathbf{t}}$

To this set belong basis functions connected to inner and boundary EVs, regular vertices as well as corners. We only present the construction for functions supported around an EV since the other cases are analogous or already presented in Section 3.2.1.

4.2.1.1 Construction of basis functions corresponding to an inner EV

For an inner extraordinary vertex of valence N, also in this construction we have attached to it $N+3$ different basis functions; this because the degrees of freedom of system (4.2) to (4.8) around an EV are the same as system (2.4) to (2.10). The computation of the basis functions attached to an inner EV defined on the knot vector $\mathbf{t}$ are obtained via knot insertion from the functions presented in Section 3.2.1.1. Let $B_k^{EV}, k = 1, \dots, N+3$ be any of these basis functions (defined on $\mathbf{t}_{\mathrm{B}}$); if $\mathbf{t}$ (as in (4.16)) is our target knot vector, we can repeatedly apply the knot insertion algorithm to all the patches defining the basis function until we reach $\mathbf{t}$. The output function $B_k^{EV^{\mathbf{t}}}$ we get from this procedure presents a support which is bigger than desired; in fact, as we already noticed in the previous two sections, the G^1 constraints across an edge only involve the points along the edge and its above and below layers. In order to reduce the support of the latter function we apply truncation, i.e. we substract to the elevated basis function a linear combination of the C^1 functions living in the inner part of the patch.

This procedure do not affect the G^1 regularity across edges. Figure 4.3 shows graphically the steps of this procedure on a patch.

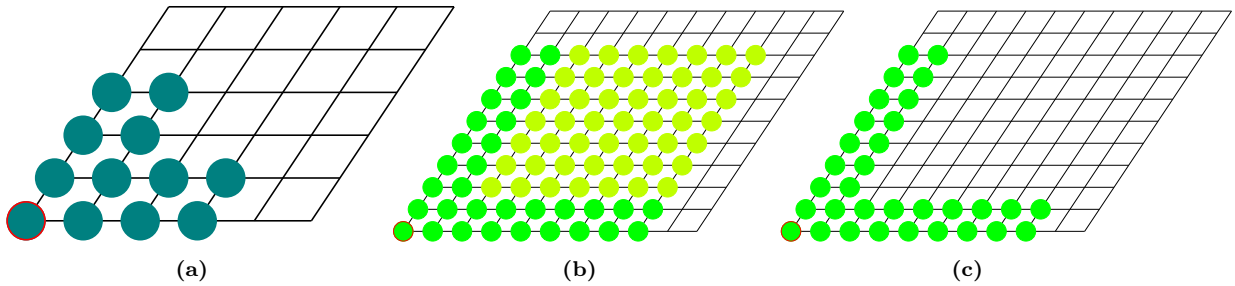


Figure 4.3. Control points involved in the computation of the new vertex basis function. The point surrounded in red refers to the EV. (a): nonzero control points of a basis function defined on $\mathbf{t}_{\mathrm{B}}$. (b): nonzero control points after degree elevation to reach the knot vector $\mathbf{t} = \{\mathbf{0}^6, \mathbf{1/2}^5, \mathbf{1}^6\}$. The light green points refer to the C^1 functions laying on the face. (c): control points defining the new vertex function on $\mathbf{t}$ after truncation.

To conclude, we can summarize the entire procedure as follow: given an EV of valence N and any related vertex function $B_k^{EV}, k = 1, \dots, N+3$ defined on $\mathbf{t}_{\mathrm{B}}$, for suitable sets of indices $\mathcal{I}, \widetilde{\mathcal{I}}$ and $\mathcal{I}^{\mathbf{t}}$, in order to derive the vertex function $B_k^{EV^{\mathbf{t}}}$ defined on $\mathbf{t}$ we do:

$$
\begin{aligned}
B_k^{EV} &= \sum_{i,j \in \mathcal{I}} \mathrm{b}_{i,j} B_{i,j} \\
&\Big\downarrow \text{knot insertion} \\
\widetilde{B}_k^{EV^{\mathbf{t}}} &= \sum_{i,j \in \widetilde{\mathcal{I}}} \widetilde{\mathrm{b}}_{i,j}^{\mathbf{t}} B_{i,j}^{\mathbf{t}} \\
&\Big\downarrow \text{truncation} \\
B_k^{EV^{\mathbf{t}}} = \sum_{i,j \in \mathcal{I}^{\mathbf{t}}} \mathrm{b}_{i,j}^{\mathbf{t}} B_{i,j}^{\mathbf{t}}, \quad &\text{where} \quad \begin{cases} \mathrm{b}_{i,j}^{\mathbf{t}} = 0 & \text{if } \ i,j \geq 2, \\ \mathrm{b}_{i,j}^{\mathbf{t}} = \widetilde{\mathrm{b}}_{i,j}^{\mathbf{t}} & \text{otherwise.} \end{cases}
\end{aligned}
$$

Figure 4.4 presents the set of vertex basis functions for an EV of valence $N = 5$ defined on the knot vector $\mathbf{t} = \{\mathbf{0}^6, \mathbf{1/2}^5, \mathbf{1}^6\}$.

4.2.1.2 Bases linked to boundary EVs

In presence of a boundary EV of valence κ, with κ the number of patches sharing the vertex, similarly to the case of an inner EV we come up with $\kappa + 3$ basis functions. Since $\kappa = N - 1$, it is equivalent to $N + 2$ and we can compute them applying the same procedure of repeated knot insertion as in Section 4.2.1.1.

4.2.1.3 Basis functions at regular vertices and corners

When dealing with a regular vertex, by construction we have attached to them standard C^1 basis functions; same fact for a corner. Both of them can be computed following the strategy in Section 3.2.1.2 and Section 3.2.1.3.

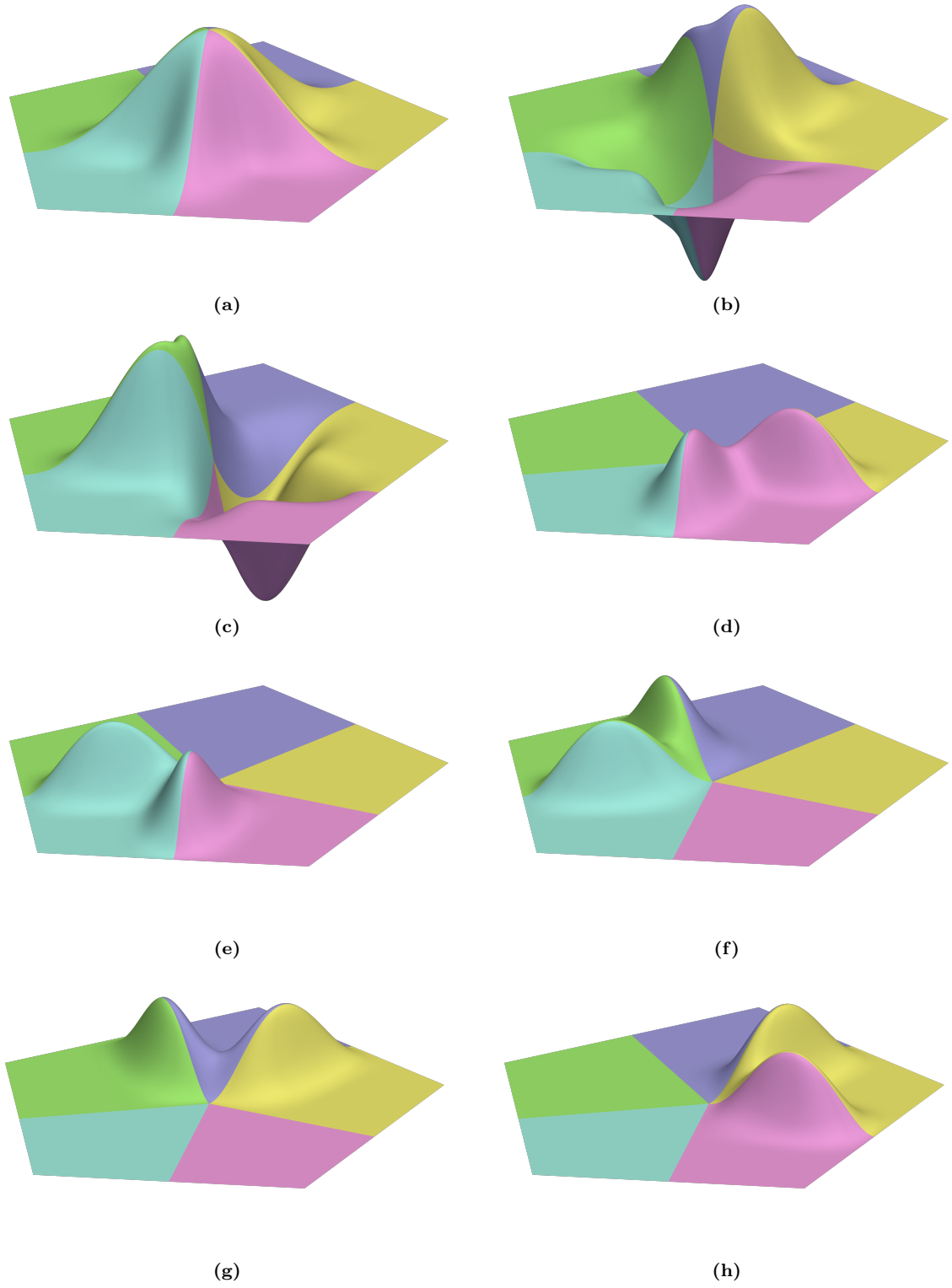


Figure 4.4. Set of truncated vertex basis functions for an EV of valence $N = 5$ defined on the knot vector $\mathbf{t} = \{\mathbf{0}^6, \mathbf{1/2}^5, \mathbf{1}^6\}$. Their non truncated version is shown in Figure 3.2.

4.2.2 Edge basis functions: the set $\mathcal{B}_E^{\mathbf{t}}$

We find in this set those functions whose support lays on the two patches shared by an edge, inner or boundary either extraordinary or regular. Here we present the explicit construction in the case of inner extraordinary since the remaining cases are analogous to Section 3.2.2.2 and Section 3.2.2.3.

4.2.2.1 Extraordinary edge basis functions

For a given knot vector $\mathbf{t}$ as in (4.16) and an extraordinary edge, we will investigate separately the construction of the edge functions which vanish along the edge from those that are nonzero along it. We assume at this step that our knot vector has only one multiple inner knot, i.e.

$$\mathbf{t} = \{\mathbf{0}^6, \mathbf{1/2}^5, \mathbf{1}^6\}. \tag{4.18}$$

The general situation will be treated later. To compute these edge basis functions we exploit the fact that knot vectors as (4.18) define an internal split of the patch in four Bézier subpatches. This split also creates across the edge what we call a virtual EV, i.e. a valence 4 point with no vanishing gluing data on it; hence we can circularly chain around this virtual EV, two patches by two, the relations (4.2) to (4.8), using each time the proper gluing data defined on the subedge identified by the split. In fact, along the extraordinary edge, at each split of the face corresponds a split of the gluing function defined on it: the coefficients identifying the new pair of subdivided (quadratic) gluing data can be computed by using the de Casteljau algorithm (cf. Figure 4.6-(a)-(b)). In addiction, since we want C^1 smoothness in the inner part of the face, when concatenating the equations between subpatches that do not cross the extraordinary edge we use (4.2) to (4.8) with null values for the gluing data coefficients that is in fact imposing C^1 continuity. The labelling for the coefficients of the new pair of gluing data function, as well as the subpatches, is made starting from the right outgoing edge of the virtual EV, in a counterclockwise ordering: we have a_0^R, a_1^R, a_2^R for the outgoing right edge and a_0^L, a_1^L, a_2^L for the outgoing left edge. It is straightforward that

$$a_0^L = -a_0^R, \quad a_0^R \neq 0.$$

As we already explained, these coefficients can be computed applying the de Casteljau algoritmh to the initial gluing data function defined along the original edge. Figure 4.6-(c) illustrates the labeling of the new gluing data and subpatches orientation. Moreover, in each subpatch we order the control point following the Bézier structure in Figure 2.2 starting from the virtual EV which represent the point $\dot{\mathrm{b}}_{0,0}^{(k)}, k = 0, \ldots, 3$, where we introduced the over dot to differentiate the notation from the classical EV case. Figure 4.5 illustrates this labelling.

After building the system, the strategy we exploit to obtain the Bézier coefficients defining the basis functions is analogous to the strategy we used in Section 3.2.1: starting from (4.2) to (4.8) (or (4.9)) we impose, one at the time, the value one to each free coefficient appearing in the G^1 system. Consequently, we solve (4.2) to (4.8) with this initial values while gradually setting the value of any unconstrained coefficients that we encounter to zero.

Since we are dealing with edge functions, we can set to zero the values of the points neighbouring the vertices defining the edge e.g. the points $\dot{\mathrm{b}}_{4,0}^{(0)}, \dot{\mathrm{b}}_{5,0}^{(0)}, \dot{\mathrm{b}}_{4,1}^{(0)}, \dot{\mathrm{b}}_{5,1}^{(0)}, \dot{\mathrm{b}}_{0,4}^{(1)}, \dot{\mathrm{b}}_{0,5}^{(1)}, \dot{\mathrm{b}}_{1,4}^{(1)}, \dot{\mathrm{b}}_{1,5}^{(1)}, \dot{\mathrm{b}}_{4,1}^{(2)}, \dot{\mathrm{b}}_{5,1}^{(2)}, \dot{\mathrm{b}}_{1,4}^{(3)}, \dot{\mathrm{b}}_{1,5}^{(3)}$. The derivation of both types of basis functions is founded on two circulant systems, which come up after the concatenating procedure explained before, whose solutions return

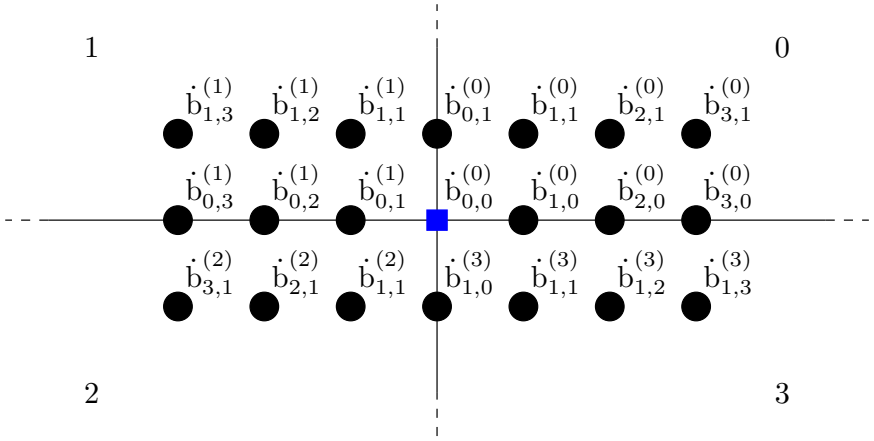


Figure 4.5. Labelling of the control points involved in the construction of the edge basis functions. The ordering is performed counterclockwise, starting from the patch zero, around the virtual EV (blue square).

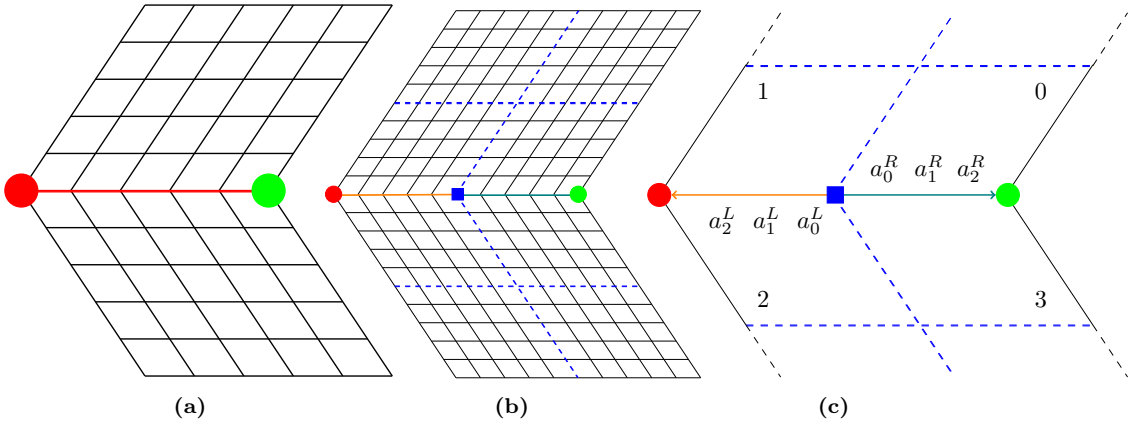


Figure 4.6. Extraordinary vertex (red) and regular vertex (green). (a): extraordinary edge where it is defined a single gluing data function (red line). (b): virtual EV (blue square) which splits in two the extraordinary edge, defining a pair of new gluing functions (orange and green lines). The dashed blue lines identify the regions where we impose C^1 continuity. (c): labelling of the subdivided gluing data and subpatches around a virtual EV.

the points $\dot{\mathbf{b}}_{1,0}$ and $\dot{\mathbf{b}}_{1,1}$ (the bold refers to the vectorial notation introduced in Section 3.2 and (4.9)).

These systems are:

$$\begin{pmatrix} -a_0^R & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & a_0^R & 1 \\ 1 & 0 & 1 & 0 \end{pmatrix} \dot{\mathbf{b}}_{1,0} = \begin{pmatrix} 2 - a_0^R \\ 2 \\ 2 + a_0^R \\ 2 \end{pmatrix} \dot{\mathbf{b}}_{0,0} \tag{4.19}$$

and

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \end{pmatrix} \dot{\mathbf{b}}_{1,1} = \mathbf{d}, \tag{4.20}$$

where

$$\mathbf{d} = \begin{pmatrix} \frac{1}{5}\left(a_0^R - 2a_1^R\right)\dot{\mathrm{b}}_{0,0}^{(0)} + \left(2 - a_0^R + \frac{2}{5}a_1^R\right)\dot{\mathrm{b}}_{1,0}^{(0)} + \frac{4}{5}a_0^R\dot{\mathrm{b}}_{2,0}^{(0)} \\ 2\dot{\mathrm{b}}_{1,0}^{(1)} \\ -\frac{1}{5}\left(a_0^R + 2a_1^L\right)\dot{\mathrm{b}}_{0,0}^{(2)} + \left(2 + a_0^R + \frac{2}{5}a_1^L\right)\dot{\mathrm{b}}_{1,0}^{(2)} - \frac{4}{5}a_0^R\dot{\mathrm{b}}_{2,0}^{(2)} \\ 2\dot{\mathrm{b}}_{1,0}^{(3)} \end{pmatrix}. \tag{4.21}$$

Let analyze their properties. If we call $A \in \mathbb{R}^{4\times 4}$ the matrix in (4.19) it is trivial to notice that

$$\det(A) = 0 \quad \text{and} \quad \operatorname{corank}(A) = 2.$$

In fact, it results that

$$\operatorname{Ker}(A) = \operatorname{Span}\{\dot{\mathbf{k}}_1, \dot{\mathbf{k}}_2\}, \quad \text{where} \quad \dot{\mathbf{k}}_1 = (-1, -a_0^R, 1, 0)^T,\ \dot{\mathbf{k}}_2 = (0, -1, 0, 1)^T$$

since

$$A\,\dot{\mathbf{k}}_1 = \mathbf{0},\ A\,\dot{\mathbf{k}}_2 = \mathbf{0} \quad \text{and} \quad \operatorname{rank}(\dot{\mathbf{k}}_1\,\dot{\mathbf{k}}_2) = 2.$$

Next, if $\widetilde{C}$ is the matrix in (4.20), since we are in the case of even valence $N = 4$, from the analysis in Section 2.3.2.1 we know that

$$\operatorname{corank}(\widetilde{C}) = 1 \quad \text{and} \quad \operatorname{Ker}(\widetilde{C}) = \operatorname{Span}\{\widehat{\mathbf{w}}\}, \quad \widehat{\mathbf{w}} = (1, -1, 1, -1)^T.$$

From the previous investigation we can conclude that in order to obtain a solution, similarly to the constructions presented in Chapter 2 and Chapter 3, we need to add extra conditions to the systems in (4.19) and (4.20); again, these additional constraints will be the orthogonality of the desired solutions to the kernel of the corresponding matrix plus the verification of the Vertex Enclosure Relation (V.E.R.) (2.20). We underline that all the solutions we achieve must be meant in a least square sense, since we will deal with overdetermined systems. Following paragraphs show how to compute explicit solutions for problems (4.19) and (4.20).

Basis functions vanishing along the edge. To obtain basis functions which are zero along the edge we need to impose

$$\dot{\mathrm{b}}_{i,0}^{(0)} = \dot{\mathrm{b}}_{i,0}^{(2)} = 0, \quad i = 0, \dots, 5. \tag{4.22}$$

Specializing (4.9) for a virtual EV, that is valence $N = 4$ and properly subdivided gluing data, and substituting in it the assumption (4.22), we get that

$$\operatorname{corank}(M_{G^1}) = 6,$$

which means there are only six edge functions vanishing along the edge being G^1 across it and C^1 internally the faces for the knot vector $\mathbf{t}$ in (4.18). Four of them are the basis living in the inner part of the two subedges which derive from (4.4) to (4.5) and, alike to Section 3.2.2.1, we have

$$\begin{aligned} \dot{\mathrm{b}}_{2,1}^{(0)} + \dot{\mathrm{b}}_{1,2}^{(3)} &= 0, \\ \dot{\mathrm{b}}_{3,1}^{(0)} + \dot{\mathrm{b}}_{1,3}^{(3)} &= 0, \\ \dot{\mathrm{b}}_{1,2}^{(1)} + \dot{\mathrm{b}}_{2,1}^{(2)} &= 0, \\ \dot{\mathrm{b}}_{1,3}^{(1)} + \dot{\mathrm{b}}_{3,1}^{(2)} &= 0, \end{aligned}$$

that can be solved by setting, for example,

$$\dot{\mathrm{b}}^{(0)}_{2,1} = \dot{\mathrm{b}}^{(0)}_{3,1} = \dot{\mathrm{b}}^{(1)}_{1,2} = \dot{\mathrm{b}}^{(1)}_{1,3} = 1,$$
$$\dot{\mathrm{b}}^{(3)}_{1,2} = \dot{\mathrm{b}}^{(3)}_{1,3} = \dot{\mathrm{b}}^{(2)}_{2,1} = \dot{\mathrm{b}}^{(2)}_{3,1} = -1.$$

Their graph is illustrated in Figure 3.4.

Consequently the remaining two basis functions will be obtained by solving systems (4.19) and (4.20). If we substitute (4.22) in (4.19), it reduces to

$$\begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix} \begin{pmatrix} \dot{\mathrm{b}}^{(1)}_{1,0} \\ \dot{\mathrm{b}}^{(3)}_{1,0} \end{pmatrix} = \mathbf{0}$$

from which we get the two (linearly independent) solutions

$$\dot{\mathrm{b}}^{(1)}_{1,0} = \dot{\mathrm{b}}^{(3)}_{1,0} = 0 \tag{4.23}$$

and

$$\dot{\mathrm{b}}^{(1)}_{1,0} = 1, \quad \dot{\mathrm{b}}^{(3)}_{1,0} = -1. \tag{4.24}$$

To deduce the remaining control points we need to substitute, one at the time, previous solutions in (4.20). First, replacing (4.23) in (4.20) one has

$$\widetilde{C}\dot{\mathbf{b}}_{1,1} = \mathbf{0}$$

whence

$$\dot{\mathbf{b}}_{1,1} \in \mathrm{Ker}(\widetilde{C}) \Longrightarrow \dot{\mathbf{b}}_{1,1} = \widehat{\mathbf{w}},$$

which identifies the nonzero coefficients of the desired basis function. Going on, filling (4.24) in (4.20) we additionally have

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix} \dot{\mathbf{b}}_{1,1} = 2 \begin{pmatrix} 0 \\ -1 \\ 0 \\ 1 \end{pmatrix}, \tag{4.25}$$

where we already imposed the extra condition $\langle \dot{\mathbf{b}}_{1,1} | \widehat{\mathbf{w}} \rangle = 0$ since $\mathrm{rank}(\widetilde{C}) = 3$. Finally, (4.25) returns the solution

$$\dot{\mathbf{b}}_{1,1} = (-1, -1, 1, 1)^T$$

that together with (4.24) defines the other sought function. In Figure 4.7-(a) and (b) are shown these two zero basis functions.

Nonzero edge basis functions. To extract this second set of functions we need to assume that there exists at least one nonvanishing control point along the edge. This assumption reflects an extra degree of freedom on (4.4) and (4.5), which can be solved symmetrically or not; in the light of Section 2.4.1 we opt for a symmetric solution. Under these hypothesis, from (4.9) adapted for a virtual EV, we obtain

$$\mathrm{corank}(M_{G^1}) = 2,$$

meaning that only two basis functions which don't vanish at the edge exist. The computation of the first basis function begins assuming that $\dot{\mathbf{b}}_{0,0} = 1$ (and consequently $\dot{\mathrm{b}}^{(0)}_{3,0} = \dot{\mathrm{b}}^{(2)}_{3,0} = 0$); from (4.8) follows

$$\dot{\mathrm{b}}^{(0)}_{2,0} = -\frac{1}{10} + \frac{1}{2}\dot{\mathrm{b}}^{(0)}_{1,0}, \quad \dot{\mathrm{b}}^{(2)}_{2,0} = -\frac{1}{10} + \frac{1}{2}\dot{\mathrm{b}}^{(2)}_{1,0}. \tag{4.26}$$

The analysis of (4.19) reveals that two extra conditions are required to ensure the uniqueness of its solution; the first additional constrain we impose is the feasibility of the points $\dot{\mathbf{b}}_{1,0}$ for the V.E.R. To do that, we derive the equation of the V.E.R from (4.21) forcing

$$\langle \mathbf{d} | \widehat{\mathbf{w}} \rangle = 0$$

which translates into

$$\left(2 - \frac{3}{5}a_0^R + \frac{2}{5}a_1^R\right)\dot{\mathrm{b}}^{(0)}_{1,0} - 2\dot{\mathrm{b}}^{(1)}_{1,0} + \left(2 + \frac{3}{5}a_0^R + \frac{2}{5}a_1^L\right)\dot{\mathrm{b}}^{(2)}_{1,0} - 2\dot{\mathrm{b}}^{(3)}_{1,0} = \frac{2}{5}\left(a_1^R + a_1^L\right), \tag{4.27}$$

where we used (4.26). Adding (4.27) to (4.19) results

$$\begin{pmatrix} -a_0^R & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & a_0^R & 1 \\ 1 & 0 & 1 & 0 \\ 2 - \frac{3}{5}a_0^R + \frac{2}{5}a_1^R & -2 & 2 + \frac{3}{5}a_0^R + \frac{2}{5}a_1^L & -2 \end{pmatrix} \dot{\mathbf{b}}_{1,0} = \begin{pmatrix} 2 - a_0^R \\ 2 \\ 2 + a_0^R \\ 2 \\ \frac{2}{5}\left(a_1^R + a_1^L\right) \end{pmatrix} \tag{4.28}$$

and, if we name A_V the matrix in (4.28), turns out that

$$\operatorname{corank}(A_V) = 1, \quad \operatorname{Ker}(A_V) = \operatorname{Span}\{\dot{\mathbf{k}}_V\} \quad \text{with} \quad \dot{\mathbf{k}}_V = (0, -1, 0, 1)^T.$$

Finally, the single solution is obtained after setting the extra condition

$$\langle \dot{\mathbf{b}}_{1,0} | \dot{\mathbf{k}}_V \rangle = 0$$

giving rise to

$$\begin{pmatrix} -a_0^R & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & a_0^R & 1 \\ 1 & 0 & 1 & 0 \\ 2 - \frac{3}{5}a_0^R + \frac{2}{5}a_1^R & -2 & 2 + \frac{3}{5}a_0^R + \frac{2}{5}a_1^L & -2 \\ 0 & -1 & 0 & 1 \end{pmatrix} \dot{\mathbf{b}}_{1,0} = \begin{pmatrix} 2 - a_0^R \\ 2 \\ 2 + a_0^R \\ 2 \\ \frac{2}{5}\left(a_1^R + a_1^L\right) \\ 0 \end{pmatrix} \tag{4.29}$$

whose solution is

$$\dot{\mathbf{b}}_{1,0} = (1, 1, 1, 1)^T. \tag{4.30}$$

To obtain the following coefficients defining the nonzero basis function we substitute (4.30) in (4.26) and (4.21); then, adding the extra relation $\langle \dot{\mathbf{b}}_{1,1} | \widehat{\mathbf{w}} \rangle = 0$ to increase the rank of the matrix in (4.20) we get

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix} \dot{\mathbf{b}}_{1,1} = \begin{pmatrix} 2 - \frac{12}{25}a_0^R \\ 2 \\ 2 + \frac{12}{25}a_0^R \\ 2 \\ 0 \end{pmatrix}$$

obtaining the solution

$$\dot{\mathbf{b}}_{1,1} = \begin{pmatrix} 1 - \frac{6}{25}a_0^R \\ 1 + \frac{6}{25}a_0^R \\ 1 + \frac{6}{25}a_0^R \\ 1 - \frac{6}{25}a_0^R \end{pmatrix}.$$

The last coefficients concerning the higher order derivatives are extracted from (4.4) and (4.5), while after substituting in them the previous solutions we got they become

$$\dot{\mathrm{b}}_{2,1}^{(0)} + \dot{\mathrm{b}}_{1,2}^{(3)} = \frac{4}{5}\left(1 - \frac{3}{5}a_1^R\right),$$
$$\dot{\mathrm{b}}_{1,2}^{(1)} + \dot{\mathrm{b}}_{2,1}^{(2)} = \frac{4}{5}\left(1 - \frac{3}{5}a_1^L\right)$$

and

$$\dot{\mathrm{b}}_{3,1}^{(0)} + \dot{\mathrm{b}}_{1,3}^{(3)} = -\frac{6}{25}a_2^R,$$
$$\dot{\mathrm{b}}_{1,3}^{(1)} + \dot{\mathrm{b}}_{3,1}^{(2)} = -\frac{6}{25}a_2^L.$$

To define the second nonzero basis functions we retrace the steps we made for the previous function by setting $\dot{\mathbf{b}}_{0,0} = \dot{\mathrm{b}}_{3,0}^{(2)} = 0$ but $\dot{\mathrm{b}}_{3,0}^{(0)} = 1$; this choice updates the edge constraint (4.8) as

$$\dot{\mathrm{b}}_{2,0}^{(0)} = \frac{1}{2}\dot{\mathrm{b}}_{1,0}^{(0)} + 1, \quad \dot{\mathrm{b}}_{2,0}^{(2)} = \frac{1}{2}\dot{\mathrm{b}}_{1,0}^{(2)}. \tag{4.31}$$

In this framework, the points $\dot{\mathbf{b}}_{1,0}$ needs to satisfy the V.E.R.

$$\left(2 - \frac{3}{5}a_0^R + \frac{2}{5}a_1^R\right)\dot{\mathrm{b}}_{1,0}^{(0)} - 2\dot{\mathrm{b}}_{1,0}^{(1)} + \left(2 + \frac{3}{5}a_0^R + \frac{2}{5}a_1^L\right)\dot{\mathrm{b}}_{1,0}^{(2)} - 2\dot{\mathrm{b}}_{1,0}^{(3)} = -\frac{4}{5}a_0^R. \tag{4.32}$$

Substituting the previous V.E.R. (4.32) in (4.29) and taking into account the initial values we fixed to extract this second basis function, we obtain

$$\begin{pmatrix} -a_0^R & 1 & 0 & 1 \\ 1 & 0 & 1 & 0 \\ 0 & 1 & a_0^R & 1 \\ 1 & 0 & 1 & 0 \\ 2 - \frac{3}{5}a_0^R + \frac{2}{5}a_1^R & -2 & 2 + \frac{3}{5}a_0^R + \frac{2}{5}a_1^L & -2 \\ 0 & -1 & 0 & 1 \end{pmatrix} \dot{\mathbf{b}}_{1,0} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ -\frac{4}{5}a_0^R \\ 0 \end{pmatrix},$$

from which

$$\dot{\mathbf{b}}_{1,0} = h\begin{pmatrix} 2 \\ a_0^R \\ -2 \\ a_0^R \end{pmatrix}, \quad h = \frac{a_0^R}{8a_0^R - a_1^R + a_1^L}, \tag{4.33}$$

where h is well defined under the hypothesis of separated EV in the input mesh. Replacing the solution (4.33) and (4.31) in (4.21), and adding the additional constraint $\langle \dot{\mathbf{b}}_{1,1} | \widehat{\mathbf{w}} \rangle = 0$ as for the previous case, for the points $\dot{\mathbf{b}}_{1,1}$ we get from (4.20) the new system

$$\begin{pmatrix} 1 & 0 & 0 & 1 \\ 1 & 1 & 0 & 0 \\ 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ 1 & -1 & 1 & -1 \end{pmatrix} \dot{\mathbf{b}}_{1,1} = 2h \begin{pmatrix} \frac{2}{5}\left(5 + \frac{13}{2}a_0^R + a_1^L\right) \\ a_0^R \\ -\frac{2}{5}\left(5 + \frac{3}{2}a_0^R + a_1^L\right) \\ a_0^R \\ 0 \end{pmatrix}$$

returning the vector

$$\dot{\mathbf{b}}_{1,1} = \frac{h}{5} \begin{pmatrix} 10 + 13a_0^R + 2a_1^L \\ -(10 + 3a_0^R + 2a_1^L) \\ -(10 + 3a_0^R + 2a_1^L) \\ 10 + 13a_0^R + 2a_1^L \end{pmatrix}.$$

Lastly, the values for the points linked to the higher derivatives can be deduced substituting (4.33), (4.31) and the initial assumption $\dot{\mathrm{b}}_{3,0}^{(0)} = 1$ in (4.4) and (4.5) coming to

$$\begin{aligned} \dot{\mathrm{b}}_{2,1}^{(0)} + \dot{\mathrm{b}}_{1,2}^{(3)} &= \frac{h}{a_0^R}\left((90 - 16a_0^R + 28a_1^R + a_2^R - 2a_1^L)a_0^R + 4\left(a_1^R + \frac{5}{2}\right)\left(a_1^L - a_1^R\right)\right), \\ \dot{\mathrm{b}}_{1,2}^{(1)} + \dot{\mathrm{b}}_{2,1}^{(2)} &= -h\left(10 - 6a_1^L + a_2^L\right) \end{aligned}$$

and

$$\begin{aligned} \dot{\mathrm{b}}_{3,1}^{(0)} + \dot{\mathrm{b}}_{1,3}^{(3)} &= \frac{h}{a_0^R}\left(-2(5 + 16a_0^R - 2a_1^R + a_2^R + 2a_1^L)a_1^R + 2(5 + a_2^R)a_1^L + (80 + 13a_2^R)a_0^R\right), \\ \dot{\mathrm{b}}_{1,3}^{(1)} + \dot{\mathrm{b}}_{3,1}^{(2)} &= 3a_2^L h. \end{aligned}$$

Figure 4.7-(c) and (d) presents these two nonzero edge basis functions.

Extraordinary edge functions for knot vectors with multiple inner knots. The previous paragraph shows the explicit construction of basis functions living across the two patches shared by an extraordinary edge when the knot vector present a single inner knot with multiplicity 5 as (4.18); the entire procedure led to reasonably simple closed formulas defining the control points of the different functions in Bézier form. In principle, the same pipeline can be applied for any knot vector with an arbitrary number of inner knots (4.16) by concatenating the G^1 constraints around every virtual EV appearing plus forcing C^1 smoothness in the inner part of the patch, but unfortunately the results are not obtained as easily as in the previous case. To explain why, let us consider a knot vector of type

$$\mathbf{t} = \{\mathbf{0}^6, \mathbf{1/3}^5, \mathbf{2/3}^5, \mathbf{1}^6\}. \tag{4.34}$$

The case with $s \geq 3$ inner knots follows directly. In presence of two inner knots our patch will be subdivided in 9 Bézier subpatches while the edge will be split in three subedges, giving rise to two virtual EVs; as consequence, also the gluing data defined over the original edge has to

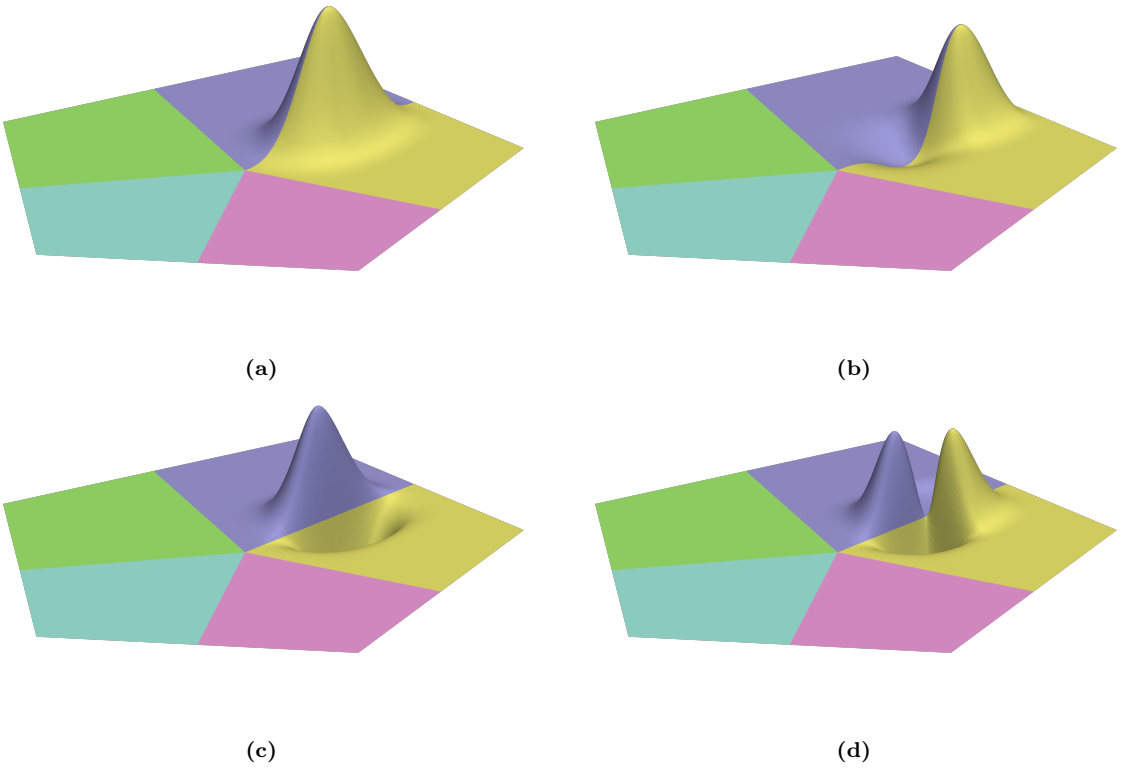
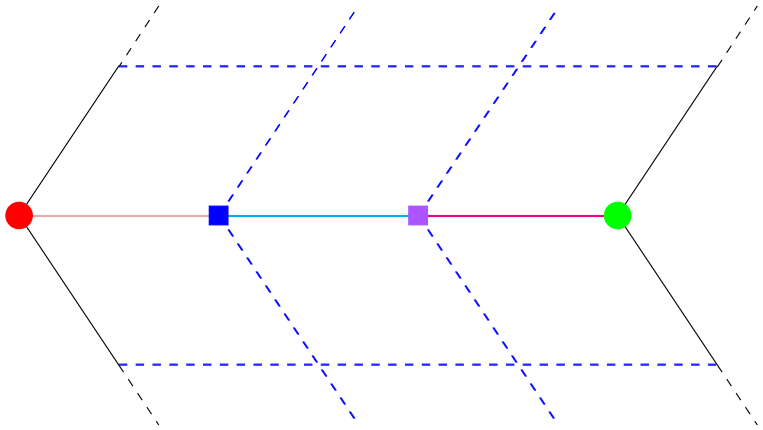


Figure 4.7. Nonzero edge basis functions (a)-(b) and zero edge basis functions (c)-(d) for an extraordinary edge on the knot vector $\mathbf{t} = \{\mathbf{0}^6, \mathbf{1/2}^5, \mathbf{1}^6\}$.

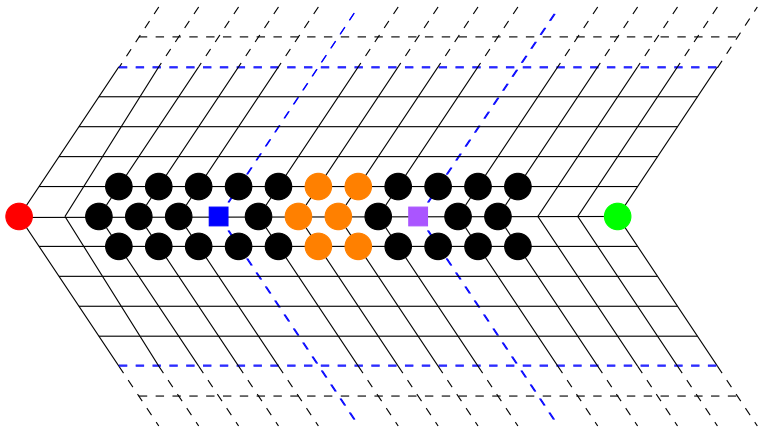
be redefined in each subedge via the de Casteljau algorithm leading to a triple of quadratic gluing data that is globally identified by 9 coefficients. Figure 4.8-(a) underlines with different colors the gluing data functions and virtual EVs living on the splitted edge. If now we chain the G^1 equations (4.2) to (4.8), first around a virtual EV and then around the next one, we realize that the two systems share the pairs of control points referring to the highest order derivatives, i.e. related to (4.4) and (4.5) (cf. Figure 4.8).

That is why, being chained by these points, the two systems must be solved simultaneously and consequently each degree of freedom relative to a virtual vertex we need to fix to begin the computation will inevitability influences the points around the second virtual EV, and vice versa. Hence, the basis functions will have a support which lays all along the control points involved in the 8-times-chained system depending therefore on all the coefficients of the local gluing data functions defined along the edge. Despite all these dependencies (from both the elevated number of involved control points and gluing data coefficients), we are able to derive explicit solutions for the desired basis functions which, however, are not elegant and easy to deal with for implementation purposes. For this reason we opt for a numerical computation of these basis functions by numerically evaluating the kernel of the resulting chained system: this is made, as in the previous explicit construction, separating the computing of basis functions vanishing across the edge from by those that are nonzero on it. Additionally, as expected, computing the corank of the chained system we find that the zero functions along the edge are 10, while the nonzero are 4.

The same chained dependency appears when dealing with knot vectors having an arbitrary



(a)



(b)

Figure 4.8. (a): subdivided gluing data (colored lines) and virtual EVs (blue and violet squares) when considering the knot vector $\mathbf{t} = \{\mathbf{0}^6, \mathbf{1}/\mathbf{3}^5, \mathbf{2}/\mathbf{3}^5, \mathbf{1}^6\}$. (b): control points involved in the G^1 constraints around two virtual EVs. In orange are highlighted the common control points responsible of the concatenating of the two G^1 systems.

number of inner knots: in this case, the pairing of the equations is made concatenating, two-by-two, the G^1 systems of two consecutive virtual EVs as well as the numerical computation. Moreover, if say our knot vector has s inner knots, we will end up with $4s + 2$ zero edge basis functions and $2s$ nonzero edge functions along an extraordinary edge.

4.2.3 Face basis functions: the set $\mathcal{B}_F^{\mathbf{t}}$

The only missing functions in this construction are those whose support only lay on a single patch. The concerned basis functions are C^1 and belong to this set also the ones obtained forcing the C^1 continuity across the split lines defined by the Bézier split. For a knot vector $\mathbf{t}$ with s inner knots, by using combinatorial formulas, we have $(4s + 2)^2$ basis functions in this set.

4.3 Analysis of the basis and space dimension

The functions we provided in previous Section 4.2 define a set of basis functions for the G^1 spline space over the mesh $\mathcal{M}$ and the knot vector $\mathbf{t}$. We denote such space as $\mathbb{G}^1_{\mathbf{t}}(\mathcal{M}) = \langle \mathcal{B}^{\mathbf{t}} \rangle$. The proofs of the following results demonstrating this statement, based on the decomposition in (4.17), are identical to those of Theorem 3.3.1 and Corollary 3.3.2 and for this they will be left out.

Theorem 4.3.1. *The functions introduced in Section 4.2 form a basis $\mathcal{B}^{\mathbf{t}}$ for the space $\mathbb{G}^1_{\mathbf{t}}(\mathcal{M})$ over a quad mesh $\mathcal{M}$.*

Corollary 4.3.2. *The space $\mathbb{G}^1_{\mathbf{t}}(\mathcal{M})$ has dimension given by:*

$$\begin{aligned}\dim\left(\mathbb{G}^1_{\mathbf{t}}(\mathcal{M})\right) &= \sum_{i=1}^{n_V} |\mathcal{B}^{\mathbf{t}}_{V_i}| + \sum_{i=1}^{n_E} |\mathcal{B}^{\mathbf{t}}_{E_i}| + \sum_{i=1}^{n_F} |\mathcal{B}^{\mathbf{t}}_{F_i}| \\ &= \sum_{i=1}^{n_{EV}} N_{EV_i} + 3n_{IEV} + 2n_{BEV} + 4(n_{RV} + n_C) + 4(2s+1)(n_{IRE} + n_{BE}) \\ &+ 2(3s+1)n_{EE} + (4s+2)^2 n_F,\end{aligned}$$

with s the number of inner knots in $\mathbf{t}$, counted without multiplicity.

We observe that, as should be, for $s = 0$ we find again the dimension formula in Corollary 3.3.2.

4.4 Numerical experiments

Here we present some numerical experiment concerning the quality of the functions we propose when dealing with isogeometric analysis simulation. The spaces involved in the approximation process are a series of nested spaces defined over subsequent finer knot vector. Given an initial set of basis functions, the refined ones are obtained inserting $2^L - 1, L \in \mathbb{N}$, uniform inner knots of multiplicity 5 in both parameter directions, where L represent the level of refinement.

Let us consider a mesh $\mathcal{M}$ composed of n_F quad faces σ_ℓ. The space of isogeometric functions we consider is given by

$$\widetilde{V} := \left\{ \widetilde{v} \in L^2(\Omega) \ : \ \widetilde{v} \circ \mathbf{G}^{(\ell)} \in \mathbb{G}^1_{\mathbf{t}}(\mathcal{M})\Big|_{\sigma_\ell}, \ \ell = 1, \dots, n_F \right\}, \tag{4.35}$$

where $\mathbf{G}^{(\ell)}$ is the geometry map defining the G^1ACC spline space introduced in Chapter 2 used to compute the ℓ-th patch of the physical domain

$$\Omega = \bigcup_{\ell=1}^{n_F} \Omega^{(\ell)}, \quad \Omega^{(\ell)} = \mathbf{G}^{(\ell)}(\sigma_\ell). \tag{4.36}$$

We parametrize each mesh's face such that $\sigma_\ell = [0,1]^2$; moreover it results that $\widetilde{V} \subset H^2(\Omega)$. The refined spaces will be denoted $\widetilde{V}_h$, where h is the size of the elements which will be properly specified in each example; generally we have $h = O(2^{-L})$.

First we use the proposed functions for numerical experiments of L^2 projection for both planar and non planar domains and then continue in solving Poisson's equation and biharmonic equation.

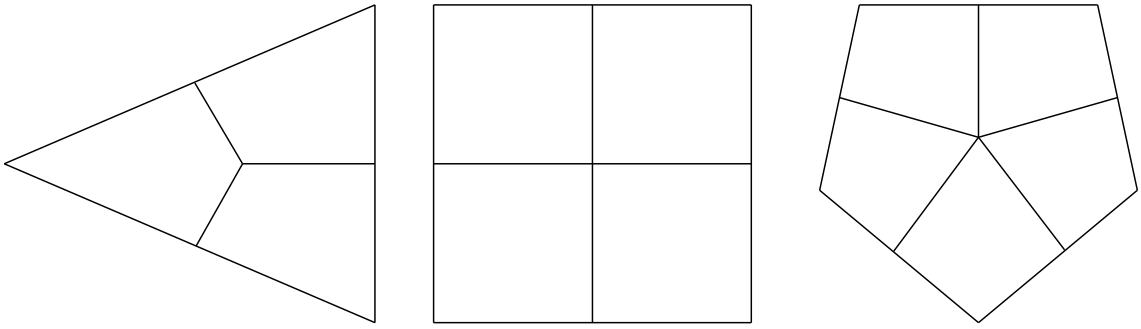


Figure 4.9. Quad meshes presenting EVs of different valences used to define our physical domain Ω involved in the IGA simulations.

4.4.1 L^2 projection

Let Ω be a multipatch domain as in (4.36), $g : \Omega \longrightarrow \mathbb{R}$ a smooth function defined on it and let $\{\phi_i\}_{i\in\mathcal{I}}$, $\mathcal{I} = \{1, \dots, \dim(\widetilde{V}_h)\}$ be a set of G^1-smooth isogeometric functions forming a basis of the subspace $\widetilde{V}_h \subset H^2(\Omega)$. We want to approximate the function g by the function

$$\widetilde{u}_h(\mathbf{x}) = \sum_{i\in\mathcal{I}} c_i\phi_i(\mathbf{x}), \quad c_i \in \mathbb{R}, \tag{4.37}$$

in the least square sense i.e. we compute the coefficients $\{c_i\}_{i\in\mathcal{I}}$ such that

$$\|\widetilde{u}_h - g\|^2_{L^2} = \int_\Omega (\widetilde{u}_h(\mathbf{x}) - g(\mathbf{x}))^2 \; d\mathbf{x} \longrightarrow \min_{c_i}. \tag{4.38}$$

Introducing the mass matrix $M = (m_{i,j})_{i,j\in\mathcal{I}}$ and the vector $\mathbf{g} = (g_i)_{i\in\mathcal{I}}$ whose entries are given by

$$m_{i,j} = \int_\Omega \phi_i(\mathbf{x})\phi_j(\mathbf{x}) \; d\mathbf{x} \quad \text{and} \quad g_i = \int_\Omega g(\mathbf{x})\phi_i(\mathbf{x}) \; d\mathbf{x},$$

the previous minimization problem (4.38) can be reformulated as

$$M\mathbf{c} = \mathbf{g},$$

where $\mathbf{c} = (c_i)_{i\in\mathcal{I}}$ is the vector containing the unknown coefficients.

Example 4.4.1. Here we investigate the L^2 projection problem (4.38) for the function

$$g(x, y) = \cos(2\pi x)\sin(2\pi y) \tag{4.39}$$

over three different domains obtained applying the G^1ACC construction on the meshes in Figure 4.9 with valences $N = 3, 4, 5$. Table 4.1 shows the numerical errors for each step of subdivision while Figure 4.10 represents their plot, indicating that the convergence rate is optimal with respect to the L^2 norm, i.e. $O(h^6)$.

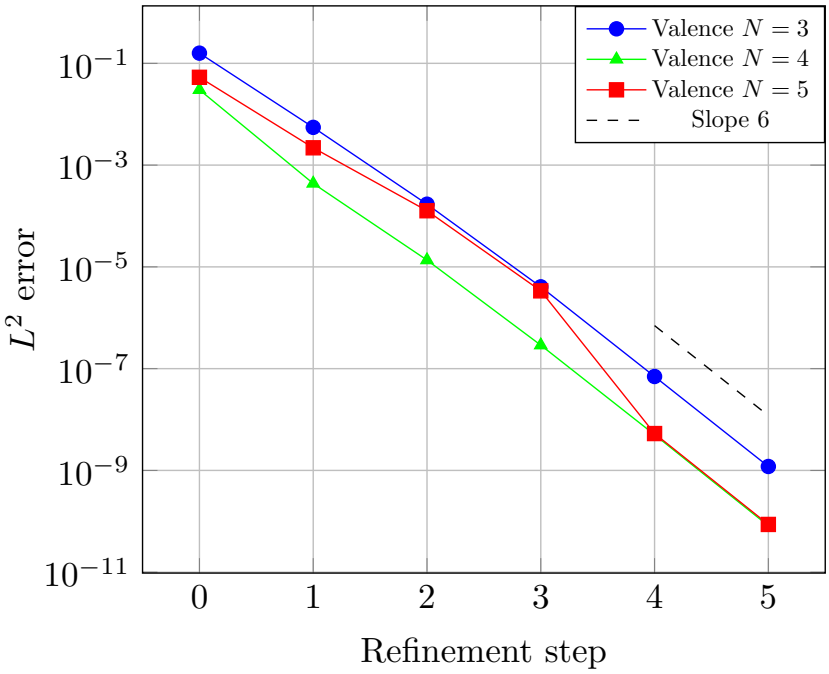


Figure 4.10. Convergence plot of the error for the L^2 projection problem in Example 4.4.1

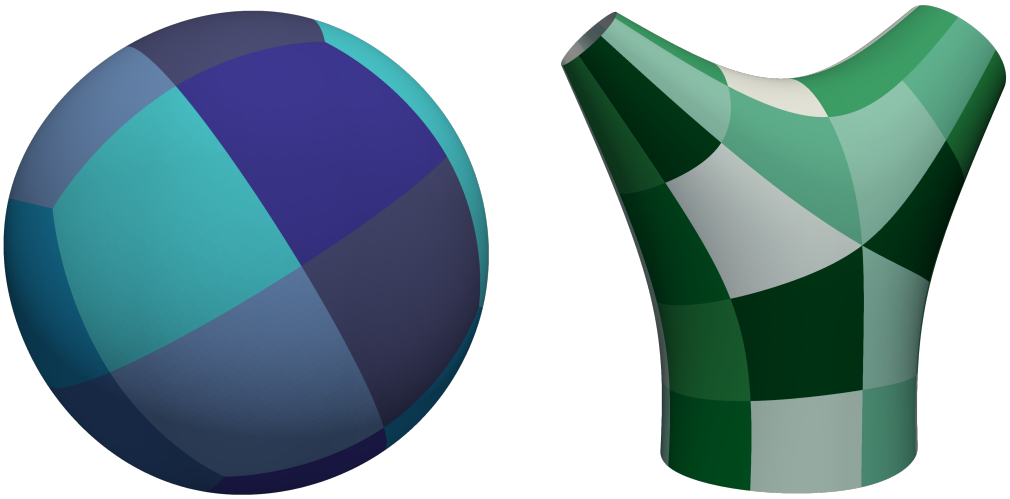
	$N = 3$		$N = 4$		$N = 5$	
L	$\dim(\widetilde{V}_h)$	L^2 error	$\dim(\widetilde{V}_h)$	L^2 error	$\dim(\widetilde{V}_h)$	L^2 error
0	72	1.586e-01	100	3.021e-02	118	5.315e-02
1	234	5.518e-03	324	4.355e-04	388	2.188e-03
2	846	1.705e-04	1156	1.359e-05	1408	1.268e-04
3	3222	4.055e-06	4356	2.892e-07	5368	3.379e-06
4	12582	7.010e-08	16900	5.035e-09	20968	5.300e-09
5	49734	1.195e-09	66564	8.157e-11	82888	8.675e-11

Table 4.1. Spline space dimension and L^2 error for each level of subdivision in Example 4.4.1

Example 4.4.2. The construction of the basis functions in Section 4.2 holds without further adjustments also for nonplanar domains. Here we present again an example of L^2 projection performed on the two multipatch domains in Figure 4.11 containing various EVs of different valences: the objective function we consider in this case is

$$g(x, y, z) = \cos\left(\frac{\pi x}{10}\right) \sin\left(\frac{\pi x}{10}\right).$$

Table 4.2 and Figure 4.12 present the numerical results we gained. We observe that also in this case we achieve the optimal convergence rate $O(h^6)$ for the L^2 error.



(a) sphere **(b)** Y-shape

Figure 4.11. Nonplanar domains utilized in the L^2 projection experiment

	Sphere		**Y − shape**	
L	$\dim(\widetilde{V}_h)$	L^2 **error**	$\dim(\widetilde{V}_h)$	L^2 **error**
0	360	1.008e-06	846	5.347e-06
1	1464	4.769e-08	3222	2.232e-07
2	5976	1.856e-09	12582	8.449e-09
3	24264	3.474e-11	49758	9.808e-11

Table 4.2. Spline space dimension and L^2 error for each subdivision level of the two domains in Example 4.4.2

4.4.2 Poisson's equation

Let consider again a multipatch domain Ω. Here we test our basis functions on the following Poisson's problem

$$\begin{cases} -\Delta u(\mathbf{x}) = f(\mathbf{x}), & \mathbf{x} \in \Omega, \\ u(\mathbf{x}) = h_D(\mathbf{x}), & \mathbf{x} \in \partial\Omega, \end{cases} \tag{4.40}$$

with $f, h_D \in L^2(\Omega)$. If $\{\phi\}_{i\in\mathcal{I}}$, $\mathcal{I} = \{1, \dots, \dim(\widetilde{V}_{h,D})\}$ is a set of basis functions for the subspace $\widetilde{V}_{h,D} \subset H^1(\Omega)$, that is the subspace of functions in (4.35) verifying the Dirichlet boundary condition (4.40), using the Galerkin's approach in Section 1.5.2 we can translate the problem (4.40) in a system of linear equations

$$K\mathbf{c} = \mathbf{f}$$

for the unknown coefficients $\mathbf{c} = (c_i)_{i\in\mathcal{I}}$, where the entries of the stiffness matrix $K = (k_{i,j})_{i,j\in\mathcal{I}}$ and the vector $\mathbf{f} = (f_i)_{i\in I}$ are defined, respectively, as

$$k_{i,j} = \int_\Omega \left(\nabla\phi_i(\mathbf{x})\right)^T \nabla\phi_j(\mathbf{x})\, d\mathbf{x} \quad \text{and} \quad f_i = \int_\Omega f(\mathbf{x})\phi_i(\mathbf{x})\, d\mathbf{x},$$

such that our solution can be written as in (4.37).

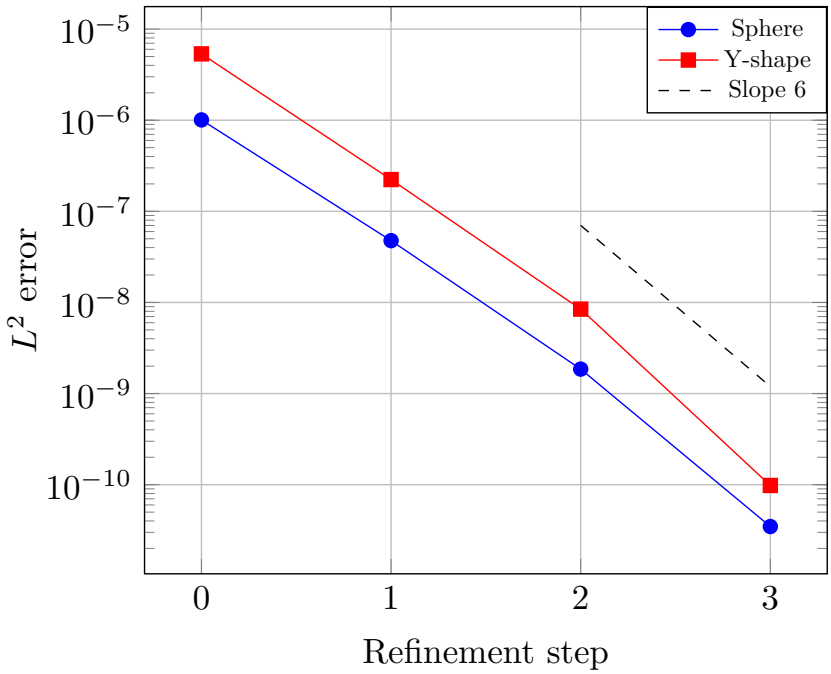


Figure 4.12. Convergence plot of the error for the L^2 projection problem over nonplanar domains in Example 4.4.2

Example 4.4.3. Here we solve the Poisson's problem (4.40) on the three domains defined by the meshes in Figure 4.9. The load function f and the boundary function h_D are selected in order to have as exact solution the function $g(x, y)$ in (4.39). The resulting H^1 errors are presented in Table 4.3 together with their plot in Figure 4.13, indicating that we achieve optimal convergence rate of $O(h^5)$ in H^1 norm.

	$N = 3$		$N = 4$		$N = 5$	
L	$\dim(\widetilde{V}_{h,D})$	H^1 error	$\dim(\widetilde{V}_{h,D})$	H^1 error	$\dim(\widetilde{V}_{h,D})$	H^1 error
0	72	5.757e-01	100	1.283e-01	118	2.035e-01
1	234	3.234e-02	324	3.485e-03	388	1.091e-02
2	846	1.772e-03	1156	1.282e-04	1408	9.592e-04
3	3222	5.826e-05	4356	4.330e-06	5368	4.660e-05
4	12582	1.830e-06	16900	1.370e-07	20968	2.063e-07
5	49734	5.806e-08	66564	4.265e-09	82888	5.257e-09

Table 4.3. Spline space dimension and H^1 error for each level of subdivision in Example 4.4.3

4.4.3 Biharmonic equation

We conclude this chapter presenting the numerical approach to a fourth-order problem, that is the biharmonic equation. Given a multipatch domain Ω, we can introduce the Cauchy problem

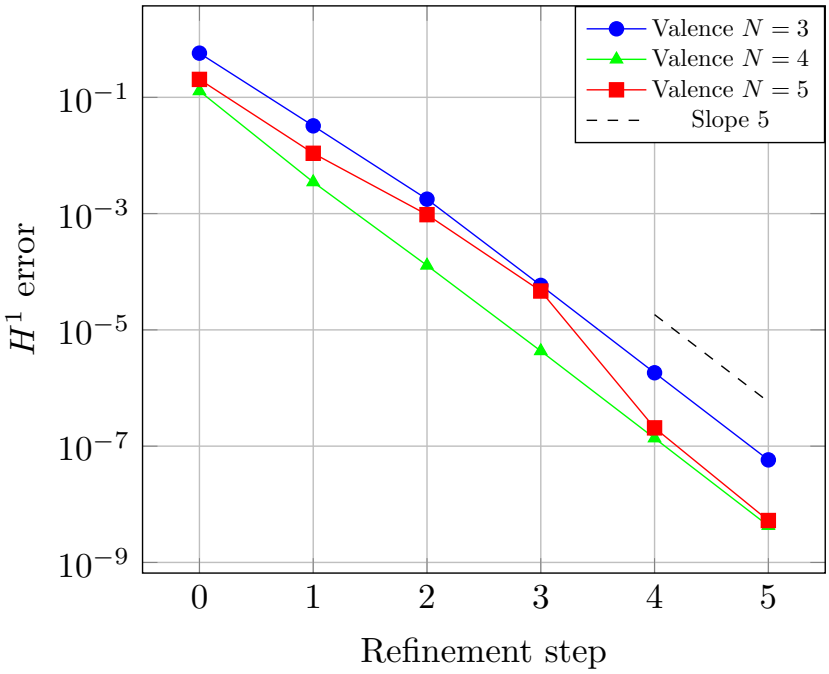


Figure 4.13. Convergence plot of the error for the Poisson's problem in Example 4.4.3

for the biharmonic equation as

$$\begin{cases} \Delta^2 u(\mathbf{x}) = f(\mathbf{x}), & x \in \Omega, \\ u(\mathbf{x}) = h_D(\mathbf{x}), & x \in \partial\Omega, \\ \dfrac{\partial u}{\partial \widehat{\mathbf{n}}} = h_N(\mathbf{x}), & x \in \partial\Omega, \end{cases} \tag{4.41}$$

where f, h_D, h_N are functions belonging to $L^2(\Omega)$. As for the Poisson's problem, using the Galerkin's approach we can translate (4.41) in a linear system. To do that, let $\{\phi_i\}_{i\in\mathcal{I}}$ be a set of basis functions for the subspace of functions in (4.35) verifying the Dirichlet-Neumann conditions (4.41) $\widetilde{V}_{h,D,N} \subset H^2(\Omega)$. Introducing the matrix $K = (k_{i,j})_{i,j\in\mathcal{I}}$ and the vector $\mathbf{f} = (f_i)_{i\in\mathcal{I}}$ whose entries are the quantities

$$k_{i,j} = \int_\Omega \Delta\phi_i(\mathbf{x})\Delta\phi_j(\mathbf{x})\, d\mathbf{x} \quad \text{and} \quad f_i = \int_\Omega f(\mathbf{x})\phi_i(\mathbf{x})\, d\mathbf{x} + \int_{\partial\Omega} h_N(\mathbf{x})\phi_i(\mathbf{x})\, d\gamma,$$

we end up solving the linear system with unknowns $\mathbf{c} = (c_i)_{i\in\mathcal{I}}$,

$$K\mathbf{c} = \mathbf{f},$$

allowing us to write the sought solution in the form of (4.37).

Example 4.4.4. We numerically solve the biharmonic equation (4.41) on the domains described by the meshes in Figure 4.9, defining the input functions f, h_D and h_N such that, similarly to the Poisson's equation, the exact solution is given by the function in (4.39). Table 4.4 resumes the H^2 errors while Figure 4.14 shows their convergence plot; also in this case we gain optimal convergence rate of $O(h^4)$ in H^2 norm.

	$N = 3$		$N = 4$		$N = 5$	
L	$\dim(\widetilde{V}_{h,D,N})$	H^2 error	$\dim(\widetilde{V}_{h,D,N})$	H^2 error	$\dim(\widetilde{V}_{h,D,N})$	H^2 error
0	72	6.184e 01	100	1.428e 01	118	2.081e 01
1	234	8.675e-00	324	1.091e-00	388	2.219e-00
2	846	1.011e-00	1156	6.770e-02	1408	3.216e-01
3	3222	5.705e-02	4356	4.221e-03	5368	3.283e-02
4	12582	3.244e-03	16900	2.637e-04	20968	4.675e-04
5	49734	1.911e-04	66564	1.747e-05	82888	2.374e-05

Table 4.4. Spline space dimension and H^2 error for each level of subdivision in Example 4.4.4

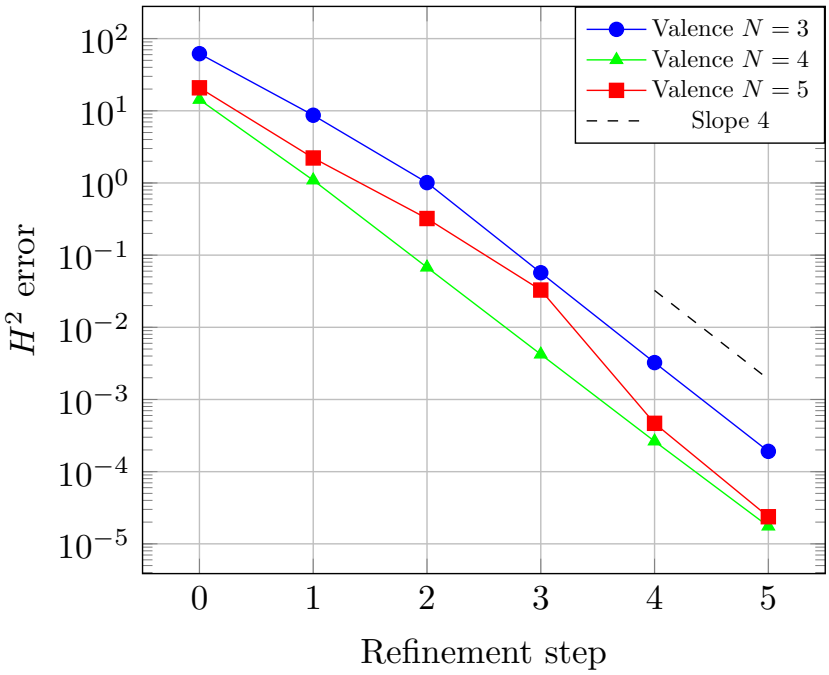


Figure 4.14. Convergence plot of the error for the biharmonic equation in Example 4.4.4

As we already mentioned in Example 4.4.2, the proposed functions can be used to solve PDEs over nonplanar domains; in next Chapter 5 we will show some numerical experiments regarding this topic.

Summary

This chapter has been devoted to the description of a set of analysis-suitable G^1 biquintic spline basis functions. Following the idea of Chapter 3, we proceeded with an extraction strategy starting from the equations describing the G^1 constraints between two adjacent spline patches defined on a knot vector whose knots presents multiplicity 5; this assumption allowed us to perform an easy and almost fully explicit construction for such functions. Several IGA simulations on classical PDEs have been provided to show the optimality of our novel proposal.

Chapter 5

A pipeline from CAD models to spline representation

Shape modelling and analysis are crucial operations which directly impact engineering and industrial processes in many sectors of our society. The last several decades have witnessed the development of many powerful tools for computer-aided design (CAD), computer-aided engineering (CAE), and computer-aided manufacturing (CAM). These tools assist in handling the complex computations required to convert from the digital model of a shape to its actual production. Also boundary representation (B-rep) is a widely used tool when representing manufacturable geometry in Mechanical CAD (MCAD) processes. Such computations can include digital shape description, model reparation, meshing, numerical simulations, and optimisation. Currently, they require specific engineering efforts, are time consuming, and prone to errors and approximations: this explains why alternative approaches are preferred. Moreover, such model are not suitable for numerical simulations.

Using the constructions developed in Chapters 2 to 4 we present here a complete pipeline to convert CAD models into smooth G^1 spline representations, which are suitable for isogeometric analysis. Starting from a CAD boundary representation of a mechanical object, we perform an *automatic* control cage extraction by means of quadrangular faces, such that its limit Catmull-Clark subdivision surface approximates accurately the input model. Then we compute a basis of the G^1 spline space over the quad mesh in order to carry out least squares fitting over a point cloud, acquired by sampling the original CAD geometry. Finally, we use the basis functions to perform isogeometric analysis simulations of realistic PDEs on the reconstructed G^1 model. This chapter is based on [Mar23], and the work has been developed in collaboration with Sam Whyman and Mark Gammon from ITI.

5.1 Control cage generation from MCAD geometry

In this section, we detail the construction of the control cage from the boundary representation of a model. The foundation to our approach is to represent each MCAD edge with a cubic B-spline with a multiplicity-4 knot at each end. This allows the end curvature to be controlled by so-called *slope control points* which are not themselves part of the control cage topology (Figure 5.1). These can equally be thought of as tangent vectors stored at the ends of the spline. The limit surface is then defined to be the tensor product surface of the boundary B-splines. Since the slope control points affect only the first two knot spans, it follows that the first two layers of patches depart from the usual behaviour of the regular regions of a Catmull-Clark subdivision surface (i.e., away from the EVs). This modification constitutes

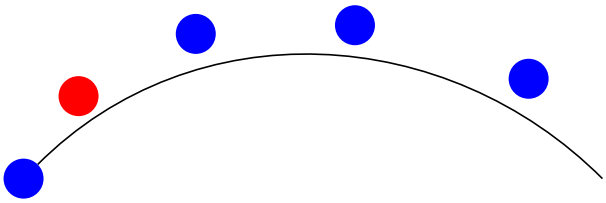
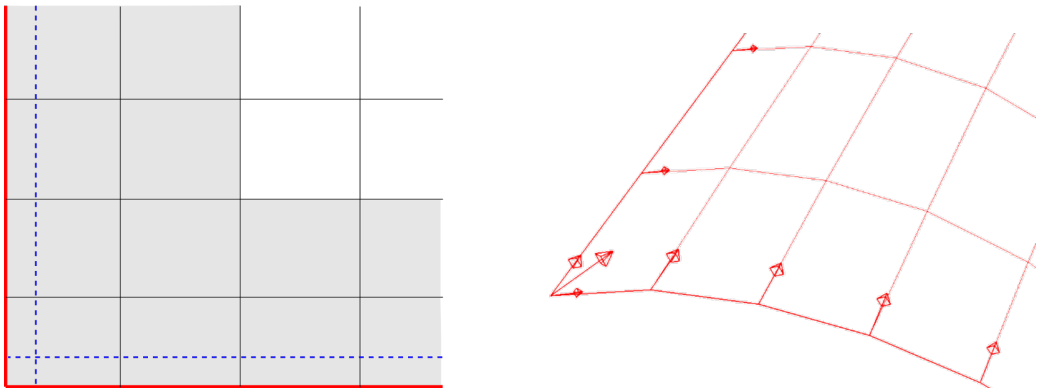


Figure 5.1. Schematic of a cubic B-spline with knot vector $\{0, 0, 0, 0, 1, 2, 3, 4, \dots\}$. The slope control point is shown in red, and all other control points in blue. The location of the red point influences the shape of the spline only within the first two knot spans.

the addition of Bézier edge conditions to the standard Catmull-Clark scheme [She14], and is illustrated by Figure 5.2. The 3D location of the limit surface at the boundary is influenced only by the control points on the boundary. Therefore, neighbouring control cages sharing the same boundary control points will maintain at least C^0 continuity between their limit surfaces, regardless of the positions of the interior control points, or the slope control points.



(a) **(b)**

Figure 5.2. (a): schematic showing the application of Bézier edge conditions to a subdivision surface. The red lines represent the rows of control points which approximate the MCAD edges as cubic B-splines. The blue dashed lines represent the rows of slope control points. Shaded patches are defined using the tensor product of the boundary B-splines, and the unshaded patches are the usual regular bicubic B-spline patches. (b): 3D control cage with vectors pointing to the implied positions of the slope control points. Each boundary control point stores one vector, whereas the corners store three.

The use of Bézier edge conditions imposes strict topological requirements upon the control cage. Namely, no EV may be placed on the boundary, nor within the first two layers of control cage faces, due to the tensor product nature of the limit surface within these regions. These topological constraints, to which engineering-grade subdivision surfaces must adhere, pose challenges when generating a suitable control cage. Each MCAD vertex must be treated as a corner, i.e., it must be associated with a 2-valent control cage vertex (such as in Figure 5.2). The appearance of any EV is restricted solely to the interior of the control cage. Meeting these requirements through traditional quad-dominant meshing techniques is challenging, and therefore we present a novel *automatic* approach referred to as *SubD layering*, which is tailored to satisfying the prescribed topological constraints. The process involves partitioning an MCAD face into regions of structured and unstructured mesh. The structured regions form a *boundary layer* from which EVs are fully excluded. These are formed by constructing

4-sided blocks around each MCAD vertex (referred to as corner blocks), and then connecting these to form 4-sided edge blocks associated with each MCAD edge. The remaining interior region constitutes a block with topology identical to the original MCAD face, and this is filled with unstructured mesh. This process is illustrated in Figure 5.3. The size and positions of

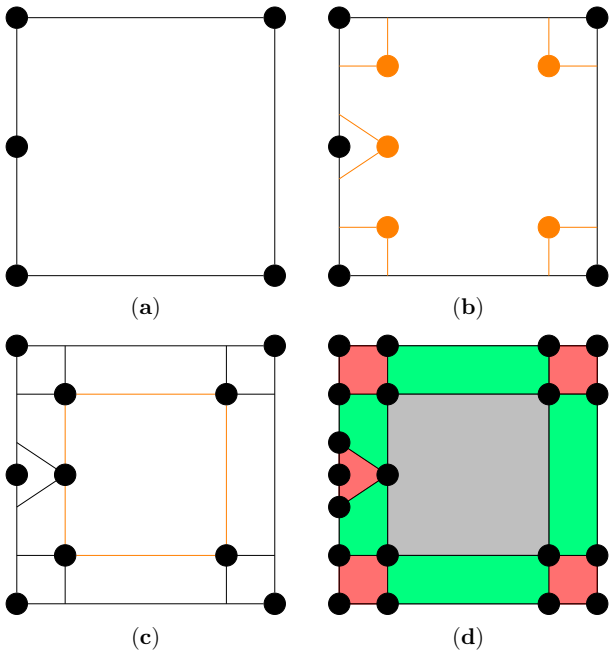


Figure 5.3. Schematic outlining the SubD layering process. (a): MCAD B-rep face with 5 vertices. (b): corner blocks are constructed around the MCAD vertices. (c): corners blocks are connected to form edge blocks. (d): Domain is partitioned into corner (red) and edge (green) blocks which can receive structured mesh, and one interior (grey) region which receives unstructured mesh.

the corner blocks are determined using the two-dimensional medial axis [Ali16] as a guide to ensure that the blocks do not intersect. An example of this is given in Figure 5.4.

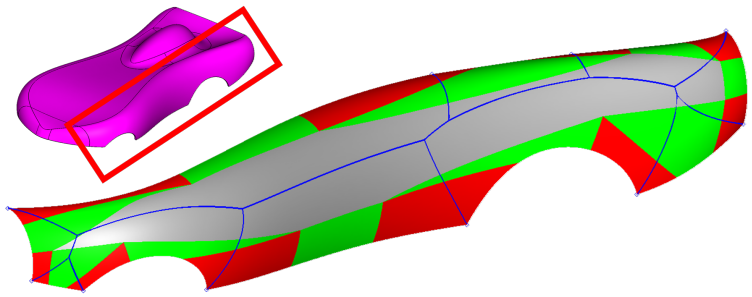


Figure 5.4. SubD layering for the wheel arch of the Car model, showing corner (red), edge (green) and unstructured regions (grey). The 2D medial axis (blue) determines the shape of the blocks so that they do not intersect.

Once the domain partitioning has been completed, we generate a coarse mesh of the layer faces. The boundary layer is meshed using a transfinite interpolation technique [GH73] and is

specified to be one element thick. The interior region is meshed using quad-dominant meshing technology [Ali16]. The coarse mesh is then subdivided twice, and this ensures that there are no EVs within the first two layers of control cage faces from the boundary, thus satisfying the topological requirements for the Bézier edge conditions. This refinement of a coarse mesh is illustrated in Figure 5.5.

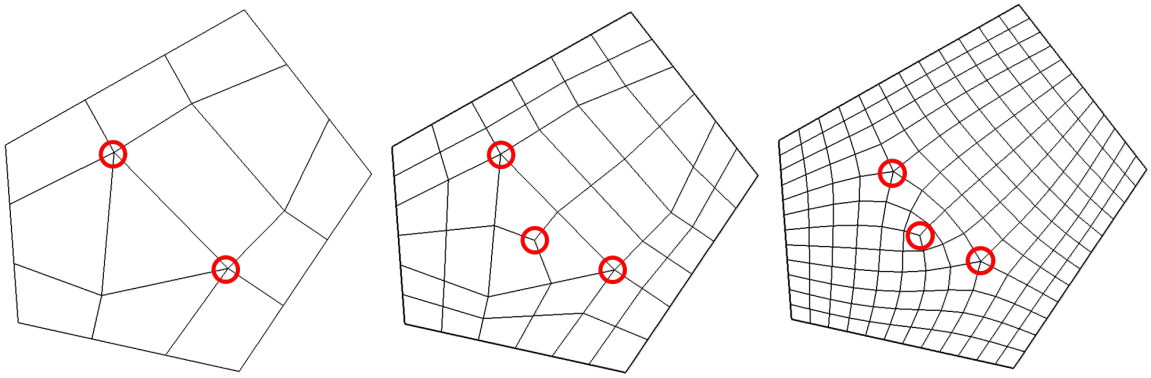


Figure 5.5. (a): initial coarse mesh for a 5-sided face. (b): one level of refinement. (c): two levels of refinement (with mesh smoothing applied). The EVs are highlighted in red. After two levels of refinement the location of the EVs meet the topological requirements for the Bézier edge conditions.

The SubD layering process is applied to each MCAD face in the model in turn, such that the resulting control cage meets the topological requirements dictated by all MCAD edges, both external and internal (i.e., connectivity 1 and 2, respectively). By stipulating that the common vertices along the edges between contiguous faces are shared, a single limit surface is able to represent the entire model while remaining fully watertight (i.e. C^0 continuous). This guarantee is not maintained for MCAD B-rep geometry as a network of NURBS patches. An example of a watertight limit surface is given in Figure 5.6. An example of the control cage

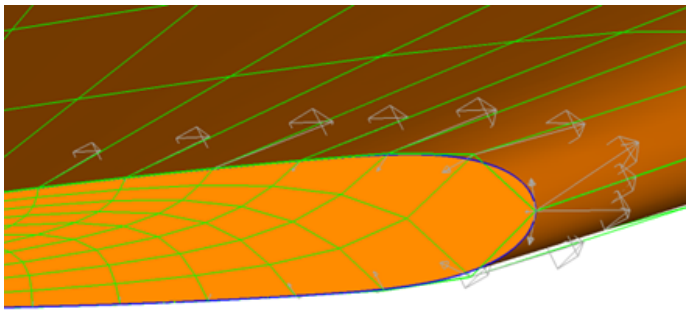


Figure 5.6. A single subdivision surface representing the wing tip of the SNC Dream Chaser model, which is comprised of multiple MCAD B-rep faces. The control cage is shown in green, and the limit surface is shaded in orange. Bézier edge conditions (with slope control vectors shown in grey) are applied such that contiguous regions of the limit surface meet with the same C^1 discontinuity as the original B-rep. However, the limit surface is guaranteed to be exactly C^0 continuous along the join reflecting the MCAD edges (blue).

computation is presented in Figure 5.7. This may be contrasted with the multiple NURBS patch representation shown in Figure 5.8.

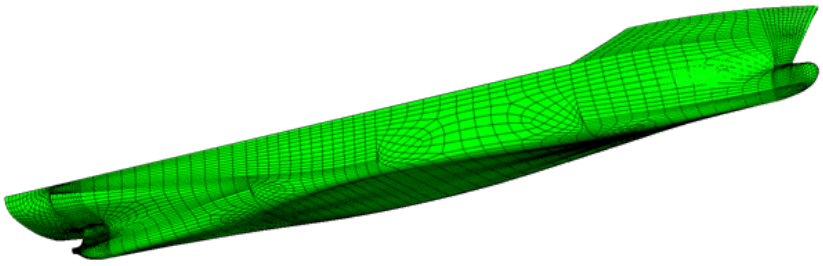
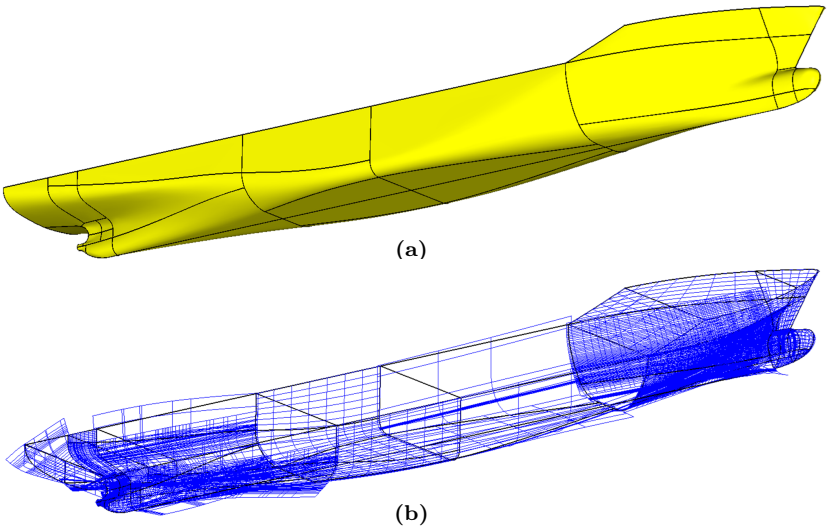


Figure 5.7. Subdivision surface approximation of the KCS ship hull. The control cage is shown in wireframe, and the limit surface shaded.



(a)

(b)

Figure 5.8. (a): shaded MCAD B-rep faces of the KCS ship hull model, with edges shown in black wireframe. (b): embedding NURBS patches for each face.

5.2 Control cage adjustment

The SubD layering process outlined in Section 5.1 is primarily focused on achieving the correct topology for the control cage. The second step of the process is to adjust the control cage so that its limit surface coincides with the target MCAD geometry. To achieve this, we use the ability of our engineering-grade subdivision surfaces to accurately represent geometry; the behaviour of the edges is governed by the B-spline edge conditions, which can be made to respect the MCAD edges via a least squares fitting process. For the interior, we found that 1000 rounds of iterative control cage adjustment give a sufficiently good approximation for each of the four models. This is demonstrated in Figure 5.9, which shows the front section of the NASA CRM.

5.3 Point cloud sampling

The fitting procedure for our multipatch G^1 spline representation combines the generation of points on the geometry with a standard regression between the sample points and the parametric points on the subdivision surface. We require a set of samples points which lie

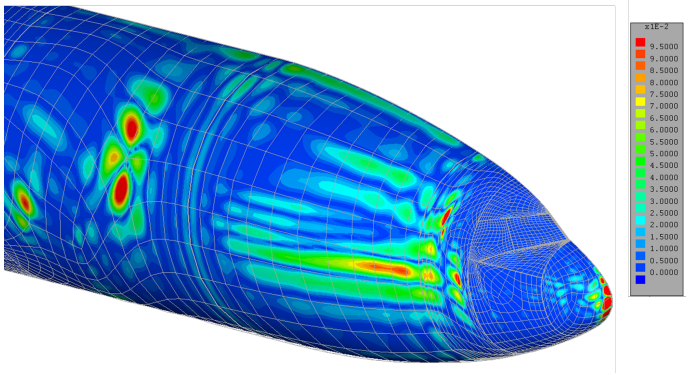


Figure 5.9. Heat map of the distance from the limit surface to the target MCAD geometry, for the forward section of the NASA CRM. A maximum fitting error of approximately 0.005% of the aircraft length is achieved after 1000 rounds of iterative control cage adjustment.

exactly on the geometry, including the edges. Each point must be mapped to a corresponding patch of the subdivision surface, together with a local patch parameter coordinate. This is achieved by sampling each patch of the limit surface at a predetermined set of parameter values, using explicit evaluation [Sta98]. The corresponding 3D positions on the limit surface (which approximates the geometry) are then moved exactly onto the MCAD geometry. Given that the limit surface samples already lie very close to the target geometry, they may be projected easily onto the embedding NURBS surfaces of the MCAD B-rep faces. We also apply additional measures aimed at preventing creases and folds appearing in highly curved regions. The result is a uniform non-folding structured grid of sample points lying on the target geometry, for each patch of the limit surface. Figure 5.10 shows an example of the typical distribution of the point cloud sampling. The procedures introduced here and in previous Section 5.1 have been carried out by using the software CADFIX [CAD].

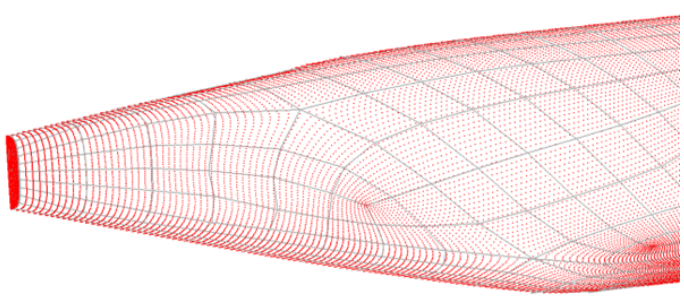


Figure 5.10. An example of the distribution of points in the sampling of the MCAD geometry, for the tail of the NASA CRM. Each patch of the subdivision surface receives a uniform grid of samples, which are projected onto the embedding surfaces of the MCAD B-rep faces.

5.4 From CAD to G^1

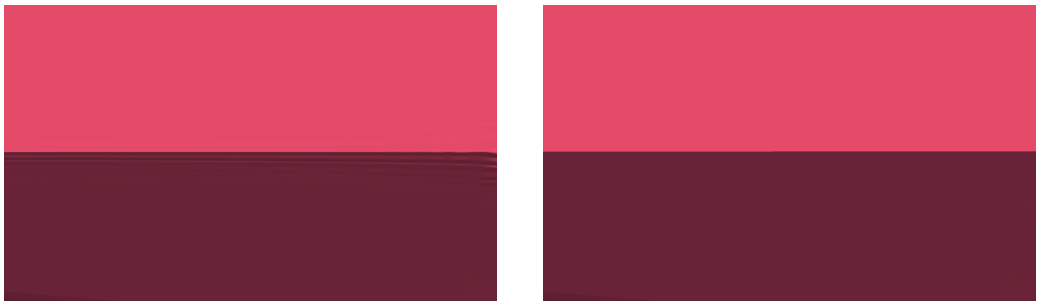
Here we report the numerical experiments concerning the different steps of the proposed construction. First we present the fitting procedure to reconstruct a G^1 surface from an

MCAD point cloud using the technique introduced in Chapter 3, then we will use the previous result as the geometric domain over which to solve the heat equation using geometrically smooth basis functions in the IGA environment. The CAD models used for the numerical investigation are standard target examples for this type of problem. These are: Car model, Dream Chaser shuttle model, KCS hull model and the NASA CRM. All of them present special features and sharp edges which are a notable characteristic to be recovered in the fitted surface. The samplings of the original models are obtained following the procedure explained in Section 5.3. In order to obtain a precise result, the target point clouds contain large amounts of data. For the same reason, the control cage obtained from the MCAD (Section 5.1) presents a significant quantity of faces. The models are represented in their original scale, i.e., 1 unit = 1 m. All of the numerical experiments have been performed on two different machines: the first has been devoted to the control cage generation and the MCAD sampling, while the second ran the basis computation and consequent spline fitting and IGA simulation. Their specifications are: *Windows 10, Intel(R) Xeon(R) CPU E3-1240 v6 @ 3.70GHz, 16.0 GB RAM, 4 cores*, and *Windows 10, Intel(R) Core(TM) CPU i7-9850 @ 2.60GHz, 16.0 GB RAM, 6 cores*, respectively. Figures 5.12 to 5.15 show for each of the four models, in order, the quad mesh extracted from the input MCAD model by using the technique in Section 5.1, the point cloud acquired from the initial geometries following the idea in Section 5.3, the fitted surface in solid and multipatch coloring and a color plot representing the approximation error in Eucledian norm. Moreover, Table 5.1 summarises the dimensions of the starting point cloud, that is the number of points it contains indicated with n_P, the number n_F of faces forming the control cage and the number of basis functions n_b generating the spline space together with the approximation errors evaluated with the formulas in (3.25). From the L^∞ errors represented in Table 5.1, as well as in the color map of the models' color plot, it can be noticed that the highest errors are located, as expected, around the EVs and near the sharp regions of the CAD models. This is because we are fitting sharp edges with high smoothness basis functions which cannot properly recreate the actual shape of the model in these regions (see Figure 5.11). In order to increase the quality of the fitting (i.e., decrease the error) and faithfully reproduce the characteristics of the input model, our construction allows us to identify the sharp edges of the CAD model which are to be preserved: this can be done by labelling as sharp the concerned edges and defining locally on those edges only C^0 basis functions. In this manner our output surface will manifest the desired sharp features. Figure 5.11 shows a detail of the KCS hull model where the top edge is first computed with G^1 smoothness, and then recomputed as a C^0 sharp feature.

Regarding IGA simulations, various experiments with the heat equation have been performed: considering the following Cauchy problem

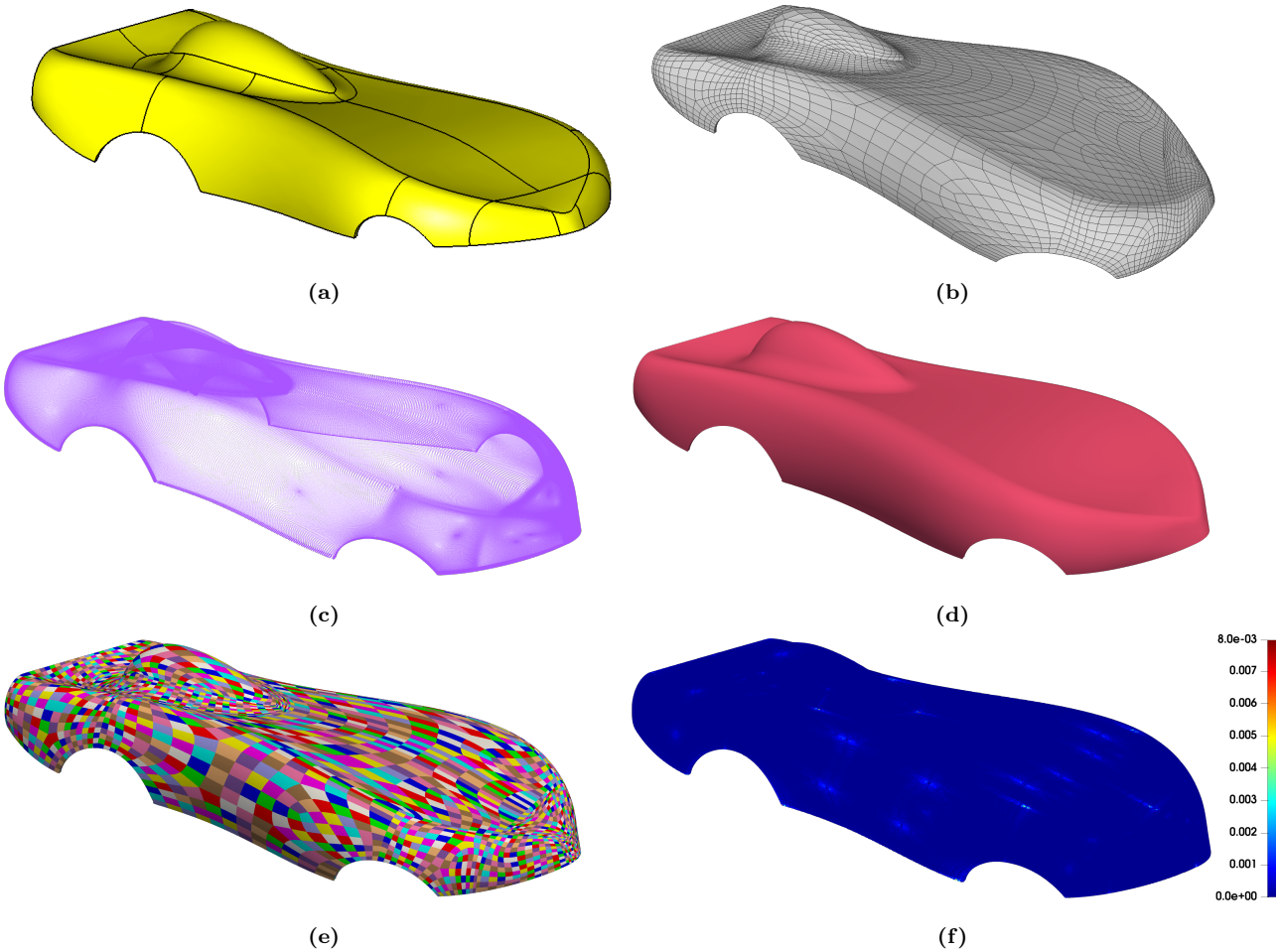
$$\begin{cases} \dfrac{\partial u}{\partial t}(\mathbf{x},t) = c^2 \Delta_\Omega u(\mathbf{x},t)\,, & (\mathbf{x},t) \in \Omega \times (0,T]\,, \\ u(\mathbf{x},0) = u_0(\mathbf{x})\,, & \mathbf{x} \in \Omega\,, \\ u(\mathbf{x},t) = u_D(\mathbf{x},t)\,, & (\mathbf{x},t) \in \partial\Omega \times (0,T]\,, \end{cases} \tag{5.1}$$

with Ω the physical domain we are looking at, Δ_Ω the Laplace-Beltrami operator defined in (1.23), $c, T > 0$ and $u_0(\mathbf{x}), u_D(\mathbf{x},t)$ given initial data, we run the simulations for each of the four CAD models, considering as the final time the instances $T = 0.1$, $T = 0.2$, $T = 0.3$ and $T = 0.4$ minutes. The Galerkin's discretization of (5.1) can be obtained in a similar way as (4.40). These time dependent integrations have been carried out using 20 time steps



(a) (b)

Figure 5.11. Zoom-in of the top edge of the KCS hull model. (a): smoothed edge with consequent oscillations due to the G^1 continuity. (b): sharp edge achieved by imposing C^0 regularity along the sharp edge.



(a) (b) (c) (d) (e) (f)

Figure 5.12. Car model. (a): original MCAD model. (b): quad mesh extrapolated from the MCAD geometry. (c): point cloud sampling of the original MCAD model. (d): surface in solid color. (e): surface in multipatch color. (f): error color plot representing the ℓ_2 distance between the point cloud and the resulting surface.

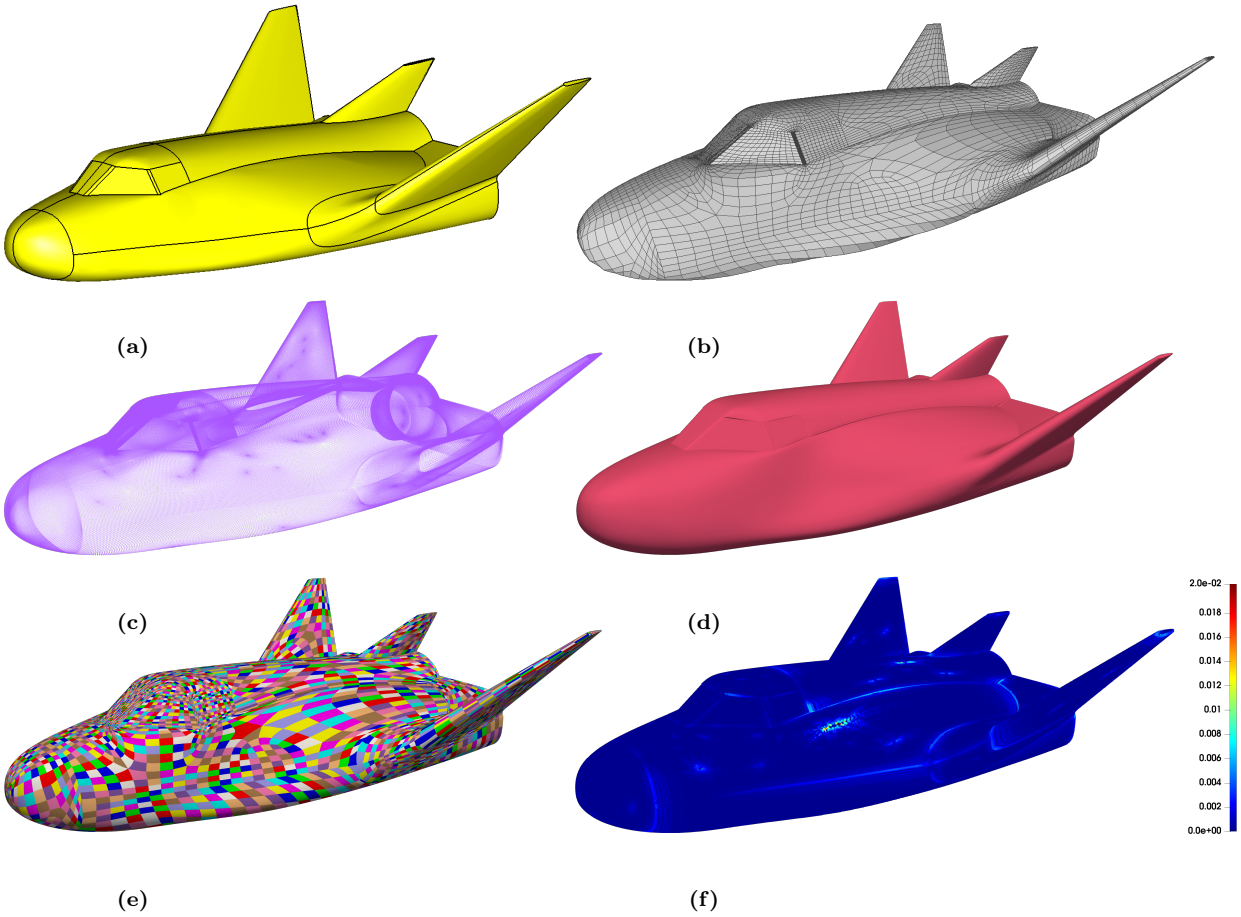


Figure 5.13. Dream Chaser shuttle model. (a): input MCAD model. (b): quad mesh extrapolated from the MCAD geometry. (c): point cloud sampling of the original MCAD model. (d): surface in solid color. (e): surface in multipatch color. (f): error color plot representing the ℓ_2 distance between the point cloud and the resulting surface.

t_{step} under the Crank-Nicolson method. Figures 5.16 to 5.19 present the results of the IGA simulations. These are obtained by setting, as initial conditions, an heat source at the likely location of the engines within the various vehicles represented. This demonstrates a realistic analysis of the thermal behaviour of such models. Table 5.2 presents the running times required to compute each fragment of the pipeline.

	Car	**Dream Chaser**	**KCS**	**NASA CRM**
n_b	166198	168187	166881	116607
n_F	10432	10576	10456	7336
n_P	1262272	1279696	1265179	887656
L^∞ **error**	7.965e-03	4.610e-02	4.644e-01	2.946e-03
RMSE	1.292e-04	1.660e-03	5.890e-03	8.264e-05
Maximal length	3	9	230	1.70

Table 5.1. Fitting errors, spline space and CAD model features for the experiments in Figures 5.12 to 5.15.

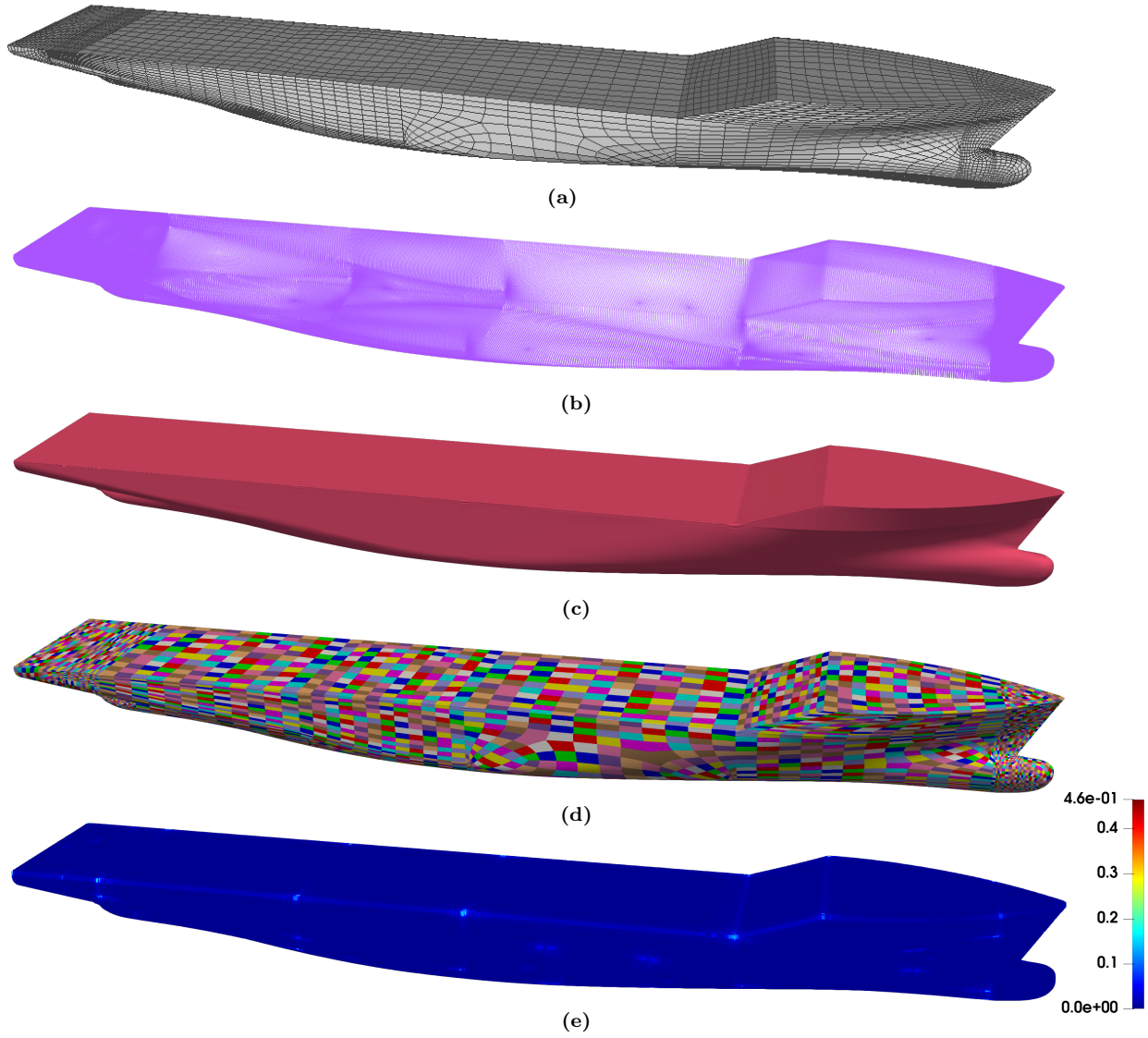


Figure 5.14. KCS hull model. (a): quad mesh extrapolated from the MCAD geometry. (b): point cloud sampling of the original MCAD model. (c): surface in solid color. (d): surface in multipatch color. (e): error color plot representing the ℓ_2 distance between the point cloud and the resulting surface.

	Car	Dream Chaser	KCS	NASA CRM
Cage generation	1 min 45 sec	1 min 4 sec	55.25 sec	57.88 sec
MCAD sampling	22 min 28 sec	26 min 24 sec	22 min 23 sec	22 min 53 sec
Basis computation	11.39 sec	10.98 sec	10.04 sec	6.03 sec
Fitting	2 min 43 sec	3 min 11 sec	2 min 51 sec	1 min 42 sec
IGA simulation	15.65 sec/t_{step}	9.39 sec/t_{step}	7.39 sec/t_{step}	1 min 17 sec/t_{step}

Table 5.2. Detailed elapsed time for each step of the procedure.

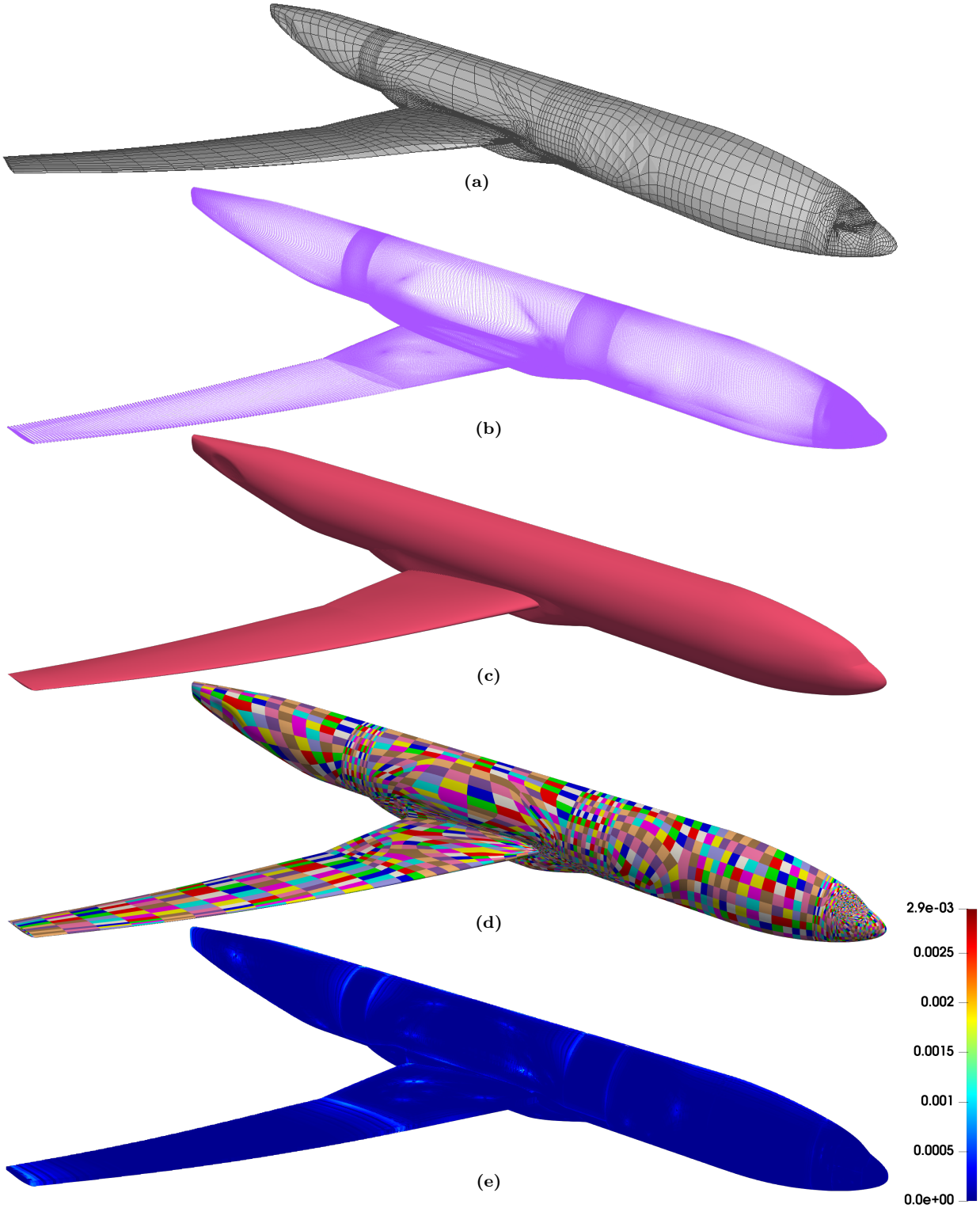


Figure 5.15. NASA CRM. (a): quad mesh extrapolated from the MCAD geometry. (b): point cloud sampling of the original MCAD model. (c): surface in solid color. (d): surface in multipatch color. (e): error color plot representing the ℓ_2 distance between the point cloud and the resulting surface.

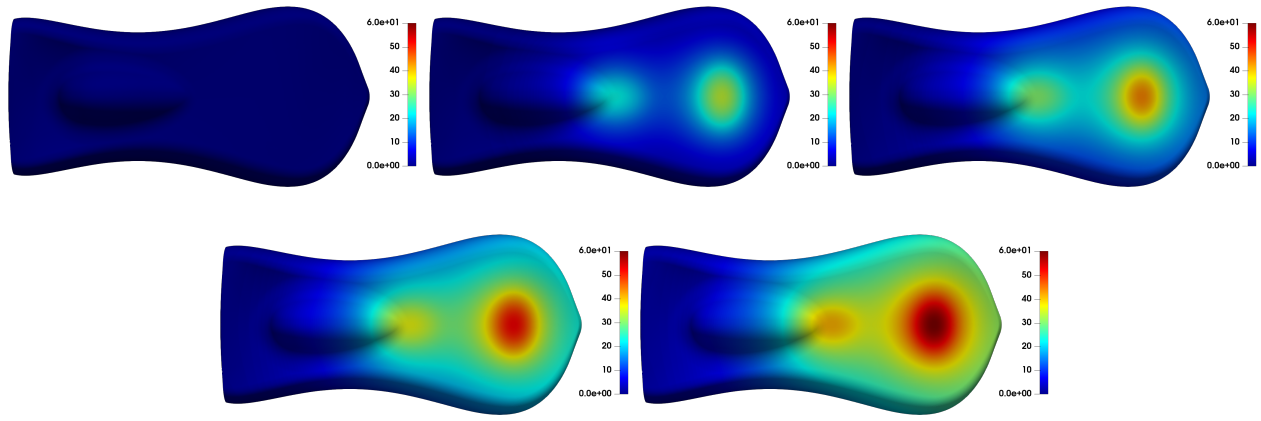


Figure 5.16. Solution for the heat equation on the Car model at the instants from $T = 0$ to $T = 0.4$ min.

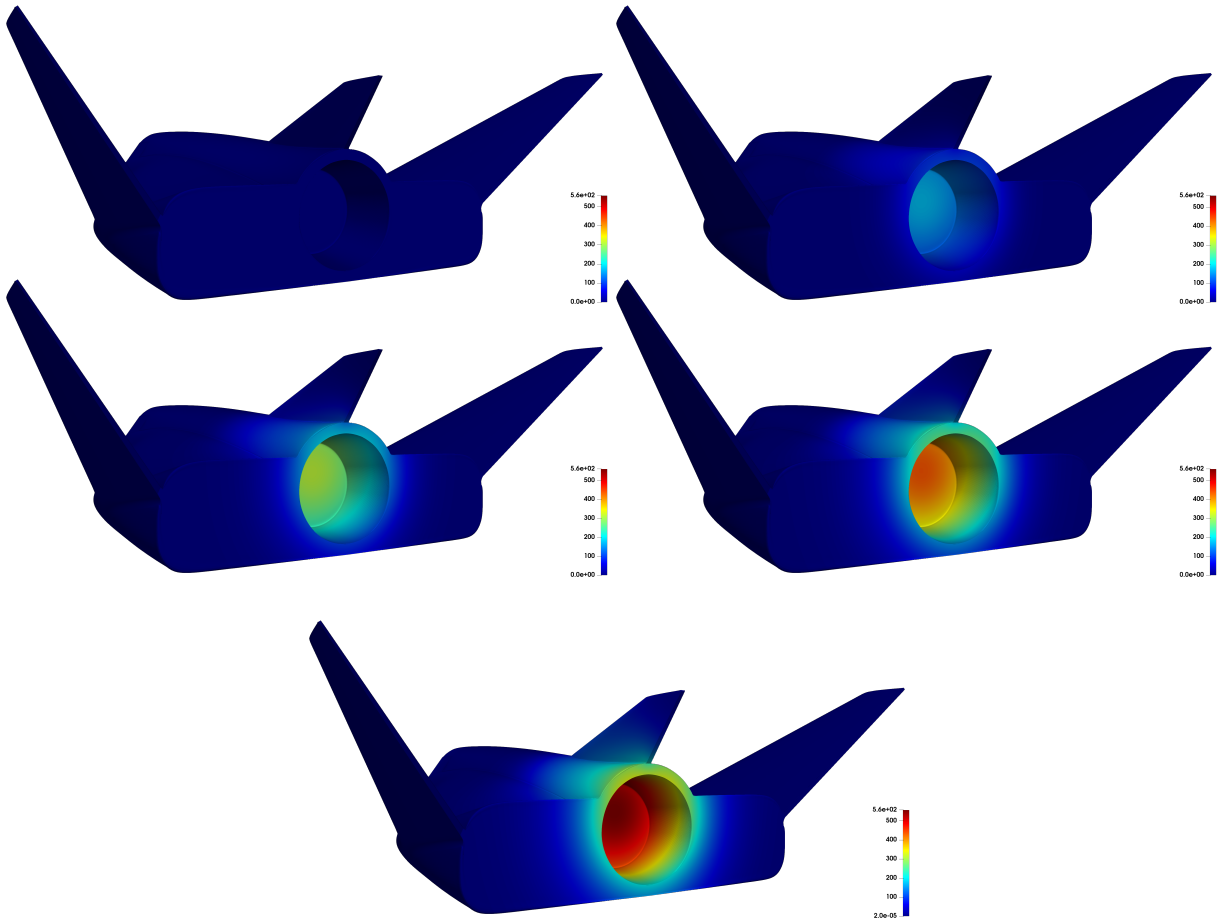


Figure 5.17. Solution for the heat equation on the Dream Chaser model at the instants from $T = 0$ to $T = 0.4$ min.

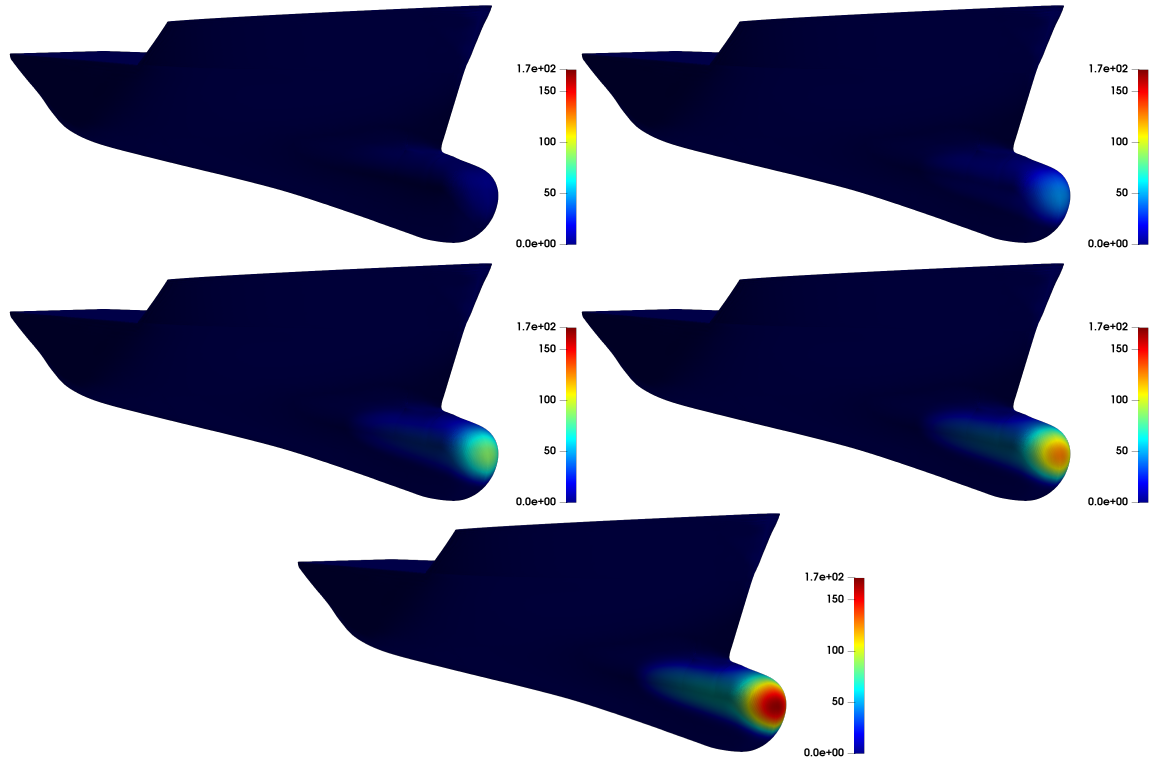


Figure 5.18. Solution for the heat equation on the KCS ship model at the instants from $T = 0$ to $T = 0.4$ min.

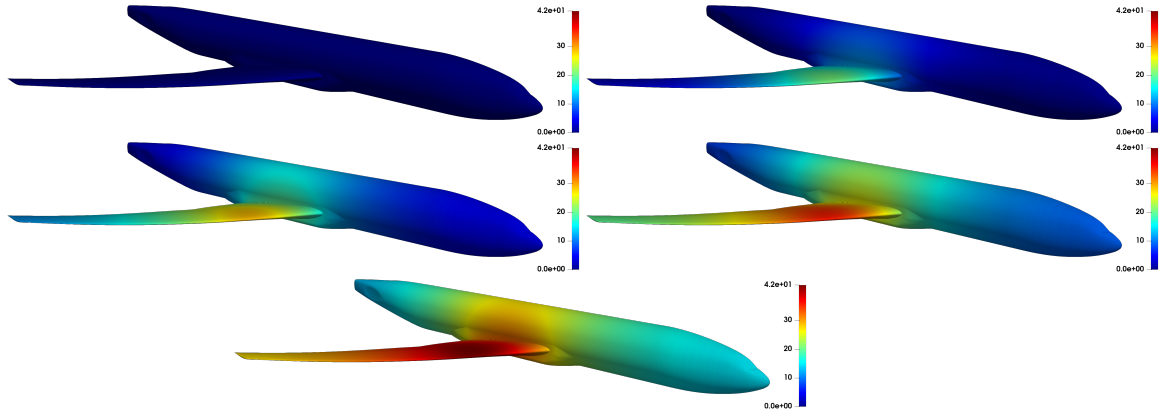


Figure 5.19. Solution for the heat equation on the NASA CRM at the instants from $T = 0$ to $T = 0.4$ min.

Summary

In this chapter we provided a practical application of the constructions described in Chapters 2 to 4 to convert CAD models into G^1 objects. Starting from an input CAD model, using tools from the software CADfix have been possible to build a quad mesh such that its limit Catmull-Clark surface faithfully approximates the CAD model and also a point cloud

discretization of the model itself. With this information, we were able to define the basis functions we previously developed on the given quad mesh and we used them to fit the cloud and obtained a smooth and analysis-suitable model on which we run IGA simulations.

Chapter 6

Free natural vibrations of a shallow lake

In this chapter we present another application of the novel constructions described in Chapters 2 to 4 concerning the IGA simulation of shallow-water equation for real case studies on lakes. Shallow-water equation is an important and popular tool to investigate the behaviour of water surfaces, i.e. their oscillations, in the shallow framework; in particular it is widely used to analyze the surface movement of rivers and lakes nearby their shores as well as coastal tsunami. As can be noticed from its formulation (1.17), the solution of the shallow-water equation strongly depends of the bathymetry function h which describes seabed's shape of the considered environment: for this reason, in order to have an accurate solution, having a precise representation of the bathymetry function is essential.

The focus of our investigation is on the numerical simulation of the so-called *free vibrations* of a shallow lake, that are the oscillations of a lake's surface due only to the force of gravity. Since there are no analytic solutions for the general shallow-water equation, we first make sure the code is correct by analyzing a particular case where an explicit solution is available; then we move further to the general setting considering domains representing real lakes whose bathymetry function is given in terms of point cloud obtained from a GPS rastering. The steps to achieve a truthful solution are the following: starting from cloud data returning the bathymetry of a selected lake, we extract from it the curve delineating the shape of the lake. Then, using Rhino3D modeler we reconstruct a quad mesh (that discretize the lake's surface) on which we build our spline space using the bases in Chapter 3; these basis functions will be used both to reconstruct the bathymetry function by fitting the input point cloud and to solve the shallow-water equation. This is the reason why, in order to obtain a very precise solution, we are going to use fine quad meshes for both the fitting and the simulation.

6.1 Problem statement

Let Ω be our physical domain identifying the surface of a lake living on the plane $\mathbf{x} = (x, z)$ and $h(\mathbf{x})$ the bathymetry function returning the bathymetry value $y = -h(\mathbf{x})$. The Cauchy problem for the shallow-water equation can be formulated as

$$\begin{cases} h(\mathbf{x})\Delta u(\mathbf{x}) + \nabla h(\mathbf{x}) \cdot \nabla u(\mathbf{x}) + \kappa^2 u(\mathbf{x}) = 0, & \mathbf{x} \in \Omega, \\ \dfrac{\partial u}{\partial \widehat{\mathbf{n}}}(\mathbf{x}) = 0, & \mathbf{x} \in \partial\Omega, \end{cases} \tag{6.1}$$

where $\kappa^2 = \sigma^2/g$, g the gravitational acceleration and σ the frequency of the free oscillations. The homogeneous Neumann condition is a natural choice for this kind of equations. We observe that (6.1) is actually describing an eigenvalue problem: introducing the differential operator $\mathcal{D}_h$, which depends on the function h, defined as

$$\mathcal{D}_h\varphi = h\Delta\varphi + \nabla h \cdot \nabla\varphi, \tag{6.2}$$

(6.1) can be reformulated as

$$\mathcal{D}_h u = -\kappa^2 u,$$

meaning that the solution we seek is an eigenfunction of the operator $\mathcal{D}_h$ with eigenvalue $-\kappa^2$. In general, there is no explicit solution for the problem (6.1) unless special cases such as very regular shape for the domain Ω and constant depth h; in order to test the correctness of our implementation we first focus on the particular case where the lake Ω is a square domain of length ℓ with depth $h = constant$.

6.1.1 The test case

Assuming constant bathymetry, (6.2) reduces to

$$\mathcal{D}_h\varphi = h\Delta\varphi,$$

from which (6.1) becomes

$$\begin{cases} \Delta u(\mathbf{x}) = -\bar{\kappa}^2 u(\mathbf{x}), & \mathbf{x} \in \Omega, \\ \dfrac{\partial u}{\partial \widehat{\mathbf{n}}}(\mathbf{x}) = 0, & \mathbf{x} \in \partial\Omega, \end{cases} \tag{6.3}$$

with $\bar{\kappa}^2 = \kappa^2/h$, that is a well-known equation in acoustic applications named *Helmholtz equation*. In the additional case of a square lake, an exact solution of (6.3) can be simply obtained via separation of variables, leading to:

$$u(x,z) = \Lambda \cos\left(\frac{k\pi x}{\ell}\right)\cos\left(\frac{j\pi z}{\ell}\right), \quad (x,z) \in \Omega, \tag{6.4}$$

with $\Lambda \in \mathbb{R}$ and k, j, ℓ any integers such that

$$\bar{\kappa}^2 = \frac{\pi^2\left(k^2 + j^2\right)}{\ell^2}.$$

Figure 6.1 shows the exact solution (6.4) in presence of a unit square (i.e. $\ell = 1$) for $k = j = 1$, $k = 2, j = 3$ and $\Lambda = 1$. Similarly to the problems in Section 4.4, by using the basis functions introduced in Chapter 4, the Galerkin's discretization of (6.3) brings to

$$K\mathbf{c} = -\bar{\kappa}^2 M, \tag{6.5}$$

where the stiffness $K = (k_{i,j})_{i,j\in\mathcal{I}}$ and mass $M = (m_{i,j})_{i,j\in\mathcal{I}}$ matrices have entries

$$k_{i,j} = \int_\Omega \left(\nabla\phi_i(\mathbf{x})\right)^T \nabla\phi_j(\mathbf{x})\, d\mathbf{x}, \quad m_{i,j} = \int_\Omega \phi_i(\mathbf{x})\phi_j(\mathbf{x})\, d\mathbf{x} \tag{6.6}$$

and $\mathbf{c} = (c_i)_{i\in\mathcal{I}}$ is the vector containing the unknown coefficients of the desired solution expressed as linear combination of the bases $\{\phi_i\}_{i\in\mathcal{I}}$ as in (4.37). Hence, our numerical investigation is devoted to the computation of the eigenvalues of the *generalized eigenvalue* problem defined in (6.5) whose related eigenfunctions are the solution of (6.3).

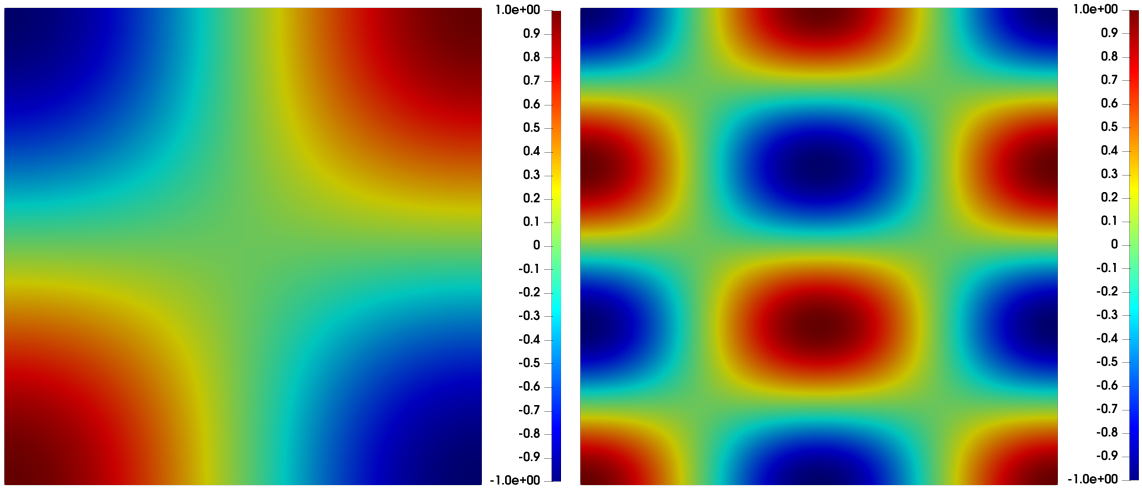


Figure 6.1. Exact solution (6.4) for the problem (6.3) when Ω is the unit square. (a): solution for $\Lambda = k = j = 1$. (b): solution for $\Lambda = 1, k = 2, j = 3$.

6.1.2 The general case

Once we make sure that our implementation is correct, we can deal with the general problem (6.1) defined over a domain Ω with arbitrary shape and any bathymetry function h. As we said above, in this setting the problem (6.1) has no explicit solution. Its Galerkin's formulation, as for the test case, leads to the generalized eigenvalue problem

$$\widetilde{K}\mathbf{c} = -\kappa^2 M,$$

with M as in (6.6) and stiffness matrix $\widetilde{K} = (\widetilde{k}_{i,j})_{i,j\in\mathcal{I}}$

$$\widetilde{k}_{i,j} = \int_\Omega h(\mathbf{x}) \left((\nabla\phi_i(\mathbf{x}))^T \nabla\phi_j(\mathbf{x}) \right) \, d\mathbf{x} + \int_\Omega \phi_j(\mathbf{x}) \left((\nabla h(\mathbf{x}))^T \nabla\phi_i(\mathbf{x}) \right) \, d\mathbf{x}$$

In the next section we present two numerical simulation for the shallow-water equations based on real data: starting from the 3D point cloud of a lake's seabed obtained from a GPS rastering we are able to delineate the shape of the lake (defining our domain Ω) as well as the h function via fitting the cloud using the basis functions in Chapter 3 defined over a quad meshing of Ω.

6.2 Simulations on real lake data

The data used in these experiments is taken from the dataset analyzed in [Kha22]. In particular we use data from the Rogagua Lake, in Bolivia and from the Orta lake, in Italy; the two lakes have been chosen based on their very different shape and bathymetry conformation.

In order to solve the shallow-water problem (6.1) using the technique introduced in Chapter 4 we need to define a mesh domain delineating the shape of the lake as well as a function returning the bathymetry of the lake. This goal can be achieved by following the next steps: first, starting from the point cloud acquired from a GPS rastering of the lake's seabed (represented as triangular mesh), using the RHINO3D modeler command *ExtractMeshEdges* we extract the

boundary edges and we get a curve identifying the shape of the lake. Since the extrapolated curve is obtained from a set of points it is a piecewise linear curve; therefore, to achieve a smoother and more realistic result we apply few iterations of Laplacian smoothing [Sor05]. This curve will identify our physical domain Ω. A triangular meshing of the internal area can be made by the command *PlanarMesh* of RHINO3D, and its conversion to quads by using the *QuadRemesh* option. As accurately replicating the lake's shape is essential for a precise simulation, the resulting mesh will exhibit a high number of faces. Finally, on this mesh we build our spline bases which will be used to both reconstruct the bathymetry function h by fitting the input point cloud and solving the shallow-water equation.

6.2.1 Rogagua lake

The first experiment is performed on the Rogagua lake model from the repository analyzed in [Kha22]. Figure 6.2 illustrates the meshing and fitting steps properly explained in Section 6.2 while Table 6.2 summarizes the features of the input point cloud, i.e. its number of points n_P, the number of quad faces n_F of the extrapolated quad mesh and the number of bases n_b defined on it as well as the errors (defined in (3.25)) while fitting the bathymetry function. It also contains the dimensions of the lake's model bounding box, expressed in meters. The error color plot in Figure 6.2-(e) points-out that the (very few) regions where the fitting error is localized corresponding to the boundary of the domain, and this is due to its slight modification induced by the Laplacian smoothing in order to get a proper outline. Finally, we computed the first 15 solutions for the shallow-water problem (6.1), shown in Figure 6.3, that correspond to the eigenfunctions with associated eigenvalues (and frequencies) listed in Table 6.1.

Eigenvalues		
2.872e-08	2.802e-05	1.601e-04
3.713e-04	5.037e-04	5.087e-04
6.527e-04	8.495e-04	9.009e-04
1.150e-03	1.294e-03	1.489e-03
1.573e-03	1.838e-03	1.878e-03

Frequencies		
5.307e-04	1.658e-02	3.962e-02
6.034e-02	7.028e-02	7.063e-02
8.001e-02	9.128e-02	9.399e-02
1.060e-01	1.127e-01	1.208e-01
1.242e-01	1.343e-01	1.357e-01

Table 6.1. First 15 eigenvalues, and corresponding frequencies, for the shallow-water simulation on Rogagua lake.

6.2.2 Orta lake

The second investigation concerns the Orta lake, whose model belongs to the same dataset as before. Its irregular shape and seabed is very different from the previous Rogagua lake, and together with the presence of an island make this case very interesting. Similarly to the previous experiment, Table 6.2 presents the point cloud features and the attributes of the quad mesh we achieved from it. Also in this case the fitting error is localized nearby the boundary of the domain. Table 6.3 lists the first 15 eigenvalues and frequencies identifying the 15 solutions of the shallow-water equation depicted in Figure 6.5.

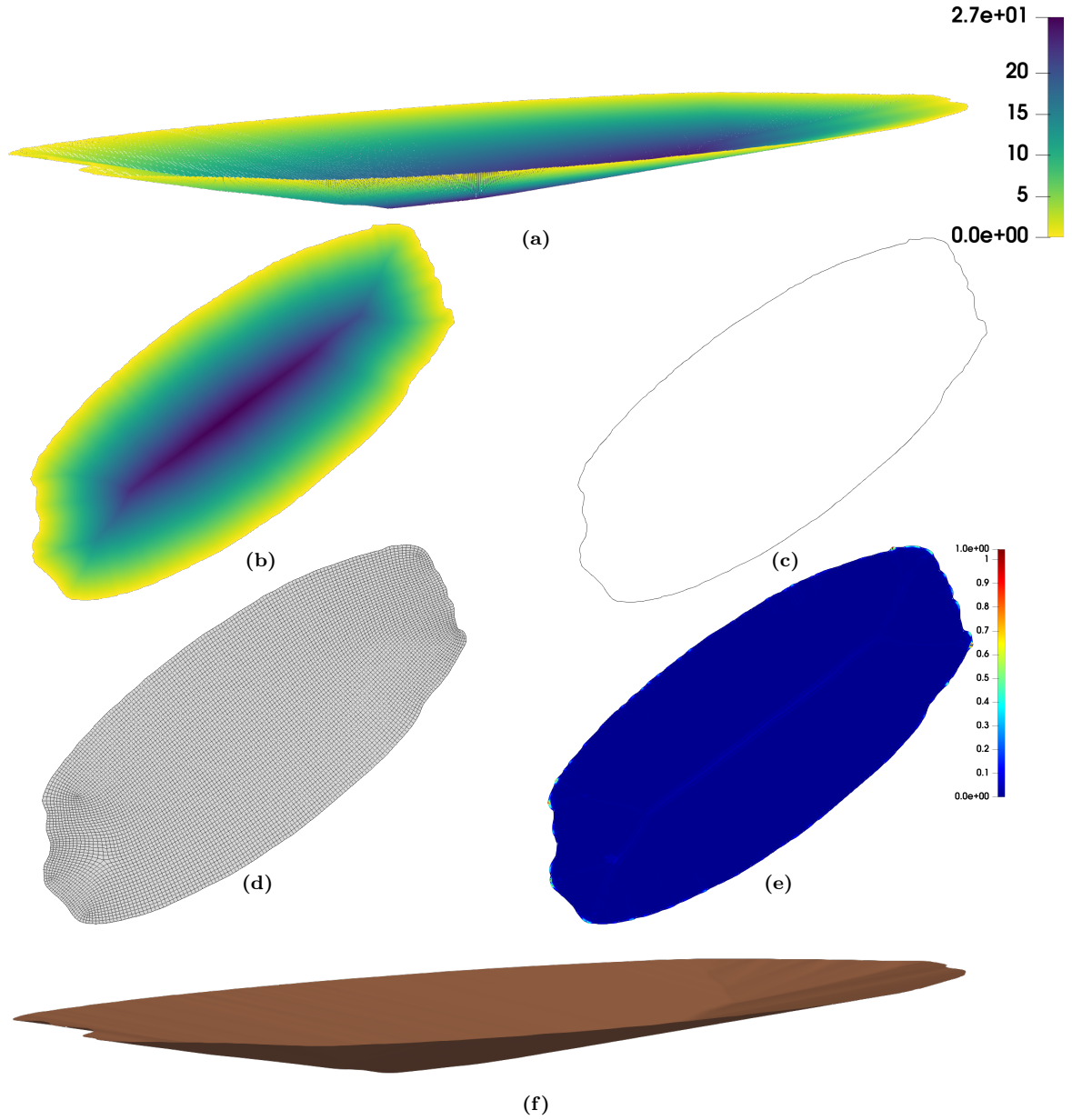


Figure 6.2. Rogagua lake. (a)-(b): point cloud representing the lake's seabed. (c): curve defining the lake's shape. (d): quad mesh of the domain identified by the boundary curve (c). (e): ℓ_2 fitting error color plot. (f): fitted surface determining the bathymetry function h.

	n_b	n_F	n_P	L^∞ error	RMSE	Length	Width	Depth
Rogagua lake	126922	7840	164989	1.043e-00	2.280e-02	520	580	27
Orta lake	268004	16516	25383	6.520e-01	1.557e-02	400	180	22

Table 6.2. Spline space, mesh, point cloud and model features concerning the reconstruction of the bathymetry functions in Figures 6.2 and 6.4.

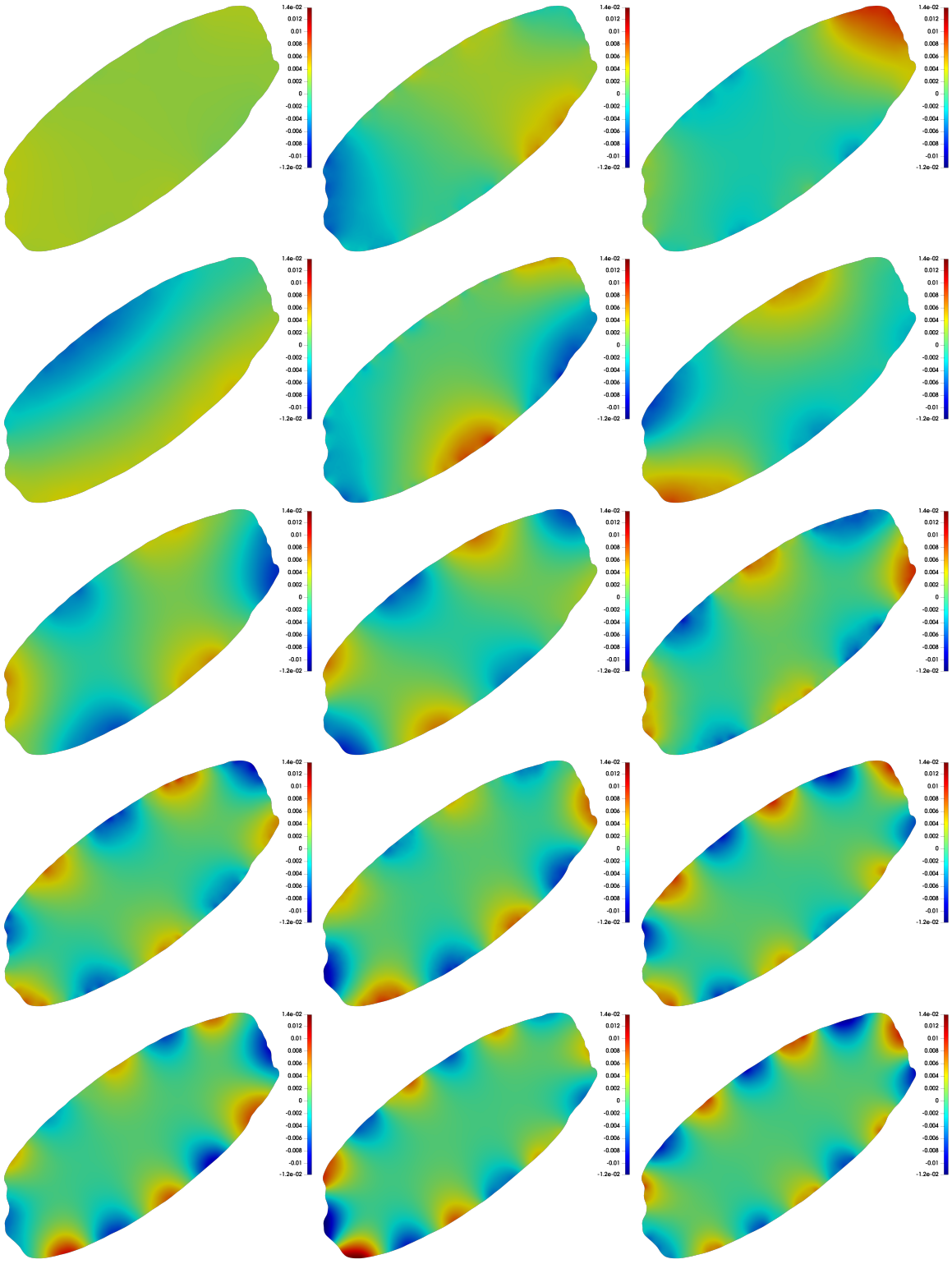


Figure 6.3. Solutions of the shallow-water equation for the Rogagua lake corresponding to the first 15 eigenvalues, ordered in increasing magnitude.

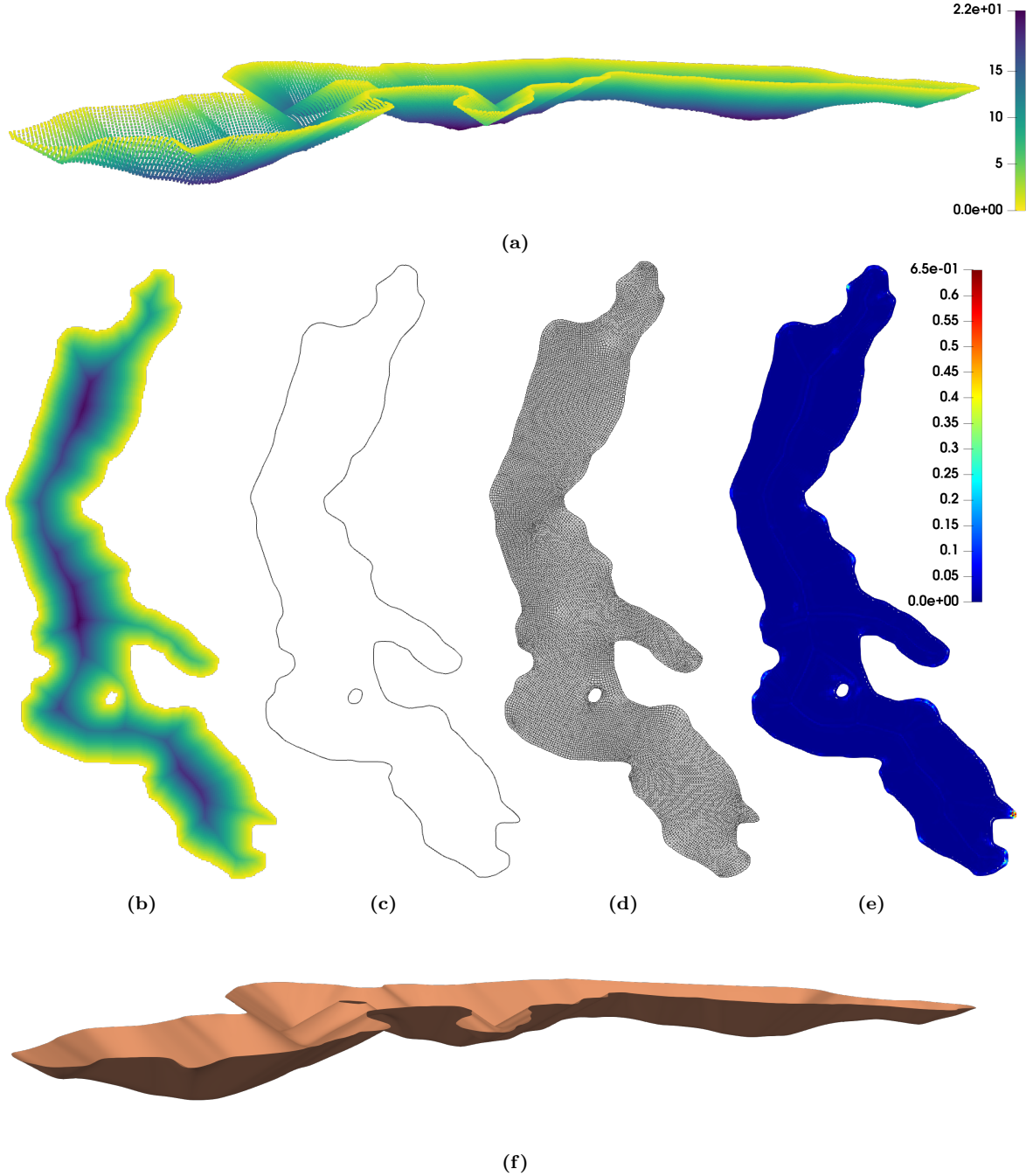


Figure 6.4. Orta lake. (a)-(b): point cloud representing the lake's seabed. (c): curve defining the lake's shape. (d): quad mesh of the domain identified by the boundary curve (c). (e): ℓ_2 fitting error color plot. (f): fitted surface determining the bathymetry function h.

Eigenvalues		
1.681e-09	2.392e-04	3.515e-04
1.368e-03	1.751e-03	2.687e-03
4.473e-03	6.067e-03	8.554e-03
8.964e-03	1.028e-02	1.059e-02
1.113e-02	1.225e-02	1.290e-02

Frequencies		
1.284e-04	4.844e-02	5.871e-02
1.158e-01	1.310e-01	1.623e-01
2.094e-01	2.439e-01	2.896e-01
2.965e-01	3.176e-01	3.223e-01
3.303e-01	3.466e-01	3.556e-01

Table 6.3. First 15 eigenvalues, and corresponding frequencies, for the shallow-water simulation on Orta lake.

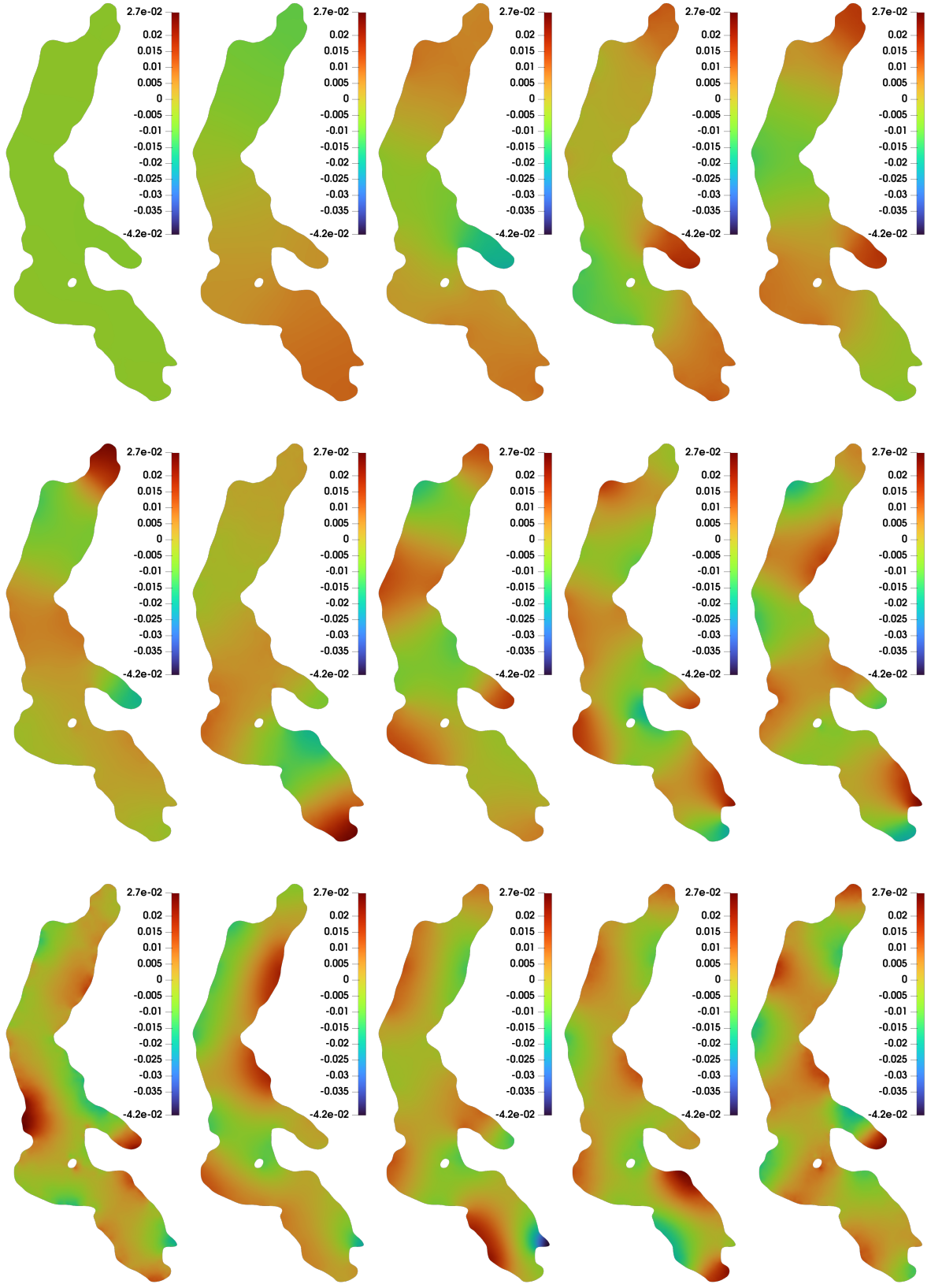


Figure 6.5. Solutions of the shallow-water equation for the Orta lake corresponding to the first 15 eigenvalues, ordered in increasing magnitude.

Summary

This chapter has presented another practical application of the tools introduced in Chapters 2 to 4 to solve the shallow-water equation when dealing with shallow lakes. Given a point cloud reproducing the seabed of a certain lake, we provided a pipeline to obtain from it a quad mesh reproducing the lake's shape; having defined on the latter mesh the G^1 basis functions, we were able, using the same basis set, to both fit the point cloud obtaining a smooth description of the bathymetry function and solve the shallow-water equation using the IGA formulation. Two examples on real lake data have been provided showing the power of the proposed approach.

Chapter 7

Cubic C^2 spline quasi-interpolants on arbitrary triangulations

Quasi-interpolation is a general approach to construct efficient local approximants in a prescribed space to a given function f or just a given set of data. It is often preferred over interpolation because it is a local method and does not require the solution of possibly large linear systems. A quasi-interpolation operator is usually obtained as a linear combination of a suitable set of functions, which are positive, stable, and locally supported, or at least have a strong fast decay, in order to achieve local control.

In this chapter we present the construction of C^2 cubic spline quasi-interpolants on a given arbitrary triangulation where each triangle is refined according to the cubic Wang-Shi (WS) split. The main tool in our construction is the simplex spline basis for the local space of C^2 cubics on the cubic WS split presented in [LMS22]. In order to guarantee global C^2 smoothness between the local simplex spline representations, our construction starts from a Hermite interpolation problem that uniquely identifies elements of the cubic WS spline space on a prescribed triangulation. For a given function f, different C^2 cubic spline quasi-interpolants are then obtained by feeding different sets of Hermite data to this Hermite interpolation problem. Finally, various numerical examples illustrating the performance of the introduced quasi-interpolants are presented. This chapter is based on [MMS23].

7.1 The spline space $\mathbb{S}_3^2(\triangle_{\mathrm{WS}_3})$

Let us consider a reference (macro-)triangle $\triangle = \langle \mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3 \rangle$ identified by three noncollinear points $\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3$ in $\mathbb{R}^2$. We divide each edge of $\triangle$ into three equal segments, respectively, resulting into nine boundary points. Then, we refine $\triangle$ into a number of subelements delineated by the complete graph connecting those boundary points, which consists of the three edges of the triangle and 18 interior lines. The resulting partition is called the WS_3 (or cubic Wang–Shi) split of $\triangle$ as it is the third member of a family of splits originally proposed by Wang and Shi [WS90]; see Figure 7.1 for an illustration. We denote the obtained mesh structure by $\triangle_{\mathrm{WS}_3}$: it has 58 vertices and 75 polygonal regions.

We consider the space of C^2 cubic splines on the partition $\triangle_{\mathrm{WS}_3}$, i.e.,

$$\mathbb{S}_3^2(\triangle_{\mathrm{WS}_3}) := \left\{ s \in C^2(\triangle) : s\Big|_\tau \in \mathbb{P}_3,\ \forall\, \tau \in \mathcal{P}_3 \right\},$$

where $\mathbb{P}_d$ denotes the space of bivariate polynomials of degree at most d and $\mathcal{P}_3$ the set of polygons in $\triangle_{\mathrm{WS}_3}$. The dimension of $\mathbb{S}_3^2(\triangle_{\mathrm{WS}_3})$ equals

$$\dim(\mathbb{S}_3^2(\triangle_{\mathrm{WS}_3})) = \dim(\mathbb{P}_3) + 18 = 28;$$

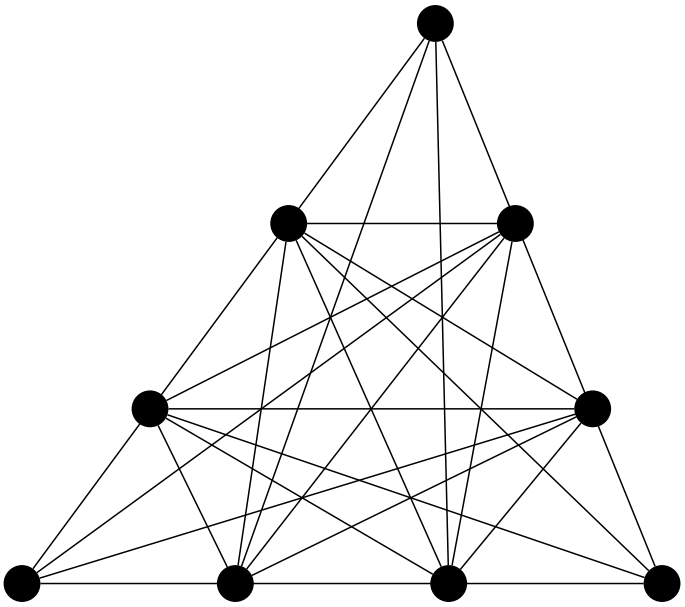


Figure 7.1. Structure of the WS_3 split.

see, e.g., [LMS22, Theorem 2]. The partition $\triangle_{\mathrm{WS}_3}$ looks (and actually is) quite complicated, not only due to the large number of regions but also because of their shape as it is a mixture of triangles, quadrilaterals, and pentagons. Therefore, dealing in practice with the elements of the space $\mathbb{S}_3^2(\triangle_{\mathrm{WS}_3})$ is prohibitive if we just rely on a direct approach by considering a polynomial representation on each region of the split and by taking into account the inter-element smoothness. However, the split has a particularly beautiful feature: it is a cross-cut partition of $\triangle$, formed by the complete graph of the nine points selected on the boundary of $\triangle$. Hence, it is natural to consider bivariate cubic simplex splines (see Section 1.2.2) to bypass the complex geometric structure of the split and to obtain an elegant representation of $\mathbb{S}_3^2(\triangle_{\mathrm{WS}_3})$. Following [LMS22, Theorem 3], a basis of $\mathbb{S}_3^2(\triangle_{\mathrm{WS}_3})$ consisting of (scaled) simplex splines,

$$\{B_1, \ldots, B_{28}\}, \tag{7.1}$$

can be constructed. More precisely, each B_i is obtained by scaling a cubic simplex spline defined by six (counting multiplicity) knots taken among the boundary points of the WS_3 split; the 28 knot sequences are reported in Figure 7.2. There are seven different types of simplex in the set (7.1): Figure 7.4 shows a representative for each type.

The basis functions in (7.1) enjoy several interesting properties:

- the scaling factors ensure that they form a nonnegative partition of unity;
- they inherit recurrence relations and differentiation formulas from the simplex spline structure;
- they admit simple conditions for C^2 joins to neighboring triangles in a given triangulation $\mathcal{T}$;
- cubic polynomials can be represented through a Marsden-like identity;

- they lead to well-conditioned collocation matrices for Lagrange and Hermite interpolation using certain sites;
- a control net can be formed that mimics the shape of the spline function represented in this basis.

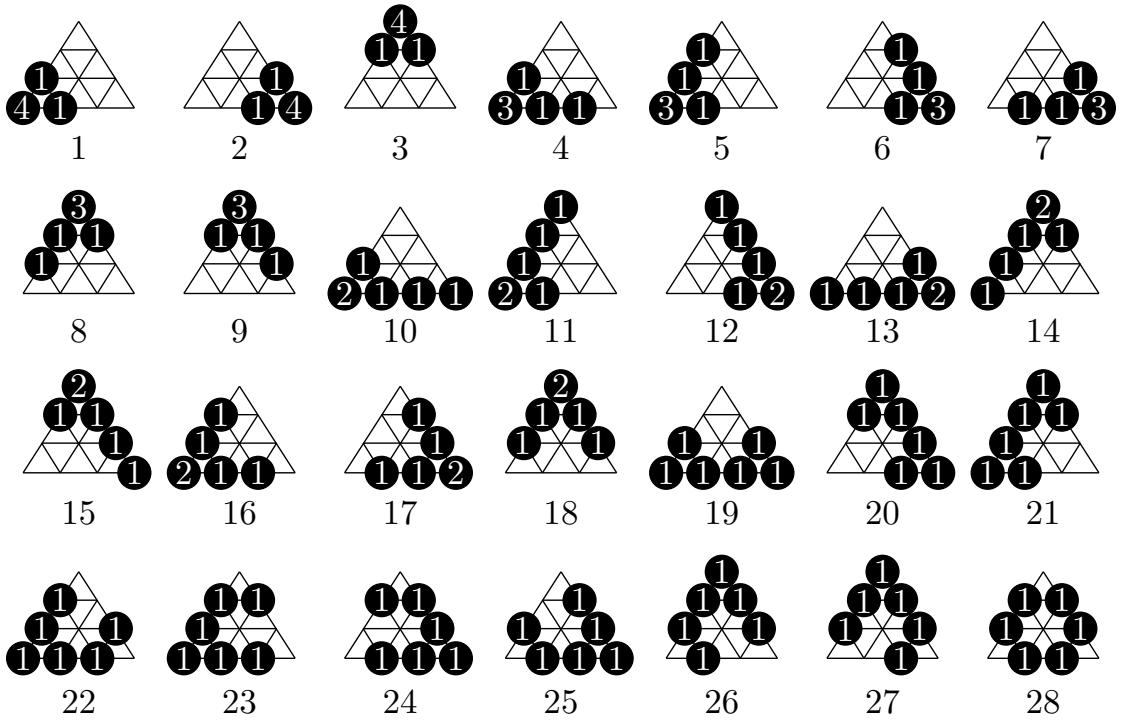


Figure 7.2. From [LMS22, Figure 3]: sequences of knots for a set of simplex spline basis functions for $\mathbb{S}_3^2(\triangle_{\mathrm{WS}_3})$. Each black disc shows the position of a knot, and the number inside indicates its multiplicity.

7.2 Hermite interpolation in $\mathbb{S}_3^2(\triangle_{\mathrm{WS}_3})$

An efficient way to identify the elements of the space $\mathbb{S}_3^2(\triangle_{\mathrm{WS}_3})$ is to solve a suitable Hermite interpolation problem. Taking into account the dimension of the space, a natural choice is to consider six Hermite data values associated with each vertex of $\triangle$, three Hermite data values associated with each edge of $\triangle$, and one function value associated with the triangle $\triangle$ itself. Due to its importance later on, we explicitly state the following result which is a byproduct of [LMS22, Theorem 3].

Proposition 7.2.1. *Let $\mathfrak{f}_j \in \mathbb{R}$, $j = 1, \ldots, 28$, be given. There exists a unique spline $s \in \mathbb{S}_3^2(\triangle_{\mathrm{WS}_3})$ such that*

$$\rho_j(s) = \mathfrak{f}_j, \quad j = 1, \ldots, 28, \tag{7.2}$$

where the operators $\rho_1, \ldots, \rho_{18}$ related to the vertices of $\triangle$ are defined as

$$\begin{aligned}
&\rho_1(\varphi) = \varphi(\mathbf{p}_1), && \rho_2(\varphi) = \varphi(\mathbf{p}_2), && \rho_3(\varphi) = \varphi(\mathbf{p}_3),\\
&\rho_4(\varphi) = D_{\mathbf{p}_1\mathbf{p}_2}\varphi(\mathbf{p}_1), && \rho_5(\varphi) = D_{\mathbf{p}_1\mathbf{p}_3}\varphi(\mathbf{p}_1), && \rho_6(\varphi) = D_{\mathbf{p}_2\mathbf{p}_3}\varphi(\mathbf{p}_2),\\
&\rho_7(\varphi) = D_{\mathbf{p}_2\mathbf{p}_1}\varphi(\mathbf{p}_2), && \rho_8(\varphi) = D_{\mathbf{p}_3\mathbf{p}_1}\varphi(\mathbf{p}_3), && \rho_9(\varphi) = D_{\mathbf{p}_3\mathbf{p}_2}\varphi(\mathbf{p}_3),\\
&\rho_{10}(\varphi) = D^2_{\mathbf{p}_1\mathbf{p}_2}\varphi(\mathbf{p}_1), && \rho_{11}(\varphi) = D^2_{\mathbf{p}_1\mathbf{p}_3}\varphi(\mathbf{p}_1), && \rho_{12}(\varphi) = D^2_{\mathbf{p}_2\mathbf{p}_3}\varphi(\mathbf{p}_2),\\
&\rho_{13}(\varphi) = D^2_{\mathbf{p}_2\mathbf{p}_1}\varphi(\mathbf{p}_2), && \rho_{14}(\varphi) = D^2_{\mathbf{p}_3\mathbf{p}_1}\varphi(\mathbf{p}_3), && \rho_{15}(\varphi) = D^2_{\mathbf{p}_3\mathbf{p}_2}\varphi(\mathbf{p}_3),\\
&\rho_{16}(\varphi) = D_{\mathbf{p}_1\mathbf{p}_2}D_{\mathbf{p}_1\mathbf{p}_3}\varphi(\mathbf{p}_1), && \rho_{17}(\varphi) = D_{\mathbf{p}_2\mathbf{p}_3}D_{\mathbf{p}_2\mathbf{p}_1}\varphi(\mathbf{p}_2), && \rho_{18}(\varphi) = D_{\mathbf{p}_3\mathbf{p}_1}D_{\mathbf{p}_3\mathbf{p}_2}\varphi(\mathbf{p}_3),
\end{aligned} \tag{7.3}$$

the operators $\rho_{19}, \ldots, \rho_{27}$ related to the edges of $\triangle$ are defined as

$$\begin{aligned}
&\rho_{19}(\varphi) = D_{\mathbf{q}_3\mathbf{p}_3}\varphi(\mathbf{q}_3), && \rho_{20}(\varphi) = D_{\mathbf{q}_1\mathbf{p}_1}\varphi(\mathbf{q}_1), && \rho_{21}(\varphi) = D_{\mathbf{q}_2\mathbf{p}_2}\varphi(\mathbf{q}_2),\\
&\rho_{22}(\varphi) = D^2_{\mathbf{p}_{3,1}\mathbf{p}_3}\varphi(\mathbf{p}_{3,1}), && \rho_{23}(\varphi) = D^2_{\mathbf{p}_{2,1}\mathbf{p}_2}\varphi(\mathbf{p}_{2,1}), && \rho_{24}(\varphi) = D^2_{\mathbf{p}_{1,2}\mathbf{p}_1}\varphi(\mathbf{p}_{1,2}),\\
&\rho_{25}(\varphi) = D^2_{\mathbf{p}_{3,2}\mathbf{p}_3}\varphi(\mathbf{p}_{3,2}), && \rho_{26}(\varphi) = D^2_{\mathbf{p}_{2,3}\mathbf{p}_2}\varphi(\mathbf{p}_{2,3}), && \rho_{27}(\varphi) = D^2_{\mathbf{p}_{1,3}\mathbf{p}_1}\varphi(\mathbf{p}_{1,3}),
\end{aligned} \tag{7.4}$$

and finally the operator ρ_{28} related to the triangle $\triangle$ is defined as

$$\rho_{28}(\varphi) = \varphi(\mathbf{q}), \tag{7.5}$$

with (see also Figure 7.3)

$$\begin{aligned}
&\mathbf{p}_{1,2} = \frac{2}{3}\mathbf{p}_2 + \frac{1}{3}\mathbf{p}_3\,, && \mathbf{p}_{1,3} = \frac{1}{3}\mathbf{p}_2 + \frac{2}{3}\mathbf{p}_3\,, && \mathbf{q}_1 = \frac{1}{2}\mathbf{p}_2 + \frac{1}{2}\mathbf{p}_3\,,\\
&\mathbf{p}_{2,1} = \frac{2}{3}\mathbf{p}_1 + \frac{1}{3}\mathbf{p}_3\,, && \mathbf{p}_{2,3} = \frac{1}{3}\mathbf{p}_1 + \frac{2}{3}\mathbf{p}_3\,, && \mathbf{q}_2 = \frac{1}{2}\mathbf{p}_1 + \frac{1}{2}\mathbf{p}_3\,,\\
&\mathbf{p}_{3,1} = \frac{2}{3}\mathbf{p}_1 + \frac{1}{3}\mathbf{p}_2\,, && \mathbf{p}_{3,2} = \frac{1}{3}\mathbf{p}_1 + \frac{2}{3}\mathbf{p}_2\,, && \mathbf{q}_3 = \frac{1}{2}\mathbf{p}_1 + \frac{1}{2}\mathbf{p}_2,
\end{aligned} \tag{7.6}$$

and

$$\mathbf{q} = \frac{1}{3}\mathbf{p}_1 + \frac{1}{3}\mathbf{p}_2 + \frac{1}{3}\mathbf{p}_3. \tag{7.7}$$

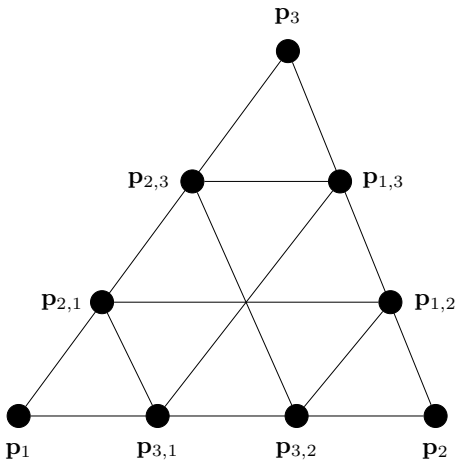


Figure 7.3. Labeling of the knots on the boundary of the triangle $\triangle$.

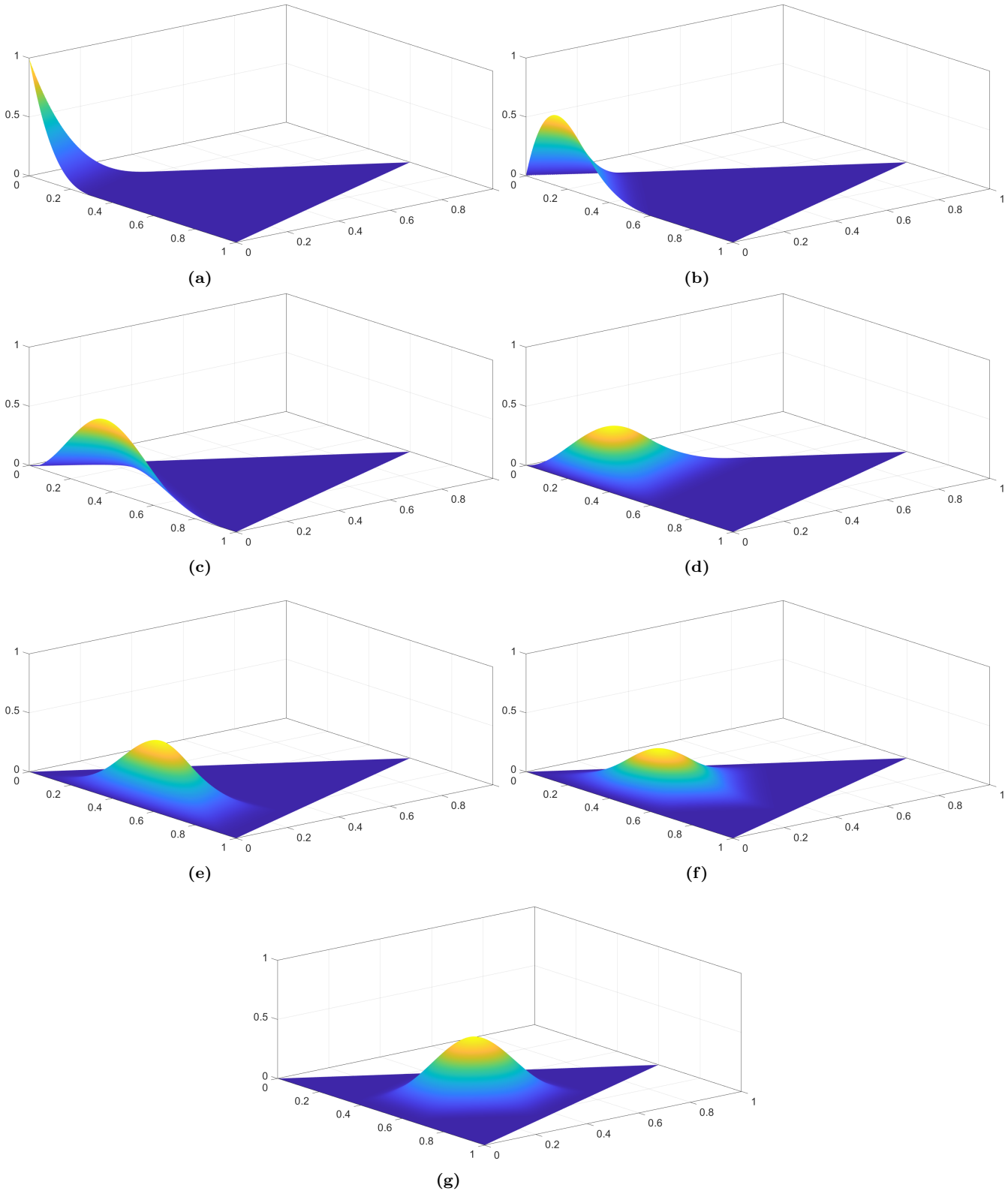


Figure 7.4. From (a) to (g): the simplex spline basis function B_1, B_4, B_{10}, B_{16}, B_{19}, B_{22}, B_{28}.

For the sake of completeness, we report in Table 7.1 the collocation matrix of the operators ρ_j applied to the basis functions B_i, $i, j = 1, \dots, 28$; see also [LMS22, Section A.3]. From [LMS22, Corollary 1] we know the following related result.

Corollary 7.2.2. *For given data $f_{k,\alpha,\beta}$, g_k, $g_{k,l}$, and h_0, there exists a unique spline $s \in \mathbb{S}_3^2(\triangle_{\mathrm{WS}_3})$ such that*

$$\begin{aligned} D_x^\alpha D_y^\beta s(\mathbf{p}_k) &= f_{k,\alpha,\beta}, \quad 0 \le \alpha+\beta \le 2, \quad k = 1,2,3, \\ D_{\mathbf{n}_k} s(\mathbf{q}_k) &= g_k, \quad D_{\mathbf{n}_k}^2 s(\mathbf{p}_{k,l}) = g_{k,l}, \quad k,l = 1,2,3, \quad k \neq l, \\ s(\mathbf{q}) &= h_0, \end{aligned} \tag{7.8}$$

where $\mathbf{n}_k$ is the outgoing normal direction of the edge opposite to the vertex $\mathbf{p}_k$, and the points $\mathbf{p}_{k,l}$, $\mathbf{q}_k$ and $\mathbf{q}$ are defined in (7.6) and (7.7).

7.3 The spline space $\mathbb{S}_3^2(\mathcal{T}_{\mathrm{WS}_3})$

Let $\mathcal{T}$ be a triangulation of a polygonal domain $\Omega \in \mathbb{R}^2$ and let $\mathcal{T}_{\mathrm{WS}_3}$ denote its refinement obtained by taking the WS_3 split of each of its triangles. We consider the space of C^2 cubic splines on $\mathcal{T}_{\mathrm{WS}_3}$, i.e.,

$$\mathbb{S}_3^2(\mathcal{T}_{\mathrm{WS}_3}) := \left\{ s \in C^2(\Omega) : s\Big|_\tau \in \mathbb{P}_3, \ \tau \text{ is polygon in } \mathcal{T}_{\mathrm{WS}_3} \right\}.$$

The unisolvency of the Hermite interpolation problem stated in Corollary 7.2.2 implies that

$$\dim(\mathbb{S}_3^2(\mathcal{T}_{\mathrm{WS}_3})) = 6n_V + 3n_E + n_T,$$

where n_V, n_E, n_T are the number of vertices, edges, and triangles of $\mathcal{T}$, respectively; see [LMS22; Wan01; WS90]. Moreover, any spline function of $\mathbb{S}_3^2(\mathcal{T}_{\mathrm{WS}_3})$ can be locally constructed on each (macro-)triangle $\triangle^i$ of $\mathcal{T}$ via the Hermite data (7.8), and the corresponding spline piece on $\triangle^i$ can be represented in the form

$$s\Big|_{\triangle^i} = \sum_{j=1}^{28} c_j^i B_j^i, \quad c_j^i \in \mathbb{R}, \tag{7.9}$$

where $\{B_1^i, \dots, B_{28}^i\}$ is the scaled simplex spline local basis (7.1) for the triangle $\triangle^i$.

Any function, which is represented locally in the form (7.9) on each $\triangle^i$ of $\mathcal{T}$, is C^2-smooth over $\triangle^i$ but not necessary C^2-smooth across the edges of $\mathcal{T}$. Conditions on the local spline coefficients c_j^i to ensure global C^2 smoothness are discussed in [LMS22, Section 4]. The construction of a stable global basis with local support for $\mathbb{S}_3^2(\mathcal{T}_{\mathrm{WS}_3})$ is also presented in [LMS22, Section 4].

	ρ_1	ρ_2	ρ_3	ρ_4	ρ_5	ρ_6	ρ_7	ρ_8	ρ_9	ρ_{10}	ρ_{11}	ρ_{12}	ρ_{13}	ρ_{14}	ρ_{15}	ρ_{16}	ρ_{17}	ρ_{18}	ρ_{19}	ρ_{20}	ρ_{21}	ρ_{22}	ρ_{23}	ρ_{24}	ρ_{25}	ρ_{26}	ρ_{27}	ρ_{28}
B_1	1	0	0	-9	-9	0	0	0	0	54	54	0	0	0	0	54	0	0	0	0	0	0	0	0	0	0	0	0
B_2	0	1	0	0	0	-9	-9	0	0	0	0	54	54	0	0	0	54	0	0	0	0	0	0	0	0	0	0	0
B_3	0	0	1	0	0	0	0	-9	-9	0	0	0	0	54	54	0	0	54	0	0	0	0	0	0	0	0	0	0
B_4	0	0	0	9	0	0	0	0	0	-81	0	0	0	0	0	-54	0	0	$-\frac{27}{32}$	0	0	$\frac{75}{2}$	0	0	0	0	0	0
B_5	0	0	0	0	9	0	0	0	0	0	-81	0	0	0	0	-54	0	0	0	0	$-\frac{27}{32}$	0	$\frac{75}{2}$	0	0	0	0	0
B_6	0	0	0	0	0	9	0	0	0	0	0	-81	0	0	0	0	-54	0	0	$-\frac{27}{32}$	0	0	0	$\frac{75}{2}$	0	0	0	0
B_7	0	0	0	0	0	0	9	0	0	0	0	0	-81	0	0	0	-54	0	$-\frac{27}{32}$	0	0	0	0	0	$\frac{75}{2}$	0	0	0
B_8	0	0	0	0	0	0	0	9	0	0	0	0	0	-81	0	0	0	-54	0	0	$-\frac{27}{32}$	0	0	0	0	$\frac{75}{2}$	0	0
B_9	0	0	0	0	0	0	0	0	9	0	0	0	0	0	-81	0	0	-54	0	$-\frac{27}{32}$	0	0	0	0	0	0	$\frac{75}{2}$	0
B_{10}	0	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0	0	0	$-\frac{189}{32}$	0	0	$\frac{31}{2}$	0	0	49	0	0	0
B_{11}	0	0	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0	0	0	0	$-\frac{189}{32}$	0	$\frac{31}{2}$	0	0	49	0	0
B_{12}	0	0	0	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0	0	$-\frac{189}{32}$	0	0	0	$\frac{31}{2}$	0	0	49	0
B_{13}	0	0	0	0	0	0	0	0	0	0	0	0	27	0	0	0	0	0	$-\frac{189}{32}$	0	0	49	0	0	$\frac{31}{2}$	0	0	0
B_{14}	0	0	0	0	0	0	0	0	0	0	0	0	0	27	0	0	0	0	0	0	$-\frac{189}{32}$	0	49	0	0	$\frac{31}{2}$	0	0
B_{15}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	0	0	0	0	$-\frac{189}{32}$	0	0	0	49	0	0	$\frac{31}{2}$	0
B_{16}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	54	0	0	$\frac{9}{8}$	0	$\frac{9}{8}$	-63	-63	0	0	0	0	0
B_{17}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	54	0	$\frac{9}{8}$	$\frac{9}{8}$	0	0	0	-63	-63	0	0	0
B_{18}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	54	0	$\frac{9}{8}$	$\frac{9}{8}$	0	0	0	0	-63	-63	0
B_{19}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\frac{45}{4}$	0	0	-120	0	0	-120	0	0	0
B_{20}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\frac{45}{4}$	0	0	0	-120	0	0	-120	0
B_{21}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\frac{45}{4}$	0	-120	0	0	-120	0	0
B_{22}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	54	27	0	0	0	0	$\frac{1}{12}$
B_{23}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	54	0	0	0	0	$\frac{1}{12}$
B_{24}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	54	27	0	0	$\frac{1}{12}$
B_{25}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	54	0	0	$\frac{1}{12}$
B_{26}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	54	27	$\frac{1}{12}$
B_{27}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	27	54	$\frac{1}{12}$
B_{28}	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	$\frac{1}{2}$

Table 7.1. Values of $\rho_j(B_i)$ for $i, j = 1, \ldots, 28$, where ρ_j is defined in (7.3), (7.4) and (7.5).

7.4 Construction of quasi-interpolants

In this section we present the construction of three C^2 cubic quasi-interpolants in $\mathbb{S}_3^2(\mathcal{T}_{\mathrm{WS}_3})$ for an arbitrary triangulation $\mathcal{T}$. For a given (sufficiently smooth) function f, the common idea of the three approaches is to build a quasi-interpolant $\mathcal{Q}f$ by using the local simplex spline basis (7.1) on each macro-triangle $\triangle^i$ of the triangulation refined according to the WS_3 split. More formally,

$$f \longrightarrow \mathcal{Q}f \in \mathbb{S}_3^2(\mathcal{T}_{\mathrm{WS}_3}), \quad \mathcal{Q}f\Big|_{\triangle^i} = \sum_{j=1}^{28} \mu_j^i(f) B_j^i, \quad i = 1, \dots, n_T,$$

where $\{\mu_j^i(f)\}_{i=1,j=1}^{n_T,28}$ are linear functionals to be constructed so that the following properties hold:

- global smoothness: $\mathcal{Q}f$ is globally C^2;
- polynomial reproduction: $\mathcal{Q}p = p$, $\forall p \in \mathbb{P}_3$;
- locality: $\mu_j^i(f)$ depends on values (and/or derivatives) of f only on triangles $\triangle^k$ such that $\triangle^k \cap \triangle^i \neq \emptyset$.

In the spirit of finite (macro-)elements, an efficient way to obtain linear functionals μ_j^i enjoying the above properties is to construct $\mathcal{Q}f|_{\triangle^i}$ as the solution of the Hermite problem in Corollary 7.2.2 where the Hermite data are directly taken (see Section 7.4.2) or accurately estimated (see Sections 7.4.3 and 7.4.4) from f.

A direct solution of such a Hermite problem, however, would require a triangle-dependent collocation matrix and thus would lead to the computation of n_T different collocation matrices in $\mathcal{T}$. To circumvent this issue, we consider the Hermite problem stated in (7.2), whose collocation matrix is the same for all triangles in $\mathcal{T}$. The values $\mathfrak{f}_j$ at the right-hand side have to be properly deduced from those in (7.8) in order to get global C^2 smoothness. This is briefly described in the next section.

7.4.1 Consistent local Hermite data

Suppose the Hermite data (7.8) are given for the triangle $\triangle$. In this section we outline how to compute the values $\mathfrak{f}_j$, $j = 1, \dots, 28$ so that the solution s of the problem stated in Proposition 7.2.1 solves the Hermite interpolation problem stated in Corollary 7.2.2 as well.

Of course, $\mathfrak{f}_{28} = h_0$. Moreover, the values $\mathfrak{f}_j$, $j = 1, \dots, 18$, can be immediately obtained by standard calculus from the data

$$f_{k,\alpha,\beta}, \quad 0 \leq \alpha + \beta \leq 2, \quad k = 1, 2, 3.$$

Let us now focus on $\mathfrak{f}_{19}$. The direction $\mathbf{q}_3\mathbf{p}_3$ involved in the definition of the operator ρ_{19} (see Figure 7.5) can be written as

$$\mathbf{q}_3\mathbf{p}_3 = \gamma \mathbf{p}_1\mathbf{p}_2 + \delta \mathbf{n}_3,$$

for some real values γ, δ. Therefore, from (7.8) we deduce

$$\rho_{19}(s) = D_{\mathbf{q}_3\mathbf{p}_3} s(\mathbf{q}_3) = \gamma D_{\mathbf{p}_1\mathbf{p}_2} s(\mathbf{q}_3) + \delta D_{\mathbf{n}_3} s(\mathbf{q}_3) = \gamma D_{\mathbf{p}_1\mathbf{p}_2} s(\mathbf{q}_3) + \delta g_3.$$

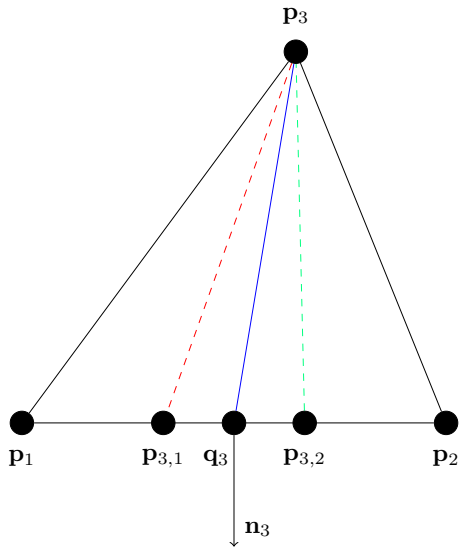


Figure 7.5. Example of directions attached to the edge $\mathbf{p}_1\mathbf{p}_2$ defining the operators $\rho_{19}, \rho_{22}, \rho_{25}$ and its outgoing normal.

Then, to obtain a consistent value for $\mathfrak{f}_{19}$, we only need to compute the value $D_{\mathbf{p}_1\mathbf{p}_2}s(\mathbf{q}_3)$. This can be achieved by observing that, since $s \in \mathbb{S}_3^2(\triangle_{\mathrm{WS}_3})$, its restriction to the edge $\mathbf{p}_1\mathbf{p}_2$ is a univariate C^2 cubic spline with two inner knots at $\mathbf{p}_{3,1}$ and $\mathbf{p}_{3,2}$, respectively. Thus, it is uniquely determined by its values and first and second derivatives at the two ends of the edge, i.e., by the values of $\rho_1(s), \rho_2(s), \rho_4(s), \rho_7(s), \rho_{10}(s), \rho_{13}(s)$ or, in other words, by the values $\mathfrak{f}_1, \mathfrak{f}_2, \mathfrak{f}_4, \mathfrak{f}_7, \mathfrak{f}_{10}, \mathfrak{f}_{13}$ previously computed. This allows us to compute $D_{\mathbf{p}_1\mathbf{p}_2}s(\mathbf{q}_3)$ and so to get a consistent value for $\mathfrak{f}_{19}$. Similarly, we can compute $\mathfrak{f}_{20}$ and $\mathfrak{f}_{21}$.

Now, let us consider $\mathfrak{f}_{22}$. The direction $\mathbf{p}_{3,1}\mathbf{p}_3$ involved in the definition of the operator ρ_{22} (see Figure 7.5) can be written as

$$\mathbf{p}_{3,1}\mathbf{p}_3 = \eta\mathbf{p}_1\mathbf{p}_2 + \zeta\mathbf{n}_3,$$

for some real values η, ζ. Thus, we have

$$\rho_{22}(s) = D^2_{\mathbf{p}_{3,1}\mathbf{p}_3}s(\mathbf{p}_{3,1}) = \eta^2 D^2_{\mathbf{p}_1\mathbf{p}_2}s(\mathbf{p}_{3,1}) + 2\eta\zeta D_{\mathbf{p}_1\mathbf{p}_2}D_{\mathbf{n}_3}s(\mathbf{p}_{3,1}) + \zeta^2 D^2_{\mathbf{n}_3}s(\mathbf{p}_{3,1}).$$

Hence, from (7.8) we set

$$\mathfrak{f}_{22} = \eta^2 D^2_{\mathbf{p}_1\mathbf{p}_2}s(\mathbf{p}_{3,1}) + 2\eta\zeta D_{\mathbf{p}_1\mathbf{p}_2}D_{\mathbf{n}_3}s(\mathbf{p}_{3,1}) + \zeta^2 g_{3,1}.$$

The value $D^2_{\mathbf{p}_1\mathbf{p}_2}s(\mathbf{p}_{3,1})$ can be computed from the analytical expression of s along the edge $\mathbf{p}_1\mathbf{p}_2$ previously obtained. Finally, the restriction of $D_{\mathbf{n}_3}s$ along the edge $\mathbf{p}_1\mathbf{p}_2$ is a univariate C^1 quadratic spline with two inner knots at $\mathbf{p}_{3,1}$ and $\mathbf{p}_{3,2}$, respectively. Therefore, such univariate spline is uniquely determined by its values and first derivatives at the two ends of the edge, which can be computed from

$$f_{k,\alpha,\beta}, \quad 1 \le \alpha + \beta \le 2, \quad k = 1, 2,$$

and by its value at the midpoint $\mathbf{q}_3$, i.e., by g_3. Similarly, we can compute $\mathfrak{f}_j$, $j = 23, \ldots, 27$.

It is worth to notice that the computed values $\mathfrak{f}_j$, $j = 1, \ldots, 28$, are linear expressions in terms of the given Hermite data (7.8).

7.4.2 Quasi-interpolant from exact Hermite data

For this construction, we assume that for each triangle $\triangle^i$ of $\mathcal{T}$, $i = 1, \ldots, n_T$, the Hermite data

$$f^i_{k,\alpha,\beta}, \quad g^i_k, \quad g^i_{k,l}, \quad \text{and} \quad h^i_0,$$

as in (7.8), are sampled from a given (sufficiently smooth) function f. These data allow us to compute the values

$$\mathfrak{f}^i_{\text{exa}} = (\mathfrak{f}^i_{\text{exa},1}, \ldots, \mathfrak{f}^i_{\text{exa},28})^T \tag{7.10}$$

according to the strategy outlined in Section 7.4.1. Then, to find the spline satisfying the problem stated in (7.2), we solve the linear system

$$B\, \boldsymbol{\mu}^i_{\text{exa}}(f) = \mathfrak{f}^i_{\text{exa}},$$

where the matrix $B \in \mathbb{R}^{28\times 28}$ is given by

$$B = (b_{l,k})_{l,k=1}^{28} = \left(\rho_l(B^i_k)\right)_{l,k=1}^{28}. \tag{7.11}$$

Note that, due to the definition of ρ_l, $l = 1, \ldots 28$, and B^i_k, $k = 1, \ldots, 28$, the matrix B is independent of i. Actually, it is just the transpose of the matrix reported in Table 7.1. It can be directly checked that

$$\|B\|_\infty = 366, \quad \|B^{-1}\|_\infty = \frac{3523}{729} \leq 5. \tag{7.12}$$

Let

$$\boldsymbol{\mu}^i_{\text{exa}}(f) = (\mu^i_{\text{exa},1}(f), \ldots, \mu^i_{\text{exa},28}(f))^T,$$

we define $\mathcal{Q}_{\text{exa}} : C^2(\Omega) \longrightarrow \mathbb{S}^2_3(\mathcal{T}_{\text{WS}_3})$ by

$$\mathcal{Q}_{\text{exa}} f\Big|_{\triangle^i} = \sum_{j=1}^{28} \mu^i_{\text{exa},j}(f) B^i_j, \quad i = 1, \ldots, n_T. \tag{7.13}$$

By construction, the quasi-interpolant $\mathcal{Q}_{\text{exa}} f$ enjoys the global smoothness, cubic polynomial reproduction, and locality properties stated at the beginning of Section 7.4. Moreover, we have the following result as a consequence from Corollary 7.2.2.

Proposition 7.4.1. *The quasi-interpolant in (7.13) is a projector onto $\mathbb{S}^2_3(\mathcal{T}_{\text{WS}_3})$, i.e.,*

$$\mathcal{Q}_{\text{exa}} s = s, \quad \forall s \in \mathbb{S}^2_3(\mathcal{T}_{\text{WS}_3}).$$

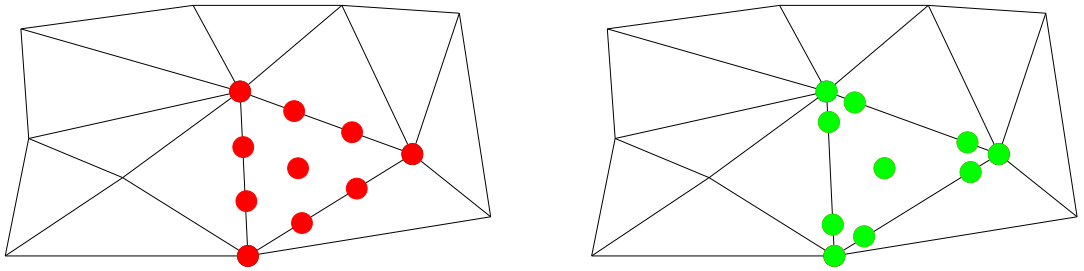
Let $\|f\|_{\infty,\Omega}$ be the standard L^∞-norm of the function f on the domain Ω,

$$\|D^k f\|_{\infty,\Omega} = \max_{\alpha+\beta=k} \|D^\alpha_x D^\beta_y f\|_{\infty,\Omega},$$

and let $h_\mathcal{T}$ be the length of the longest edge of the triangulation $\mathcal{T}$ with $\theta_\mathcal{T}$ its smallest angle. Then, from its construction described in Section 7.4.1, we know that $\mathfrak{f}^i_{\text{exa}}$ in (7.10) can be bounded by

$$\|\mathfrak{f}^i_{\text{exa}}\|_\infty \leq C\left(\|f\|_{\infty,\Omega} + h_\mathcal{T}\|Df\|_{\infty,\Omega} + h^2_\mathcal{T}\|D^2 f\|_{\infty,\Omega}\right), \tag{7.14}$$

for some constant C, which is independent of f and $h_\mathcal{T}$ but depends on $\theta_\mathcal{T}$.



(a) (b)

Figure 7.6. (a) Positioning of the Bézier domain points (7.15) on a triangle. (b) Positioning of the modified Bézier domain points (7.21) with $\varepsilon = 1/2$ on a triangle.

Proposition 7.4.2. *The quasi-interpolant in* (7.13) *satisfies*

$$\|\mathcal{Q}_{\text{exa}} f\|_{\infty,\Omega} \leq 5C \left(\|f\|_{\infty,\Omega} + h_{\mathcal{T}} \|Df\|_{\infty,\Omega} + h_{\mathcal{T}}^2 \|D^2 f\|_{\infty,\Omega} \right),$$

where the constant C is defined in (7.14).

Proof. Consider a triangle $\triangle^i$ of $\mathcal{T}$. Since the basis functions in (7.1) form a nonnegative partition of unity on $\triangle^i$, we have

$$\|\mathcal{Q}_{\text{exa}} f\|_{\infty,\triangle^i} \leq \|\boldsymbol{\mu}^i_{\text{exa}}(f)\|_\infty \leq \|B^{-1}\|_\infty \|\mathbf{f}^i_{\text{exa}}\|_\infty.$$

The bounds in (7.12) and (7.14) conclude the proof. □

7.4.3 Construction with Hermite data from averaged polynomials

In practical problems it is often unrealistic to assume the availability of data describing a bivariate function up to second order derivatives. Hence, in order to solve the problem stated in Corollary 7.2.2, we need to be able to generate reasonable values up to second derivatives for our spline approximation. We describe a first strategy in this section, making use of local cubic polynomial interpolants defined on each triangle of $\mathcal{T}$.

Let us start by defining the Bézier domain points $\mathbf{b}^i_\ell$, $\ell = 1, \ldots, 10$, related to the triangle $\triangle^i$ of $\mathcal{T}$, $i \in \{1, \ldots, n_T\}$. For simplicity, we define them in barycentric coordinates:

$$\begin{aligned}
&\mathbf{b}^i_1 = (1,0,0)\,, \quad \mathbf{b}^i_2 = \left(\frac{2}{3},\frac{1}{3},0\right), \quad \mathbf{b}^i_3 = \left(\frac{2}{3},0,\frac{1}{3}\right), \quad \mathbf{b}^i_4 = \left(\frac{1}{3},\frac{2}{3},0\right), \quad \mathbf{b}^i_5 = \left(\frac{1}{3},\frac{1}{3},\frac{1}{3}\right), \\
&\mathbf{b}^i_6 = \left(\frac{1}{3},0,\frac{2}{3}\right), \quad \mathbf{b}^i_7 = (0,1,0)\,, \quad \mathbf{b}^i_8 = \left(0,\frac{2}{3},\frac{1}{3}\right), \quad \mathbf{b}^i_9 = \left(0,\frac{1}{3},\frac{2}{3}\right), \quad \mathbf{b}^i_{10} = (0,0,1)\,.
\end{aligned} \tag{7.15}$$

See Figure 7.6-(a) for an illustration of these points. Suppose we know the values of a function f at them, i.e.,

$$z^i_\ell = f(\mathbf{b}^i_\ell), \quad \ell = 1, \ldots, 10,$$

and set $\mathbf{z}^i = (z^i_1, \ldots, z^i_{10})^T$. We can then build the local cubic polynomial interpolant $\mathcal{I}^i f$ on $\triangle^i$, defined by

$$\mathcal{I}^i f = \sum_{0 \leq \alpha+\beta \leq 3} c^i_{\alpha,\beta} x^\alpha y^\beta, \tag{7.16}$$

where $\mathbf{c}^i = (c^i_{0,0}, c^i_{0,1} \ldots, c^i_{3,0})^T$ is obtained by solving the linear system

$$V_i\, \mathbf{c}^i = \mathbf{z}^i,$$

and $V_i \in \mathbb{R}^{10\times 10}$ is the Vandermonde matrix in terms of the monomial basis of $\mathbb{P}_3$ evaluated at the Bézier domain points of $\triangle^i$. The idea behind this construction is to use these local polynomials to produce reasonable derivative values to be used in our approximation process. Since the Hermite data in (7.3) to (7.5) refer to values attached to the vertices, edges, and triangles of $\mathcal{T}$, we can orient our investigation towards the same features.

Let $\mathbf{v}^k$ be a vertex of $\mathcal{T}$, $k \in \{1, \ldots, n_V\}$. Making use of the local interpolants defined over the triangles sharing the vertex $\mathbf{v}^k$, we define the averaged vertex polynomial $\mathcal{V}^k f$ as

$$\mathcal{V}^k f = \sum_{l\,:\,\mathbf{v}^k \in \triangle^l} \omega_l\, \mathcal{I}^l f, \quad \omega_l = \frac{\dfrac{1}{|\triangle^l|}}{\displaystyle\sum_{m\,:\,\mathbf{v}^k \in \triangle^m} \frac{1}{|\triangle^m|}}, \tag{7.17}$$

where $|\triangle|$ stands for the area of the triangle $\triangle$. The choice of the weights ω_l in the above averaging favors the polynomials defined over triangles with small area. Similarly to the vertex case, for each edge $\mathbf{e}^j$ of $\mathcal{T}$, $j \in \{1, \ldots, n_E\}$, we introduce the averaged edge polynomial $\mathcal{E}^j f$ as

$$\mathcal{E}^j f = \sum_{l\,:\,\mathbf{e}^j \in \triangle^l} \omega_l\, \mathcal{I}^l f, \quad \omega_l = \frac{\dfrac{1}{|\triangle^l|}}{\displaystyle\sum_{m\,:\,\mathbf{e}^j \in \triangle^m} \frac{1}{|\triangle^m|}}. \tag{7.18}$$

Furthermore, to define the triangle (or face) polynomial $\mathcal{F}^i f$ attached to the triangle $\triangle^i$, $i \in \{1, \ldots, n_T\}$, we simply take the local cubic interpolant $\mathcal{I}^i f$, i.e.,

$$\mathcal{F}^i f = \mathcal{I}^i f. \tag{7.19}$$

The positioning of these vertex, edge, and triangle polynomials is visualized in Figure 7.7-(a).

We have now all the ingredients at hand to build the quasi-interpolant $\mathcal{Q}_{\text{ave}} f$. Consider the triangle $\triangle^i$ of $\mathcal{T}$, $i \in \{1, \ldots, n_T\}$. With reference to (7.8), from the vertex polynomials $\mathcal{V}^k f$ in (7.17) we sample the values $f^i_{k,\alpha,\beta}$; from the edge polynomials $\mathcal{E}^j f$ in (7.18) we sample the values g^i_k and $g^i_{k,l}$; and finally, the value h^i_0 is sampled from the triangle polynomial $\mathcal{F}^i f$ in (7.19). Given these Hermite data we follow the same construction as in Section 7.4.2.

More precisely, according to the strategy outlined in Section 7.4.1, we first compute the values

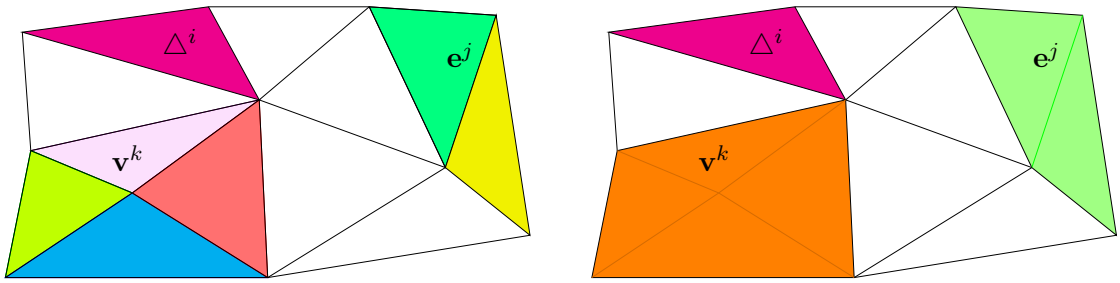
$$\mathfrak{f}^i_{\text{ave}} = (\mathfrak{f}^i_{\text{ave},1}, \ldots, \mathfrak{f}^i_{\text{ave},28})^T,$$

and solve the linear system

$$B\, \boldsymbol{\mu}^i_{\text{ave}}(f) = \mathfrak{f}^i_{\text{ave}},$$

with B defined in (7.11). Let

$$\boldsymbol{\mu}^i_{\text{ave}}(f) = (\mu^i_{\text{ave},1}(f), \ldots, \mu^i_{\text{ave},28}(f))^T,$$



(a) **(b)**

Figure 7.7. (a): visualization of the positioning of the local cubic polynomial interpolants attached to the vertex $\mathbf{v}^k$, edge $\mathbf{e}^j$, and triangle $\triangle^i$. (b): visualization of the positioning of the weighted least-squares cubic polynomials around the vertex $\mathbf{v}^k$, edge $\mathbf{e}^j$, and triangle $\triangle^i$.

we then define the quasi-interpolant $\mathcal{Q}_{\text{ave}} : C^0(\Omega) \longrightarrow \mathbb{S}_3^2(\mathcal{T}_{\text{WS}_3})$ by

$$\mathcal{Q}_{\text{ave}} f\Big|_{\triangle^i} = \sum_{j=1}^{28} \mu^i_{\text{ave},j}(f) B^i_j, \quad i = 1, \ldots, n_T. \tag{7.20}$$

From its construction it follows that the quasi-interpolant $\mathcal{Q}_{\text{ave}} f$ enjoys the global smoothness, cubic polynomial reproduction, and locality properties stated at the beginning of Section 7.4.

7.4.4 Construction with Hermite data from local weighted least-squares polynomials

In this section we propose a different method to reconstruct the Hermite data required in Corollary 7.2.2 by building local cubic polynomials through weighted least squares.

Let us assume, as in Section 7.4.3, that the function values of f are known in certain points, which we select to be a modification of the standard Bézier domain points (7.15). Fix $\varepsilon > 0$. For a given triangle $\triangle^i$, $i \in \{1, \ldots, n_T\}$, we define the modified Bézier domain points $\widetilde{\mathbf{b}}^i_\ell$, $\ell = 1, \ldots, 10$, in barycentric coordinates as:

$$\begin{aligned}
&\widetilde{\mathbf{b}}^i_1 = (1,0,0)\,, && \widetilde{\mathbf{b}}^i_2 = \Big(1-\frac{\varepsilon}{3}, \frac{\varepsilon}{3}, 0\Big)\,, && \widetilde{\mathbf{b}}^i_3 = \Big(1-\frac{\varepsilon}{3}, 0, \frac{\varepsilon}{3}\Big)\,, && \widetilde{\mathbf{b}}^i_4 = \Big(\frac{\varepsilon}{3}, 1-\frac{\varepsilon}{3}, 0\Big)\,,\\
&\widetilde{\mathbf{b}}^i_5 = \Big(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\Big)\,, && \widetilde{\mathbf{b}}^i_6 = \Big(\frac{\varepsilon}{3}, 0, 1-\frac{\varepsilon}{3}\Big)\,, && \widetilde{\mathbf{b}}^i_7 = (0,1,0)\,, && \widetilde{\mathbf{b}}^i_8 = \Big(0, 1-\frac{\varepsilon}{3}, \frac{\varepsilon}{3}\Big)\,,\\
& && \widetilde{\mathbf{b}}^i_9 = \Big(0, \frac{\varepsilon}{3}, 1-\frac{\varepsilon}{3}\Big)\,, && \widetilde{\mathbf{b}}^i_{10} = (0,0,1)\,.
\end{aligned} \tag{7.21}$$

Note that the standard Bézier domain points are a special case with $\varepsilon = 1$. The smaller the value of the parameter ε, the closer the points will be to the vertices of the triangle. Figure 7.6-(b) shows the positioning of these points for $\varepsilon = 1/2$. The idea here is to build a single cubic polynomial defined over the neighborhood of each vertex or edge of the triangulation obtained through weighted least squares, where the weights are defined by measuring the distance of each modified Bézier domain point to the target vertex or edge.

More precisely, let

$$\widetilde{z}^i_\ell = f(\widetilde{\mathbf{b}}^i_\ell), \quad \ell = 1, \ldots, 10, \quad i = 1, \ldots, n_T,$$

be the known values of a given function f. As in Section 7.4.3, we build different polynomials related to the features of the considered triangulation, namely its vertices, edges, and faces. Let $\mathbf{v}^k$, $k \in \{1, \ldots, n_V\}$, be a vertex of $\mathcal{T}$ and denote by $|\mathbf{v}^k|$ the number of triangles attached to $\mathbf{v}^k$. We define the weighted vertex polynomial $\mathcal{V}_\varepsilon^k f$ as

$$\mathcal{V}_\varepsilon^k f = \sum_{0 \le \alpha+\beta \le 3} c_{\alpha,\beta}^k x^\alpha y^\beta, \tag{7.22}$$

where $\mathbf{c}^k = (c_{0,0}^k, c_{0,1}^k \ldots, c_{3,0}^k)^T$ is obtained by solving the overdetermined linear system

$$W_k \, \widetilde{V}_k \, \mathbf{c}^k = W_k \, \widetilde{\mathbf{z}}^k$$

in least-squares sense. Here $\widetilde{V}_k \in \mathbb{R}^{10|\mathbf{v}^k| \times 10}$ is the Vandermonde matrix in terms of the monomial basis of $\mathbb{P}_3$ evaluated at all the modified Bézier domain points of the triangles sharing $\mathbf{v}^k$, and $\widetilde{\mathbf{z}}^k$ is the vector containing the values

$$\widetilde{\mathbf{z}}^k = \{\widetilde{z}_\ell^i, \ i : \mathbf{v}^k \in \triangle^i, \ \ell = 1, \ldots, 10\}.$$

Moreover, $W_k \in \mathbb{R}^{10|\mathbf{v}^k| \times 10|\mathbf{v}^k|}$ is the diagonal matrix containing the weights corresponding to the points in (7.21). More precisely, for a given point $\widetilde{\mathbf{b}}_\ell^i$ that belongs to a triangle $\triangle^i$ attached to $\mathbf{v}^k$, its weight $\omega(\widetilde{\mathbf{b}}_\ell^i; \mathbf{v}^k)$ is chosen as

$$\omega(\widetilde{\mathbf{b}}_\ell^i; \mathbf{v}^k) = \frac{1}{1 + 10 \left(\frac{\mathrm{dist}(\widetilde{\mathbf{b}}_\ell^i, \mathbf{v}^k)}{h_{\triangle^i}} \right)^2}, \tag{7.23}$$

where $\mathrm{dist}(\widetilde{\mathbf{b}}_\ell^i, \mathbf{v}^k) = \|\widetilde{\mathbf{b}}_\ell^i - \mathbf{v}^k\|_2$ is the standard Eucledian distance and $h_{\triangle^i}$ is the length of the longest edge of $\triangle^i$. A similar procedure can be followed to define the weighted edge polynomial $\mathcal{E}_\varepsilon^j f$ attached to the edge $\mathbf{e}^j$ of $\mathcal{T}$, $j \in \{1, \ldots, n_E\}$,

$$\mathcal{E}_\varepsilon^j f = \sum_{0 \le \alpha+\beta \le 3} c_{\alpha,\beta}^j x^\alpha y^\beta, \tag{7.24}$$

where $\mathbf{c}^j = (c_{0,0}^j, c_{0,1}^j \ldots, c_{3,0}^j)^T$ is determined by solving the overdetermined linear system

$$W_j \, \widetilde{V}_j \, \mathbf{c}^j = W_j \, \widetilde{\mathbf{z}}^j$$

in least-squares sense. Here $\widetilde{V}_j \in \mathbb{R}^{20 \times 10}$ is the Vandermonde matrix in terms of the monomial basis of $\mathbb{P}_3$ evaluated at all the modified Bézier domain points of the triangles sharing $\mathbf{e}^j$, and $\widetilde{\mathbf{z}}^j$ is the vector containing the values

$$\widetilde{\mathbf{z}}^j = \{\widetilde{z}_\ell^i, \ i : \mathbf{e}^j \in \triangle^i, \ \ell = 1, \ldots, 10\}.$$

The weights in the diagonal matrix $W_j \in \mathbb{R}^{20 \times 20}$ are calculated similar to (7.23). For a given point $\widetilde{\mathbf{b}}_\ell^i$ that belongs to a triangle $\triangle^i$ attached to $\mathbf{e}^j$, we choose the weight $\omega(\widetilde{\mathbf{b}}_\ell^i; \mathbf{e}^j)$ as

$$\omega(\widetilde{\mathbf{b}}_\ell^i; \mathbf{e}^j) = \frac{1}{1 + 10 \left(\frac{\mathrm{dist}(\widetilde{\mathbf{b}}_\ell^i, \mathbf{e}^j)}{h_{\triangle^i}} \right)^2}.$$

Lastly, the triangle (or face) polynomial $\mathcal{F}_\varepsilon^i f$ attached to the triangle $\triangle^i$, $i \in \{1, \dots, n_T\}$,

$$\mathcal{F}_\varepsilon^i f = \sum_{0 \le \alpha+\beta \le 3} \widetilde{c}_{\alpha,\beta}^i x^\alpha y^\beta, \tag{7.25}$$

is defined as the cubic polynomial interpolating at the points in (7.21), which can be built in a manner similar to (7.16). Figure 7.7-(b) illustrates the positioning of the above weighted polynomials defined in (7.22), (7.24), and (7.25). These polynomials can take the same role as the averaged polynomials introduced in Section 7.4.3 and can be used to recover the Hermite values involved in Corollary 7.2.2.

The new quasi-interpolant $\mathcal{Q}_{\text{wls}} f$ is built as follows. Consider the triangle $\triangle^i$ of $\mathcal{T}$, $i \in \{1, \dots, n_T\}$. With reference to (7.8), from the weighted vertex polynomials $\mathcal{V}_\varepsilon^k f$ in (7.22) we sample the values $f_{k,\alpha,\beta}^i$; from the weighted edge polynomials $\mathcal{E}_\varepsilon^j f$ in (7.24) we sample the values g_k^i and $g_{k,l}^i$; and finally, the value h_0^i is sampled from the triangle polynomial $\mathcal{F}_\varepsilon^i f$ in (7.25). Given these Hermite data we follow the same construction as in Section 7.4.2. More precisely, applying the conversion strategy explained in Section 7.4.1, we find the values

$$\mathfrak{f}_{\text{wls}}^i = (\mathfrak{f}_{\text{wls},1}^i, \dots, \mathfrak{f}_{\text{wls},28}^i)^T,$$

from which we compute the coefficients

$$\boldsymbol{\mu}_{\text{wls}}^i(f) = (\mu_{\text{wls},1}^i(f), \dots, \mu_{\text{wls},28}^i(f))^T$$

by solving

$$B\, \boldsymbol{\mu}_{\text{wls}}^i(f) = \mathfrak{f}_{\text{wls}}^i,$$

with B as in (7.11). The quasi-interpolant $\mathcal{Q}_{\text{wls}} : C^0(\Omega) \longrightarrow \mathbb{S}_3^2(\mathcal{T}_{\text{WS}_3})$ is then identified by

$$\mathcal{Q}_{\text{wls}} f \Big|_{\triangle^i} = \sum_{j=1}^{28} \mu_{\text{wls},j}^i(f) B_j^i, \quad i = 1, \dots, n_T. \tag{7.26}$$

Once more, by construction, the quasi-interpolant $\mathcal{Q}_{\text{ave}} f$ enjoys the global smoothness, cubic polynomial reproduction, and locality properties stated at the beginning of Section 7.4.

7.4.5 Error estimates

In this section we analyze the approximation order of the proposed quasi-interpolants to a sufficiently smooth function f. To this end, we recall a classical approximation result about polynomials on triangles; see, e.g., [LS07, Theorem 1.3]. Given a triangle $\triangle$ with maximal edge length $h_\triangle$, for every $f \in C^{d+1}(\triangle)$ there exists a polynomial $p_d \in \mathbb{P}_d$ such that

$$\|D^k(f - p_d)\|_{\infty,\triangle} \le K_d\, h_\triangle^{d+1-k}\, \|D^{d+1} f\|_{\infty,\triangle}, \tag{7.27}$$

for all $0 \le k \le d$. The constant K_d depends only on d. Note that a similar estimate in L^q-norm can be shown for any function f belonging to the Sobolev space $W^{d+1,q}(\triangle)$; see [LS07].

Proposition 7.4.3. *Given a function $f \in C^4(\Omega)$, the quasi-interpolant in (7.13) satisfies*

$$\|f - \mathcal{Q}_{\text{exa}} f\|_{\infty,\Omega} \le K_{\text{exa}}\, h_{\mathcal{T}}^4\, \|D^4 f\|_{\infty,\Omega}, \tag{7.28}$$

where the constant K_{exa} is independent of f and $h_{\mathcal{T}}$.

Proof. Consider a triangle $\triangle^i$ of $\mathcal{T}$. Then, using a triangle inequality,

$$\|f - \mathcal{Q}_{\text{exa}} f\|_{\infty,\triangle^i} \leq \|f - p_3\|_{\infty,\triangle^i} + \|p_3 - \mathcal{Q}_{\text{exa}} f\|_{\infty,\triangle^i},$$

where p_3 is the same polynomial as in (7.27) for $d = 3$. Moreover, the polynomial reproduction property of the quasi-interpolant implies $\|p_3 - \mathcal{Q}_{\text{exa}} f\|_{\infty,\triangle^i} = \|\mathcal{Q}_{\text{exa}}(p_3 - f)\|_{\infty,\triangle^i}$ and from Proposition 7.4.2 we deduce

$$\|p_3 - \mathcal{Q}_{\text{exa}} f\|_{\infty,\triangle^i} \leq 5C \left(\|f - p_3\|_{\infty,\triangle^i} + h_{\mathcal{T}} \|D(f - p_3)\|_{\infty,\triangle^i} + h_{\mathcal{T}}^2 \|D^2(f - p_3)\|_{\infty,\triangle^i} \right).$$

Combining the above estimates with (7.27) results in (7.28), where $K_{\text{exa}} = K_3(1 + 15C)$. □

Similar approximation estimates can be derived for the quasi-interpolants $\mathcal{Q}_{\text{ave}} f$ in (7.20) and $\mathcal{Q}_{\text{wls}} f$ in (7.26) as well.

7.5 Numerical experiments

This section contains several numerical experiments illustrating the quality of the proposed quasi-interpolants $\mathcal{Q}_{\text{exa}} f$, $\mathcal{Q}_{\text{ave}} f$, and $\mathcal{Q}_{\text{wls}} f$, described in Section 7.4. We start by applying our constructions to a degree four polynomial and the well-known Franke's function, both defined on a square domain. We consider a sequence of nested tridirectional meshes to verify the approximation order of the quasi-interpolants (see Section 7.4.5). Next, we apply our constructions to a sigmoid function and a function with three peaks, again defined on a square domain, but now we use adaptively refined triangular meshes in order to capture more accurately the local features. Lastly, we present an example dealing with a non-square domain. In all the examples we take the value $\varepsilon = 1/2$ in (7.21) for the construction of $\mathcal{Q}_{\text{wls}} f$.

Being able to evaluate any spline $s \in \mathbb{S}_3^2(\triangle_{\text{WS}_3})$ represented in the basis (7.1) is an important aspect in our constructions. Since the basis functions are (scaled) simplex splines, they could be evaluated through recurrence or simply by means of a lookup-table process; see [LMS22, Section 5.1] and in particular [LMS22, Tables 1 and 2] for their explicit expressions. Any point $\mathbf{p} \in \triangle$ can be uniquely characterized and searched via a Boolean vector. Indeed, thanks to the cross-cut nature of the WS_3 split, each of the 75 polygonal regions in $\triangle_{\text{WS}_3}$ is uniquely identified by the sign of the linear equations defining the 18 interior lines of the split; see [LMS22, Eq. (28)]. Hence, in order to detect in which polygonal region of the macro-triangle a given point $\mathbf{p}$ belongs to, we only need to evaluate all the 18 interior lines at $\mathbf{p}$ and to collect the resulting signs in a Boolean vector.

7.5.1 Polynomial of degree four

We start our numerical investigation considering as given function a polynomial of degree four. From Section 7.4.5 we expect that the L^∞ norm of error we commit when approximating a polynomial of degree four with our three QIs decreases as $O(h_{\mathcal{T}}^4)$, with $h_{\mathcal{T}}$ the maximum length of the macroelements in the triangulation. In order to demonstrate the optimality of our proposed approximations, we consider five nested tridirectional meshes in $[0, 1]^2$ with axis-aligned edge lengths from 2^0 to 2^{-4}; this experiment has been developed by using the bivariate quartic polynomial $f \in \mathbb{P}_4$

$$f(x, y) = \frac{1}{4} x^4 + x^3 y. \tag{7.29}$$

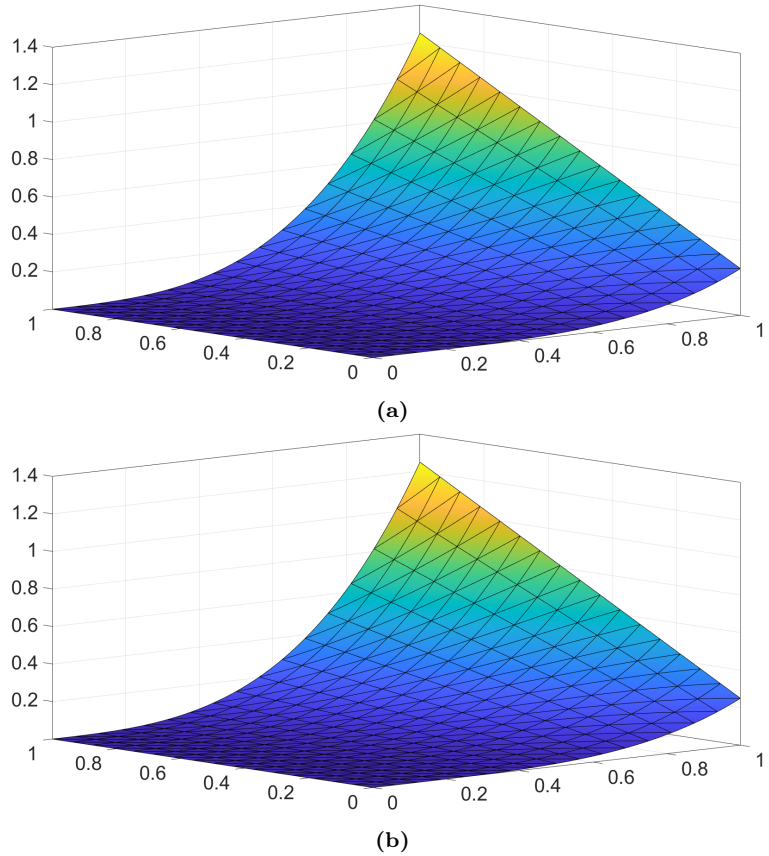


Figure 7.8. (a): plot of the polynomial $f(x, y)$ in (7.29). (b): plot of $\mathcal{Q}_{\text{ave}} f$.

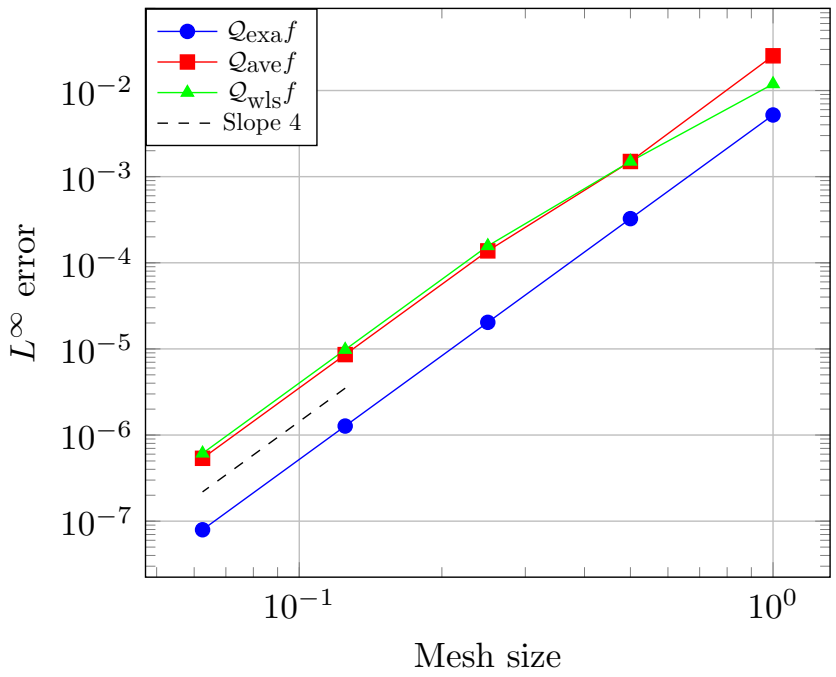


Figure 7.9. Convergence plot in logarithmic scale of the L^∞ error for the quasi-interpolants computed from the polynomial f in (7.29) over the nested meshes in Figure 7.10.

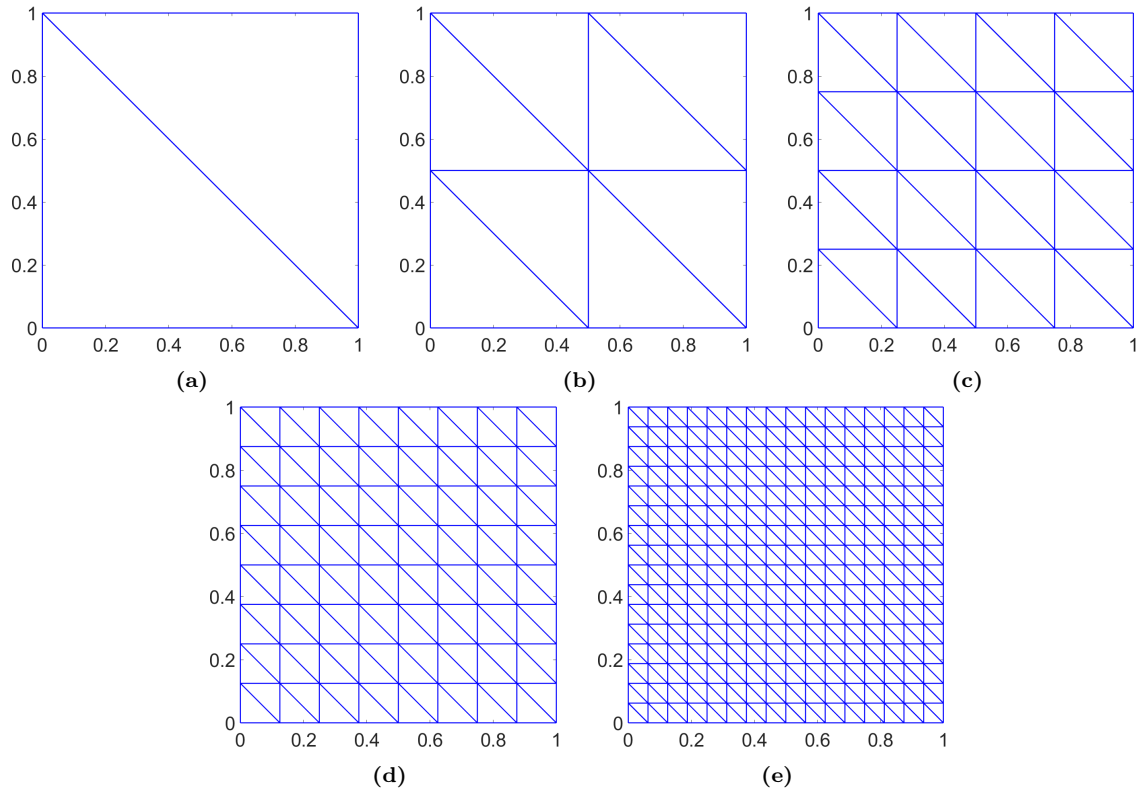


Figure 7.10. From (a) to (e): triangulations of the domain $[0,1]^2$ with axis-aligned edge lengths 2^{-i}, $i = 0,\ldots,4$ involved in the examples in Sections 7.5.1 and 7.5.2.

In Figure 7.8 are presented, respectively, the graph of the polynomial f and its corresponding quasi-interpolant $\mathcal{Q}_{\text{ave}}f$, while Figure 7.10 contains the 5 nested meshes we used to test the optimal convergence rate shown in Figure 7.9. As we can notice from the convergence plot, for the polynomial case the errors obtained with the averaging and weighted least-squares are almost the same, while with the exact data the error is smaller.

7.5.2 Franke's function

As first non-polynomial example we consider the well-known Franke's function, which is a standard test in approximation methods. Franke's function presents two Gaussian peaks of different heights and a smaller dip; it is analytically defined as:

$$f(x,y) = \frac{3}{4}e^{-\frac{(9x-2)^2+(9y-2)^2}{4}} + \frac{3}{4}e^{-\frac{(9x+12)^2}{49}-\frac{9y+1}{10}} + \frac{1}{2}e^{-\frac{(9x-7)^2+(9y-3)^2}{4}} - \frac{1}{5}e^{-((9x-4)^2+(9y-7)^2)}. \tag{7.30}$$

Imitating Section 7.5.1, we test the error convergence of our proposed constructions over tridirectional meshes in $[0,1]^2$ with axis-aligned edge lengths 2^{-i}, $i = 0,\ldots,4$ (see Figure 7.10). Figure 7.11 presents the graph of the Franke's function and its corresponding quasi-interpolant $\mathcal{Q}_{\text{ave}}f$. In this case, we notice from the error plot in Figure 7.12 that the quasi-interpolant $\mathcal{Q}_{\text{wls}}f$ returns smaller errors than $\mathcal{Q}_{\text{ave}}f$, when the size of the mesh becomes small enough; the construction with exact data $\mathcal{Q}_{\text{exa}}f$ returns the most accurate representation. The optimal convergence rate is once again achieved.

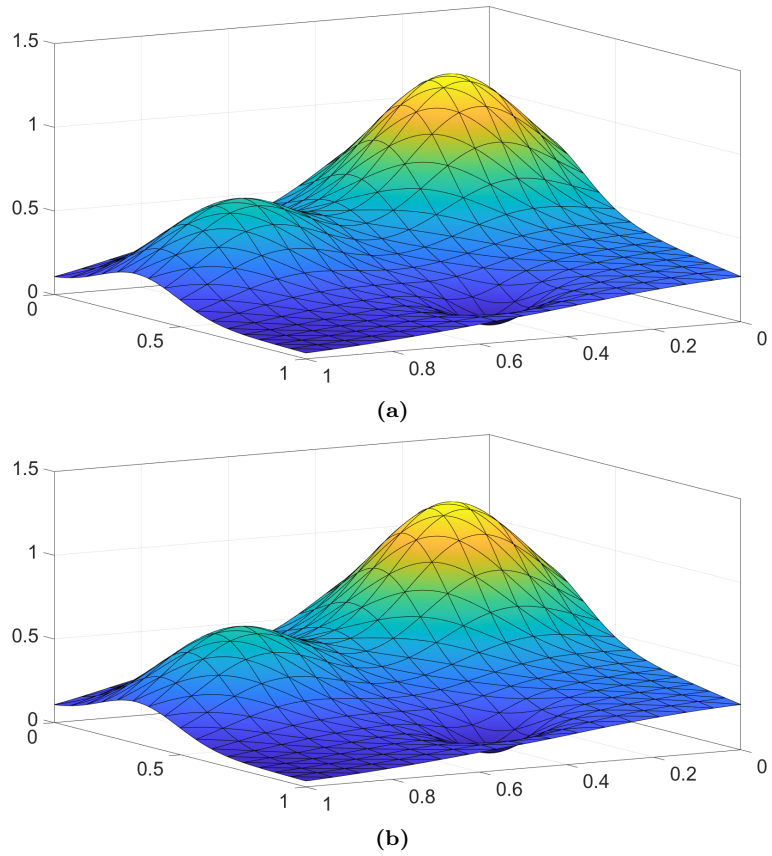


Figure 7.11. (a): graph of the Franke's function in (7.30). (b): plot of $\mathcal{Q}_{\text{ave}} f$.

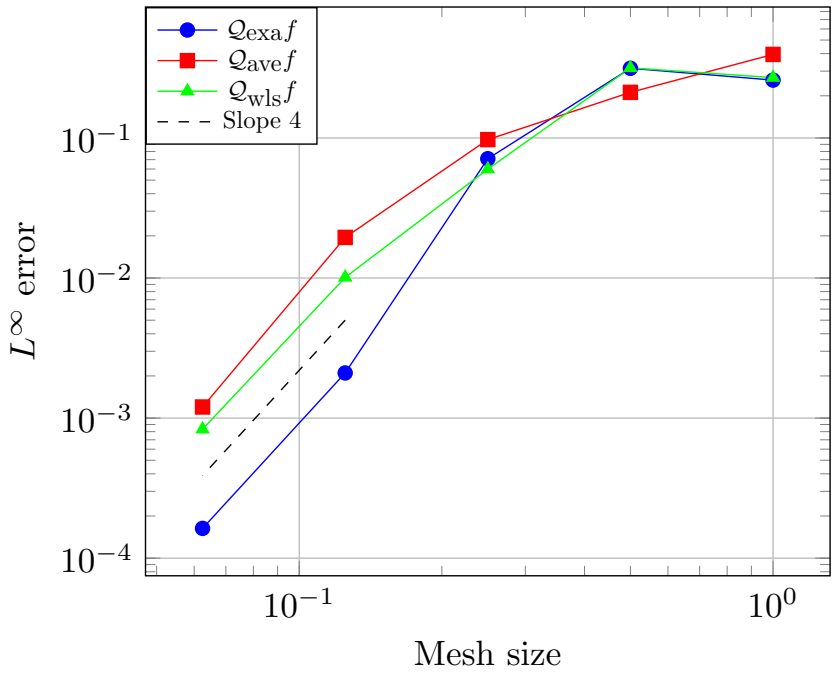


Figure 7.12. Error plot in logarithmic scale of the L^∞ error for the quasi-interpolants computed from the Franke's function f in (7.30) over the nested meshes in Figure 7.10

7.5.3 Sigmoid function

Let introduce the sigmoid function f whose explicit equation is

$$f(x,y) = \frac{\tanh(9y - 9x) + 1}{9}. \tag{7.31}$$

We restrict its definition to the squared domain $[-1,1]^2$. It easy to notice that in the region neighboring the line of equation $y = x$ the sigmoid presents a very steep jump between its maximum and minimum value. Hence, in order to construct precise quasi-interpolants preserving this feature, we will use in this example non-uniform triangulations which are finer, at every step, in the regions of the domain close to the slope. The graph of the sigmoid function and its $\mathcal{Q}_{\text{exa}}f$ quasi-interpolant are contained in Figure 7.13, while Figure 7.14 shows these adaptative triangulations. Figure 7.15 presents the error plot for this investigation. Similarly to the previous example, the best approximation is achieved with the quasi-interpolant $\mathcal{Q}_{\text{exa}}f$, while when dealing with recreated Hermit data the $\mathcal{Q}_{\text{wls}}f$ quasi-interpolant does a better job.

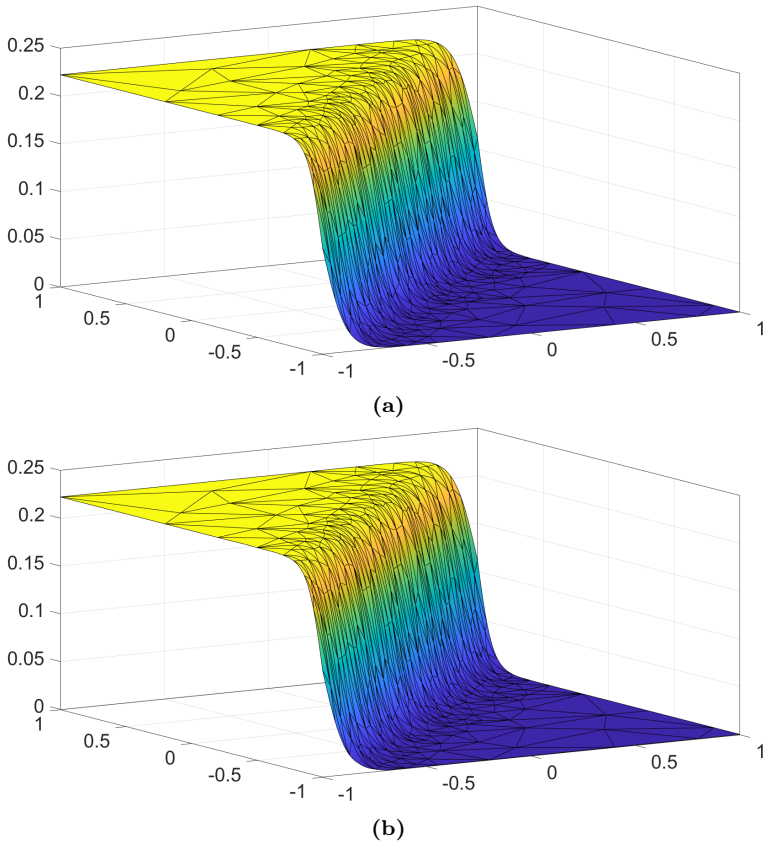


Figure 7.13. (a): plot of the sigmoid in (7.31). (b): plot of $\mathcal{Q}_{\text{ave}}f$.

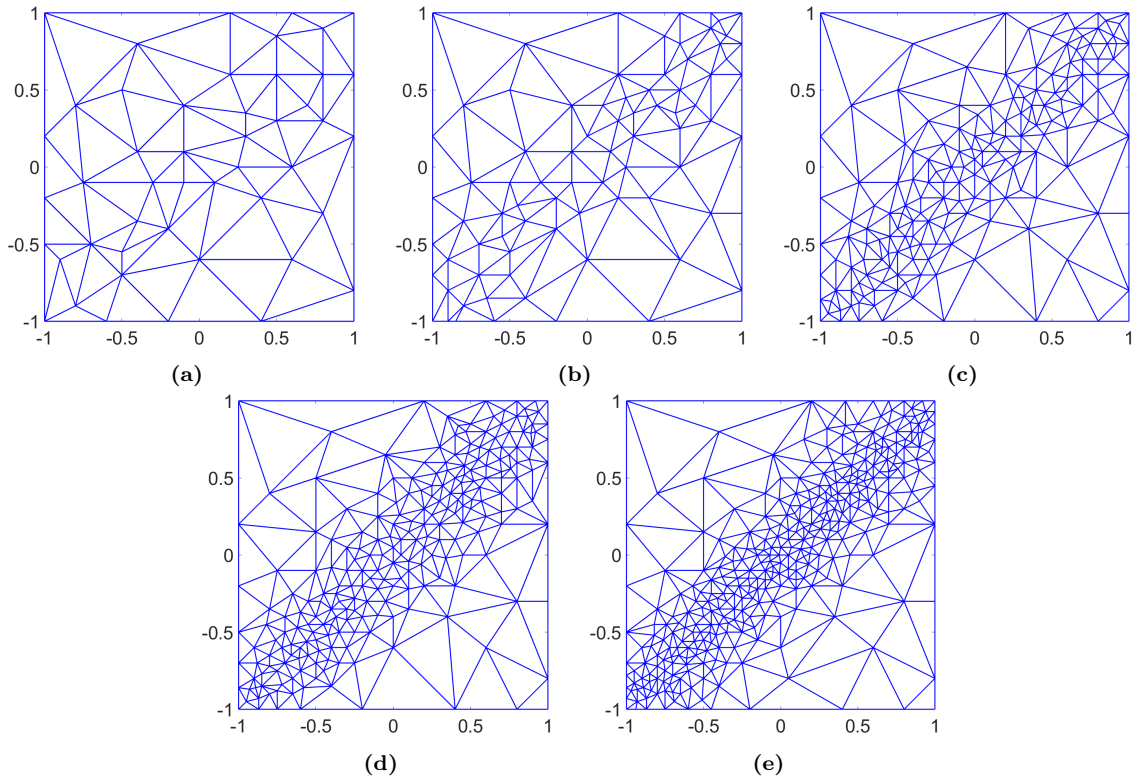


Figure 7.14. From (a) to (e): adaptative triangulations of the domain $[-1,1]^2$ utilized for the construction of $\mathcal{Q}_{\text{exa}}f$, $\mathcal{Q}_{\text{ave}}f$ and $\mathcal{Q}_{\text{wls}}f$ in the example in Section 7.5.3.

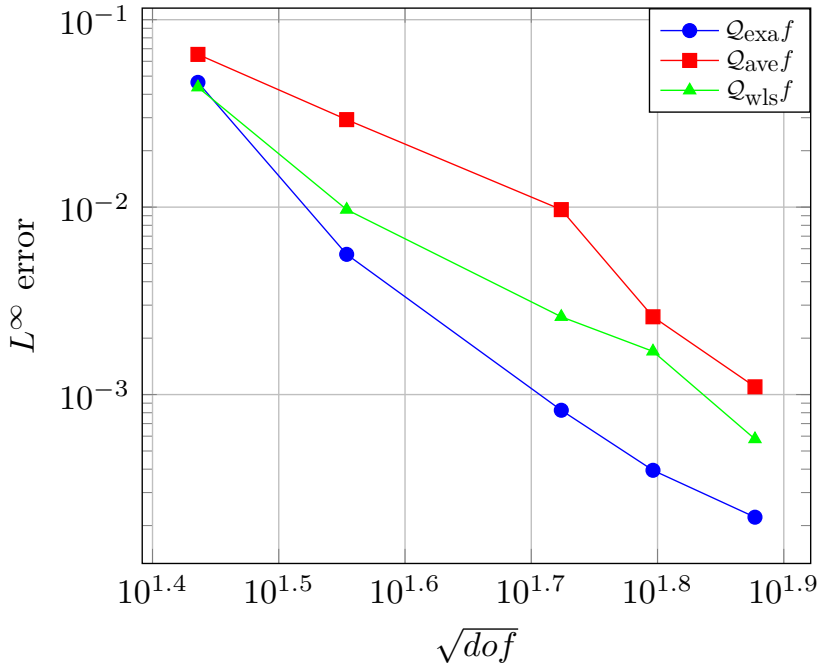


Figure 7.15. Error plot in logarithmic scale of the L^∞ error for the quasi-interpolants of the sigmoid function (7.31) over the adaptative meshes in Figure 7.14.

7.5.4 Function with three peaks

In real applications it is common to deal with functions which presents some discontinuities: in these situations, most of the usual numerical tools have to be adapted in order to treat them. Here we show that our construction works perfectly also in this challenging setting without any ah-hoc tuning. Let f be the function defined as

$$f(x,y) = \frac{2}{3e^{\sqrt{(10x-3)^2+(10y-3)^2}}} + \frac{2}{3e^{\sqrt{(10x)^2+(10y)^2}}} + \frac{2}{3e^{\sqrt{(10x+3)^2+(10y+3)^2}}}. \tag{7.32}$$

From standard computation it is easy to notice that the function f presents three cusps in correspondence of the pairs $(-3,-3),(0,0),(3,3)$ for x and y, respectively. Thus, in these points its first (and second) derivative will result discontinuous; as consequence, in this experiment we won't be able to construct the quasi-interpolant $\mathcal{Q}_{\text{exa}}f$, while there are no problems for the other two approaches $\mathcal{Q}_{\text{ave}}f$ and $\mathcal{Q}_{\text{wls}}f$. Furthermore, an interesting aspect of our construction is to regularize discontinuous functions with faithful C^2 approximations. Figure 7.16-(a) shows the plot of the function f defined in (7.32), while Figure 7.16-(b) the graph of its QI $\mathcal{Q}_{\text{ave}}f$. Due to the presence of the peaks we decided to use adaptative meshes which present an increasing number of triangles nearby the regions of the domain where the irregularities are; Figure 7.17 shows these locally refined triangulations while in Figure 7.18 is illustrated the behavior of the L^∞ error for $\mathcal{Q}_{\text{ave}}f$ and $\mathcal{Q}_{\text{wls}}f$. Once again, we notice that the approximation achieved with Hermite data from local weighted least-square polynomials produces a smaller error.

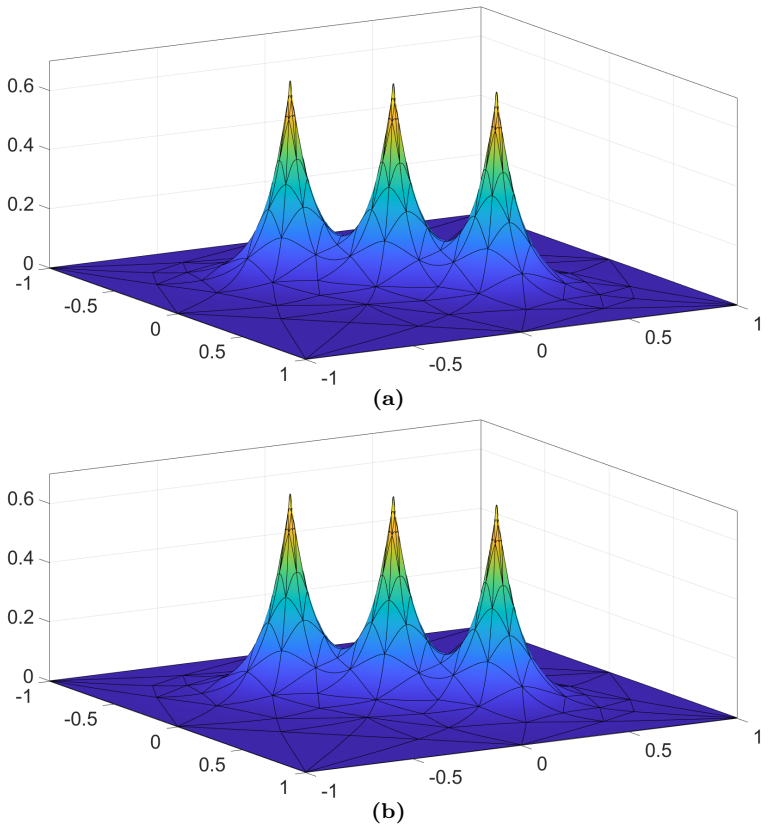


Figure 7.16. (a): graph of the function in (7.32). (b): plot of $\mathcal{Q}_{\text{ave}}f$.

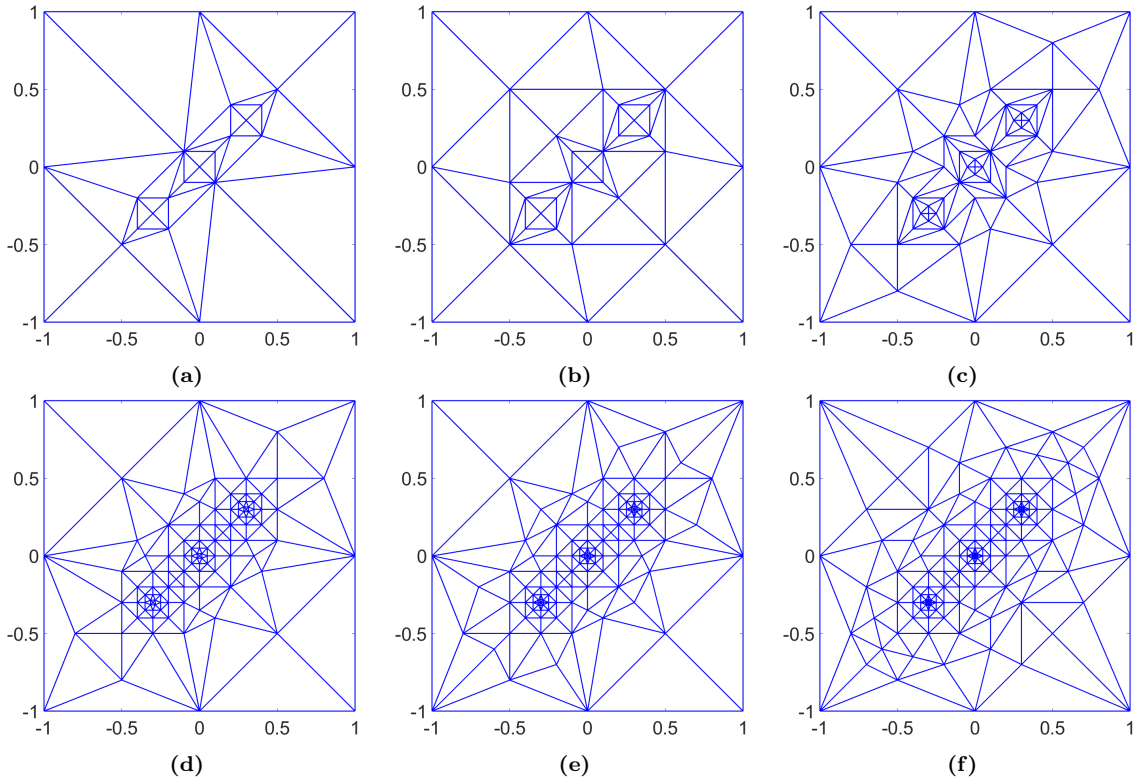


Figure 7.17. From (a) to (f): adaptative triangulations of the domain $[-1,1]^2$ used in the construction of $\mathcal{Q}_{\text{ave}}f$ and $\mathcal{Q}_{\text{wls}}f$ in the example in Section 7.5.4.

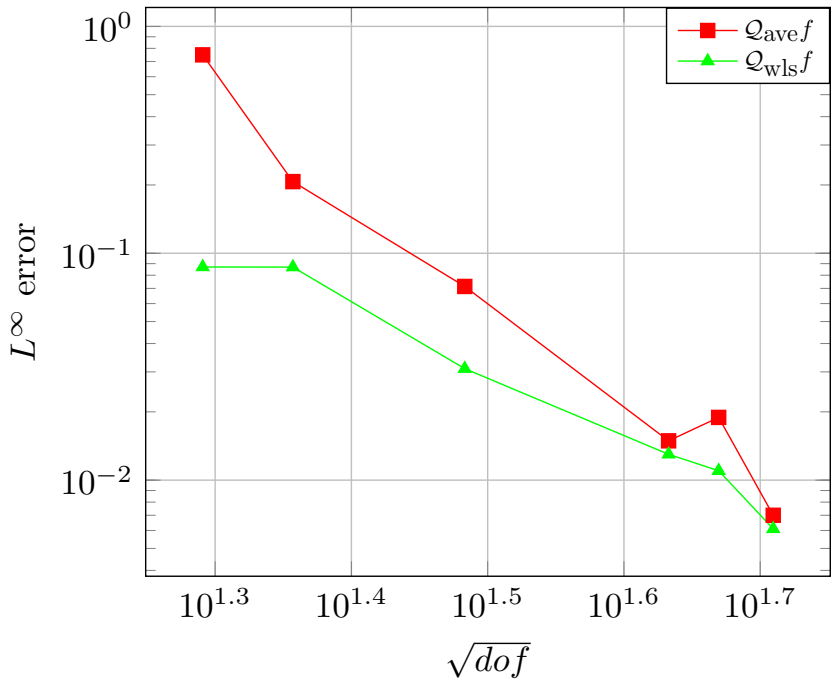


Figure 7.18. Error plot in logarithmic scale of the L^∞ error for the two quasi-interpolants $\mathcal{Q}_{\text{ave}}f$ and $\mathcal{Q}_{\text{wls}}f$ of the function in (7.32) over the adaptative meshes in Figure 7.17.

7.5.5 Function on a pentagon domain

In the previous examples we set our approximation problem over triangulations of a squared domain; moreover, this is not a restrictive assumption. In fact, another key point of our construction is its generality over arbitrary triangular meshes. In the following experiment we consider the polygonal domain $\pentagon \subset [0,1]^2$ identified by the set of vertices

$$\{(0.2,0),(0,0.55),(0.5,1),(1,0.55),(0.8,0)\} \tag{7.33}$$

on which we define a function f that, for a matter of graphic convenience, we take such that it has zero value along the boundaries of $\pentagon$. Let $r_i(x,y) = 0$, $i = 1,\dots,5$ be the implicit equations of the five lines identifying the pentagonal domain (7.33); so we have the polynomial

$$q(x,y) = \prod_{i=1}^{5} r_i(x,y),$$

which is used to define the objective function f of our investigation

$$f(x,y) = e^{-\left((x-0.5)^2+(y-0.45)^2\right)} q(x,y). \tag{7.34}$$

Figure 7.19 shows the plot of the function f and its quasi-interpolant $\mathcal{Q}_{\text{ave}} f$, respectively, while in Figure 7.20 are reported the consequent triangulations of the domain $\pentagon$ which have been used to investigate the behavior of the L^∞ error presented in Figure 7.21.

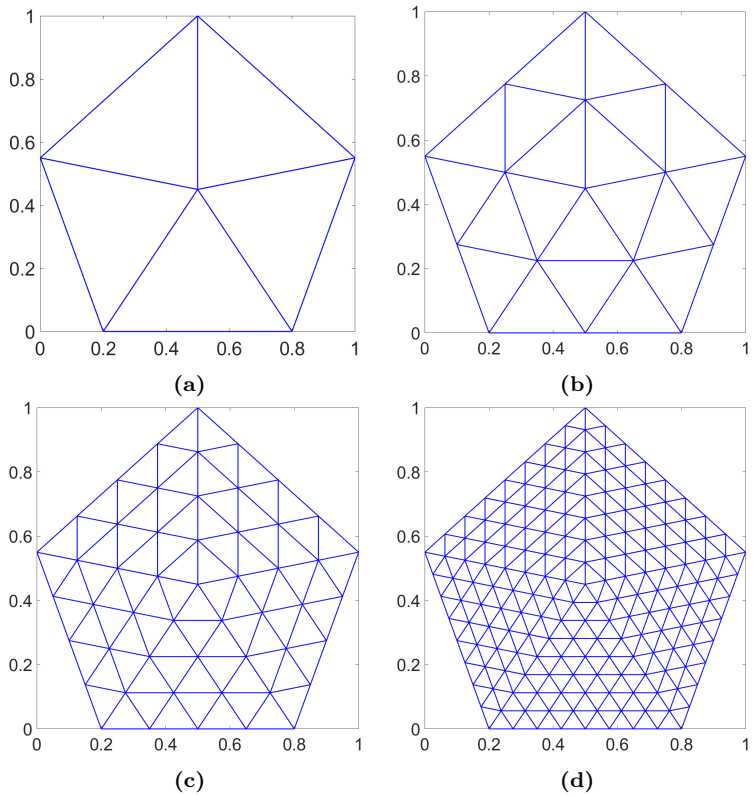


Figure 7.20. From (a) to (d): successive triangulations of the pentagonal domain $\pentagon \subset [0,1]^2$ used in the example in Section 7.5.5 for the assembly of $\mathcal{Q}_{\text{exa}} f$, $\mathcal{Q}_{\text{ave}} f$ and $\mathcal{Q}_{\text{wls}} f$.

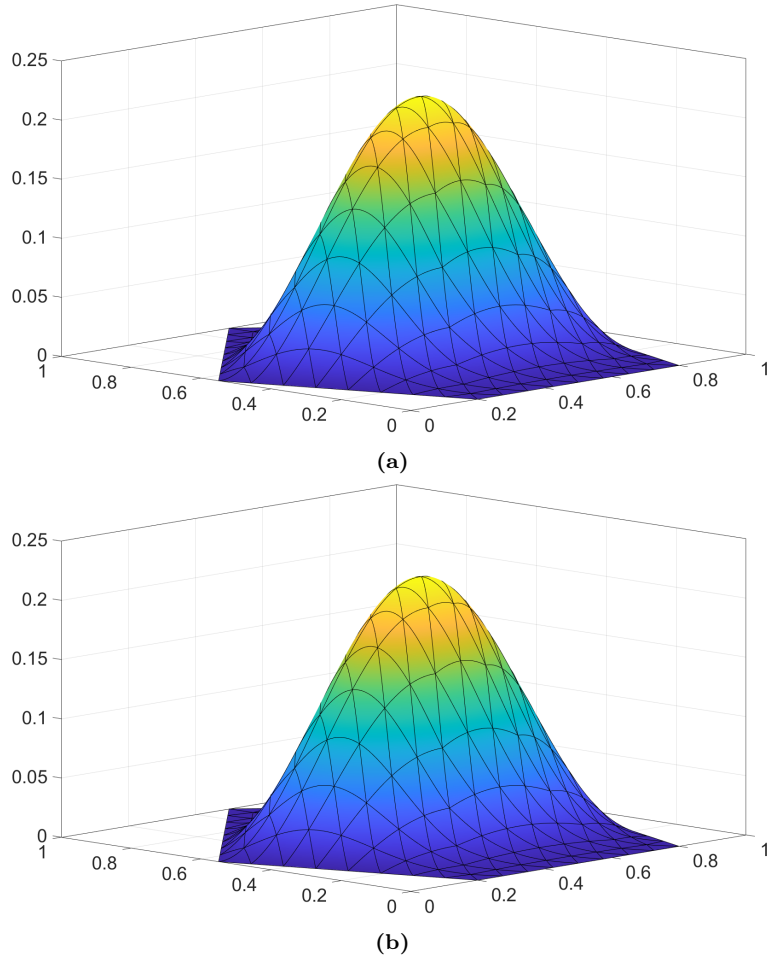


Figure 7.19. (a): plot of the function in (7.34). (b): plot of $\mathcal{Q}_{\text{ave}} f$.

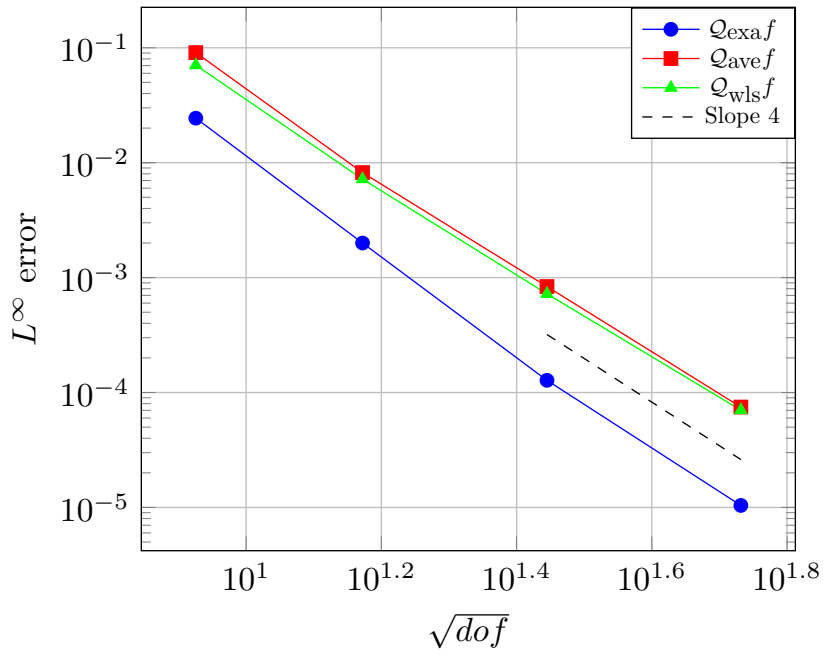


Figure 7.21. Error plot in logarithmic scale of the L^∞ error for the quasi-interpolants of the function defined in (7.34) over the meshes in Figure 7.20.

The quasi-interpolant $\mathcal{Q}_{\text{wls}}f$, among the two defined from reconstructed Hermite data, shows a slightly smaller approximation error. We reach again the expected rate of convergence.

Summary

In this chapter we provided the construction of three cubic C^2 quasi-interpolants over arbitrary triangulations. The quasi-interpolants are locally spanned by a simplex spline basis defined on cubic Wang-Shi refinement of the polygonal domain, making the whole construction very elegant. Thanks to their locality, we are able to compute the coefficients defining each quasi-interpolant just solving a small linear system whose right-hand side contains proper Hermite data ensuring the global C^2 continuity. If not given as input, we presented two different techniques to reconstruct consistent Hermite informations from local cubic polynomials defined thanks to a set of known values of a given function. Several numerical experiments are provided to state the actual approximation quality of the proposed quasi-interpolants.

Conclusion and perspectives

In this thesis, we have investigated new smooth spline structures suitable for modelling shapes, as well for approximation problems and numerical simulations.

We began in Chapter 2 presenting the construction of a G^1 family of surfaces, described by mean of smoothing masks, having the property of approximating the well-known Catmull-Clark subdivision surface. The coefficients of the above masks are explicitly defined through explicit formulas generated from the equations ensuring the geometric continuity between Bézier patches. The solving procedure led to multiple solutions which have been individually analyzed in order to select the one returning the smoothest result, as has been proven in the several proposed experiments.

Then we moved to the investigation of a set of basis function generating the G^1 spline space introduced in Chapter 2. Chapter 3 was devoted to the creation of such functions through a constructive procedure together with a deep analysis of their properties and a dimension formula for the space they generate which only depends on the combinatorial features of the mesh. We used these objects to solve point cloud fitting problems, whose high-quality result is provided by the various numerical experiment we proposed.

Chapter 4 was dedicated to the investigation of analysis-suitable basis functions for IGA simulations. This construction is essentially led by mimicking the building process made for the bases in Chapter 3, but generalizing the previous definitions for more general knot vectors. Also in this case a closed formula for the spline space dimension has been given. The suitability of the proposed bases in numerical IGA simulations of PDEs is shown in some classical tests.

Chapters 5 and 6 presented two interesting applications of the novel constructions in Chapters 2 to 4. The first one concerned the conversion, via point cloud fitting, of CAD models into smooth spline representations. We proposed a complete pipeline which describes the several steps of this translations combined with practical examples. Moreover, this approach turns CAD geometries (that are not appropriate for simulations) to analysis-suitable structures. Equally interesting, the second application focused on the solution of the shallow-water equation when dealing with lakes. Due to its strong dependence to the bathymetry conformation, a crucial point for a realistic study is to gain a precise representation of it: in our case, it is obtained via fitting a cloud describing a real lake seabed with our technique, to which follows the numerical resolution of the equation.

Finally, Chapter 7 was centered on the construction of cubic C^2 quasi-interpolant operators supported on a generic triangulation refined according to the cubic Wang-Shi split. The quasi-interpolation operators are computed by means of a local simplex spline basis which allows a simple derivation of coefficients for their representation in the simplex basis. Their approximation power and quality is confirmed by the examples we proposed which treat several challenging situations.

Several interesting perspectives can be considered. Concerning the construction of the G^1ACC surface, it might be generalized to patches of arbitrary degree; in fact, the choice to use bidegree 5 Bézier patches, in order to have sufficient degrees of freedom, can be substituted

with spline patches of any degree defined on an appropriate knot vector. Also interesting is the generalization to quad meshes presenting connected EVs. In this framework, the systems defining the geometric constraints around each vertex result to be interconnected with each other, making the solution of the equations more difficult. It would also be interesting to extend of the ACC scheme for the construction of G^2 surfaces; actually, starting from the G^1 scheme presented in this thesis, we believe that the G^2 smoothness can be reached imitating the solving strategy used for the tangent plane continuity, but applied to the proper equations governing the desired higher continuity. More challenging, but not impossible, is the extension of the G^1ACC construction to the volumetric case. It is not a straightforward task, since some theoretical tools are not well defined yet, as for example the definition of gluing data function across faces of a volumetric mesh and the compatibility condition around a vertex.

A similar outlook may be applied to the construction of the basis functions. More precisely, the definition of G^1 functions can be generalized to patches of arbitrary degree and knot vector as well as to topologies with connected EVs. Also the G^2 case is interesting to be studied and the more demanding volumetric generalization.

Also of interest is to develop a pipeline converting CAD objects to IGA suitable models that takes into account the features of the input geometry, since it is a sensitive aspect in practical applications.

Lastly, regarding the quasi-interpolation operators, a similar approach may be used to define quartic C^3 quasi-interpolants by using the simplex spline basis defined in [LMS24]. It is also interesting to investigate their application to the isogeometric analysis environment, e.g. in the definition of ad-hoc quadrature rules for triangular domains.

Bibliography

[Ali16] Z. Ali, J. Tyacke, P.G. Tucker, and S. Shahpar. "Block Topology Generation for Structured Multi-block Meshing with Hierarchical Geometry Handling". *Procedia Engineering* 26 (2016), pp. 212–224.

[AS02] P. Alfeld and L. L. Schumaker. "Smooth macro-elements based on Powell–Sabin triangle splits". *Advances in Computational Mathematics* 16 (2002), pp. 29–46.

[AS10] L.-E. Andersson and N.F. Stewart. *Introduction to the Mathematics of Subdivision Surfaces*. Society for Industrial and Applied Mathematics, 2010.

[Bar08] D. Barrera, M. J. Ibáñez, P. Sablonnière, and D. Sbibih. "Near-best univariate spline discrete quasi-interpolants on nonuniform partitions". *Constructive Approximation* 28 (2008), pp. 237–251.

[BC13] P.B. Bornemann and F. Cirak. "A subdivision-based implementation of the hierarchical b-spline finite element method". *Computer Methods in Applied Mechanics and Engineering* 253 (2013), pp. 584–598.

[Bei14] L. Beirão da Veiga, A. Buffa, G. Sangalli, and R. Vázquez. "Mathematical analysis of variational isogeometric methods". *Acta Numerica* 23 (2014), pp. 157–287.

[Ber12] S. Bernstein. "Démonstration du théorème de Weierstrass fondée sur le calcul des probabilités". *Communications of the Kharkov Mathematical Society* 13 (1912), pp. 1–2.

[Ber99] F. Bernardini, J Mittleman, H. Rushmeier, C. Silva, and G. Taubin. "The ball-pivoting algorithm for surface reconstruction". *IEEE Transactions on Visualization and Computer Graphics* 5.4 (1999), pp. 349–359.

[Bez17] J. Bezanson, A. Edelman, S. Karpinski, and V.B. Shah. "Julia: A fresh approach to numerical computing". *SIAM Review* 59.1 (2017), pp. 65–98.

[BG15] A. Buffa and C. Giannelli. "Adaptive isogeometric methods with hierarchical splines: Error estimator and convergence". *Mathematical Models and Methods in Applied Sciences* 26.01 (2015), pp. 1–25.

[BH14] G.-P. Bonneau and S. Hahmann. "Flexible G^1 interpolation of quad meshes". *Graphical Models* 76.6 (2014), pp. 669–681.

[BHR93] C. de Boor, K. Höllig, and S. Riemenschneider. *Box Splines*. Springer New York, 1993.

[BM17] M. Bercovier and T. Matskewich. *Smooth Bézier Surfaces over Unstructured Quadrilateral Meshes*. Springer, 2017.

[BMV17] A. Blidia, B. Mourrain, and N. Villamizar. "G^1-smooth splines on quad meshes with 4-split macro-patch elements". *Computer Aided Geometric Design* 52-53 (2017), pp. 106–125.

[BMX20] A. Blidia, B. Mourrain, and G. Xu. "Geometrically smooth spline bases for data fitting and simulation". *Computer Aided Geometric Design* 78 (2020), p. 101814.

[Boe84] W. Boehm. "Calculating with box splines". *Computer Aided Geometric Design* 1.2 (1984), pp. 149–162.

[Boo72] C. de Boor. "On calculating with B-splines". *Journal of Approximation Theory* 6.1 (1972), pp. 50–62.

[Boo78] C. de Boor. *A Practical Guide to Spline*. Springer, 1978.

[Bra16] C. Bracco, C. Giannelli, F. Mazzia, and A. Sestini. "Bivariate hierarchical Hermite spline quasi-interpolation". *BIT Numerical Mathematics* 56 (2016), pp. 1165–1188.

[Bra20] C. Bracco, C. Giannelli, M. Kapl, and R. Vázquez. "Isogeometric analysis with C^1 hierarchical functions on planar two-patch geometries". *Computers & Mathematics with Applications* 80.11 (2020), pp. 2538–2562.

[Bra23a] C. Bracco, C. Giannelli, M. Kapl, and R. Vázquez. "Adaptive isogeometric methods with C^1 (truncated) hierarchical splines on planar multi-patch domains" (2023). arXiv: 2204.10000.

[Bra23b] C. Bracco, C. Giannelli, A. Reali, M. Torre, and R. Vázquez. "Adaptive isogeometric phase-field modeling of the Cahn-Hilliard equation: Suitably graded hierarchical refinement and coarsening on multi-patch geometries" (2023). arXiv: 2306.07112.

[Bre11] H. Brezis. *Functional Analysis, Sobolev Spaces and Partial Differential Equations*. Springer, 2011.

[Bre13] A. Bressan. "Some properties of LR-splines". *Computer Aided Geometric Design* 30.8 (2013), pp. 778–794.

[BS08] S.C. Brenner and L.R. Scott. *The Mathematical Theory of Finite Element Methods*. Vol. 15. Texts in Applied Mathematics. Springer, 2008.

[BS16] A. Buffa and G. Sangalli, eds. *IsoGeometric Analysis: A New Paradigm in the Numerical Approximation of PDEs*. Vol. 2161. Lecture Notes in Mathematics. Springer International Publishing, 2016.

[Buf16] A. Buffa, E.M. Garau, C. Giannelli, and G. Sangalli. "On Quasi-Interpolation Operators in Spline Spaces". *Lecture Notes in Computational Science and Engineering*. Springer International Publishing, 2016, pp. 73–91.

[CAD] CADfix. https://www.iti-global.com/cadfix.

[CC78] E. Catmull and J. Clark. "Recursively generated B-spline surfaces on arbitrary topological meshes". *Computer-Aided Design* 10.6 (1978), pp. 350–355.

[CHB09] J. Cottrell, T. Hughes, and Y. Bazilevs. *Isogeometric analysis: toward integration of CAD and FEA*. John Wiley & sons, Ltd., 2009.

[CLR84] E. Cohen, T. Lyche, and R. Riesenfeld. "Discrete box splines and refinement algorithms". *Computer Aided Geometric Design* 1.2 (1984), pp. 131–148.

[CS66] H.B. Curry and I.J. Schoenberg. "On Pólya frequency functions IV: The fundamental spline functions and their limits". *Journal d'Analyse Mathématique* 17.1 (1966), pp. 71–107.

[CST16] A. Collin, G. Sangalli, and T. Takacs. "Analysis-suitable G^1 multi-patch parametrizations for C^1 isogeometric spaces". *Computer Aided Geometric Design* 47 (2016), pp. 93–113.

[CT65] R.W. Clough and J.L. Tocher. "Finite element stiffness matrices for analysis of plates in bending". *Proceedings of the Conference on Matrix Methods in Structural Mechanics*. Wright-Patterson Air Force Base, 1965, pp. 515–545.

[DLG90] N. Dyn, D. Levine, and J.A. Gregory. "A Butterfly Subdivision Scheme for Surface Interpolation with Tension Control". *ACM Trans. Graph.* 9.2 (1990), pp. 160–169.

[DLP13] T. Dokken, T. Lyche, and K.F. Pettersen. "Polynomial splines over locally refined box-partitions". *Computer Aided Geometric Design* 30.3 (2013), pp. 331–356.

[DQ15] L. Dedè and A. Quarteroni. "Isogeometric Analysis for second order Partial Differential Equations on surfaces". *Computer Methods in Applied Mechanics and Engineering* 284 (2015), pp. 807–834.

[DRS13] C. Dagnino, S. Remogna, and P. Sablonnière. "Error bounds on the approximation of functions and partial derivatives by quadratic spline quasi-interpolants on non-uniform criss-cross triangulations of a rectangular domain". *BIT Numerical Mathematics* 53 (2013), pp. 87–109.

[DS78] D. Doo and M. Sabin. "Behaviour of recursive division surfaces near extraordinary points". *Computer-Aided Design* 10.6 (1978), pp. 356–360.

[Eva18] J.A. Evans, M.A. Scott, K.M Shepherd, D.C. Thomas, and R. Vázquez. "Hierarchical B-spline complexes of discrete differential forms". *IMA Journal of Numerical Analysis* 40.1 (2018), pp. 422–473.

[Fal18] A. Falini, C. Giannelli, T. Kanduč, M.L. Sampoli, and A. Sestini. "An adaptive IgA-BEM with hierarchical B-splines based on quasi-interpolation quadrature schemes". *International Journal for Numerical Methods in Engineering* 117.10 (2018), pp. 1038–1058.

[Far23] A. Farahat, B. Jüttler, M. Kapl, and T. Takacs. "Isogeometric analysis with C^1-smooth functions over multi-patch surfaces". *Computer Methods in Applied Mechanics and Engineering* 403 (2023), p. 115706.

[Far79] G. Farin. "Subsplines über Dreiecken". PhD thesis. Braunschweig, FRG, 1979.

[Far82] G. Farin. "Visually C^2 cubic splines". *Computer-Aided Design* 14.3 (1982), pp. 137–139.

[Far86] G. Farin. "Triangular Bernstein-Bézier patches". *Computer Aided Geometric Design* 3.2 (1986), pp. 83–127.

[Fre71] P.O. Frederickson. "Triangular spline interpolation. Generalized triangular splines". *Math. Reports 6/70 and 7/71. Lakehead University* (1970/71).

[GH73] W.N. Gordon and C.A. Hall. "Construction of curvilinear coordinate systems and application to mesh generation". *International J. Num. Methods in Eng.* 7 (1973), pp. 461–477.

[GJM20] A. Giust, B. Jüttler, and A. Mantzaflaris. "Local (T)HB-spline projectors via restricted hierarchical spline fitting". *Computer Aided Geometric Design* 80 (2020), p. 101865.

[GJS12] C. Giannelli, B. Jüttler, and H. Speleers. "THB-splines: The truncated basis for hierarchical splines". *Computer Aided Geometric Design* 29.7 (2012), pp. 485–498.

[Gro24] J. Grošelj, M. Kapl, M. Knez, T. Takacs, and V. Vitrih. "C^1-smooth isogeometric spline functions of general degree over planar mixed meshes: The case of two quadratic mesh elements". *Applied Mathematics and Computation* 460 (2024), p. 128278.

[GS18] J. Grošelj and H. Speleers. "Three recipes for quasi-interpolation with cubic Powell–Sabin splines". *Computer Aided Geometric Design* 67 (2018), pp. 47–70.

[Hah89] J.M. Hahn. "Geometric continuous patch complexes". *Computer Aided Geometric Design* 6.1 (1989), pp. 55–67.

[HBC08] S. Hahmann, G.-P. Bonneau, and B. Caramiaux. "Bicubic G^1 Interpolation of Irregular Quad Meshes Using a 4-Split". *Advances in Geometric Modeling and Processing*. Ed. by Falai Chen and Bert Jüttler. Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2008, pp. 17–32.

[HCB05] T.J.R. Hughes, J.A. Cottrell, and Y. Bazilevs. "Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement". *Computer Methods in Applied Mechanics and Engineering* 194.39 (2005), pp. 4135–4195.

[Hel17] L. Heltai, J. Kiendl, A. DeSimone, and A. Reali. "A natural framework for isogeometric fluid–structure interaction based on BEM–shell coupling". *Computer Methods in Applied Mechanics and Engineering* 316 (2017), pp. 522–546.

[Hos88] J. Hoschek. "Intrinsic parametrization for approximation". *Computer Aided Geometric Design* 5.1 (1988), pp. 27–31.

[Hug21] T.J.R. Hughes, G. Sangalli, T. Takacs, and D. Toshniwal. "Smooth multi-patch discretizations in Isogeometric Analysis". *Geometric Partial Differential Equations - Part II*. Elsevier, 2021, pp. 467–543.

[JQ14] N. Jaxon and X. Qian. "Isogeometric analysis on triangulations". *Computer-Aided Design* 46 (2014), pp. 45–57.

[Jüt14] B. Jüttler, U. Langer, A. Mantzaflaris, S. Moore, and W. Zulehner. "Geometry + Simulation Modules: Implementing Isogeometric Analysis". *Proc. Appl. Math. Mech.* 14.1 (2014), pp. 961–962.

[Kap15] M. Kapl, V. Vitrih, B. Jüttler, and K. Birner. "Isogeometric analysis with geometrically continuous functions on two-patch geometries". *Computers & Mathematics with Applications* 70.7 (2015), pp. 1518–1538.

[Kha22] B. Khazaei, L.K. Read, M. Casali, K.M. Sampson, and D.N. Yates. "GLOBathy, the global lakes bathymetry dataset". *Scientific Data* 9.1 (2022).

[Kos18] K.V. Kostas, M.M. Fyrillas, C.G. Politis, A.I. Ginnis, and P.D. Kaklis. "Shape optimization of conductive-media interfaces using an IGA-BEM solver". *Computer Methods in Applied Mechanics and Engineering* 340 (2018), pp. 600–614.

[KP17a] K. Karčiauskas and J. Peters. "A New Class of Guided C^2 Subdivision Surfaces Combining Good Shape with Nested Refinement". *Computer Graphics Forum* 37.6 (2017), pp. 84–95.

[KP17b] K. Karčiauskas and J. Peters. "Guided subdivision surfaces: modeling, shape and refinability" (2017).

[KP23] K. Karčiauskas and J. Peters. "Evolving Guide Subdivision". *Computer Graphics Forum* 42.2 (2023), pp. 321–332.

[Kra97] R. Kraft. *Adaptive and Linearly Independent Multilevel B-splines*. Bericht. SFB 404, Geschäftsstelle, 1997.

[KS99] B.I. Ksasov and P. Sattayatham. "GB-splines of arbitrary order". *Journal of Computational and Applied Mathematics* (1999).

[KST18] M. Kapl, G. Sangalli, and T. Takacs. "Construction of analysis-suitable G^1 planar multi-patch parameterizations". *Computer-Aided Design* 97 (2018), pp. 41–55.

[KTC22] K.J. Koh, D. Toshniwal, and F. Cirak. "An optimally convergent smooth blended B-spline construction for semi-structured quadrilateral and hexahedral meshes". *Computer Methods in Applied Mechanics and Engineering* 399 (2022), p. 115438.

[Lan15] U. Langer, A. Mantzaflaris, S. Moore, and I. Toulopoulos. "Multipatch discontinuous Galerkin isogeometric analysis". *Isogeometric Analysis and Applications*. Lecture Notes in Computational Science and Engineering. Springer, 2015, pp. 1–32.

[Li11] G. Li, C. Ren, J. Zhang, and W. Ma. "Approximation of Loop Subdivision Surfaces for Fast Rendering". *IEEE Transactions on Visualization and Computer Graphics* 17.4 (2011), pp. 500–514.

[LMS08] T. Lyche, C. Manni, and P. Sablonnière. "Quasi-interpolation projectors for box splines". *Journal of Computational and Applied Mathematics* 221 (2008), pp. 416–429.

[LMS22] T. Lyche, C. Manni, and H. Speleers. "Construction of C^2 Cubic Splines on Arbitrary Triangulations". *Foundations of Computational Mathematics* 22.5 (2022), pp. 1309–1350.

[LMS24] T. Lyche, C. Manni, and H. Speleers. "A local simplex spline basis for C^3 quartic splines on arbitrary triangulations". *Appl. Math. Comput.* 462 (2024), p. 128330.

[Loo87] C.T. Loop. "Smooth Subdivision Surfaces Based on Triangles" (1987).

[LS07] M.-J. Lai and L.L. Schumaker. *Spline Functions on Triangulations*. Cambridge University Press, 2007.

[LS08] C. Loop and S. Schaefer. "Approximating Catmull-Clark subdivision surfaces with bicubic patches". *ACM Transactions on Graphics* 27 (2008), 8:1–8:11.

[Man20] A. Mantzaflaris. "An Overview of Geometry Plus Simulation Modules". *Mathematical Aspects of Computer and Information Sciences*. Cham: Springer International Publishing, 2020, pp. 453–456.

[Man23] A. Mantzaflaris, B. Mourrain, N. Villamizar, and B. Yuan. "An algebraic framework for geometrically continuous splines" (2023). arXiv: 2305.09096.

[Mar23] M. Marsala, A. Mantzaflaris, B. Mourrain, M. Gammon, and S. Whyman. "From CAD to Representations Suitable for Isogeometric Analysis: a Complete Pipeline" (2023). hal: 04185850.

[Mar70] M.J. Marsden. "An identity for spline functions with applications to variation-diminishing spline approximation". *Journal of Approximation Theory* 3.1 (1970), pp. 7–49.

[MAT20] MATLAB. *version 9.9.0. (R2020b)*. Natick, Massachusetts: The MathWorks Inc., 2020.

[Maz04] M.-L. Mazure. "Chebyshev Spaces and Bernstein Bases". *Constructive Approximation* 22.3 (2004), pp. 347–363.

[McN10] R. McNeel et al. "Rhinoceros 3D". *Robert McNeel & Associates, Seattle, WA* (2010).

[MMM22] M. Marsala, A. Mantzaflaris, and B. Mourrain. "G^1–Smooth biquintic approximation of Catmull-Clark subdivision surfaces". *Computer Aided Geometric Design* 99 (2022), p. 102158.

[MMM23] M. Marsala, A. Mantzaflaris, and B. Mourrain. "Analysis-suitable G^1 bases on quadrilateral meshes". *In preparation* (2023).

[MMM24] M. Marsala, A. Mantzaflaris, and B. Mourrain. "G^1 spline functions for point cloud fitting". *Applied Mathematics and Computation* 460 (2024), p. 128279.

[MMS23] M. Marsala, C. Manni, and H. Speleers. "Maximally smooth cubic spline quasi-interpolants on arbitrary triangulations". *Submitted for publication* (2023).

[MPS11] C. Manni, F. Pelosi, and M.L. Sampoli. "Generalized B-splines as a tool in isogeometric analysis". *Computer Methods in Applied Mechanics and Engineering* 200.5-8 (2011), pp. 867–881.

[MS07] C. Manni and P. Sablonnière. "Quadratic spline quasi-interpolants on Powell–Sabin partitions". *Advances in Computational Mathematics* 26 (2007), pp. 283–304.

[MVV16] B. Mourrain, R. Vidunas, and N. Villamizar. "Dimension and bases for geometrically continuous splines on surfaces of arbitrary topology". *Computer Aided Geometric Design* 45 (2016), pp. 108–133.

[NP16] T. Nguyen and J. Peters. "Refinable C^1 spline elements for irregular quad layout". *Computer Aided Geometric Design* 43 (2016), pp. 123–130.

[PBP02] H. Prautzsch, W. Boehm, and M. Paluszny. *Bézier and B-Spline Techniques*. Springer, 2002.

[Pet] J. Peters. *SurfLab Shape Gallery Page*.

[Pet00] J. Peters. "Patching Catmull-Clark meshes". *Proceedings of the 27th annual conference on Computer graphics and interactive techniques - SIGGRAPH '00* (2000).

[Pot93] H. Pottmann. "The geometry of Tchebycheffian splines". *Computer Aided Geometric Design* 10.3-4 (1993), pp. 181–210.

[PR08] J. Peters and U. Reif. *Subdivision surfaces*. Springer, 2008.

[PS77] M.J.D. Powell and M.A. Sabin. "Piecewise Quadratic Approximations on Triangles". *ACM Transactions on Mathematical Software* 3.4 (1977), pp. 316–325.

[PT95] L. Piegl and W. Tiller. *The NURBS Book*. Springer, 1995.

[Qua16] A. Quarteroni. *Modellistica Numerica per Problemi Differenziali*. Vol. 97. UNITEXT. Springer Milan, 2016.

[Rei97] U. Reif. "TURBS—Topologically Unrestricted Rational B-Splines". *Constructive Approximation* 14.1 (1997), pp. 57–77.

[RMS23] K. Raval, C. Manni, and H. Speleers. "Tchebycheffian B-splines in isogeometric Galerkin methods". *Computer Methods in Applied Mechanics and Engineering* 403 (2023), p. 115648.

[Sab85] P. Sablonnière. "Composite finite elements of class C^k". *Journal of Computational and Applied Mathematics* 12–13 (1985), pp. 541–550.

[Sch46] I. J. Schoenberg. "Contribution to the problem of approximation of equidistant data by analytic functions. Part A. On the problem of smoothing or graduation. A first class of analytic approximation formulae". *Quart. Appl. Math.* 4 (1946), pp. 45–99.

[Sed03] T.W. Sederberg, J. Zheng, A. Bakenov, and A. Nasri. "T-splines and T-NURCCs". *ACM Transactions on Graphics* 22.3 (2003), pp. 477–484.

[She14] J. Shen, J. Kosinka, M.A. Sabin, and N.A. Dodgson. "Conversion of trimmed NURBS surfaces to Catmull-Clark subdivision surfaces". *Computer Aided Geometric Design* 31.7-8 (2014), pp. 486–498.

[SM16] H. Speleers and C. Manni. "Effortless quasi-interpolation in hierarchical spaces". *Numerische Mathematik* 132 (2016), pp. 155–184.

[Sor05] O. Sorkine. "Laplacian Mesh Processing". *Eurographics 2005 - State of the Art Reports*. The Eurographics Association, 2005.

[Spe13a] H. Speleers. "Construction of normalized B-splines for a family of smooth spline spaces over Powell–Sabin triangulations". *Constructive Approximation* 37 (2013), pp. 41–72.

[Spe13b] H. Speleers. "Multivariate normalized Powell–Sabin B-splines and quasi-interpolants". *Computer Aided Geometric Design* 30 (2013), pp. 2–19.

[SS06] L. L. Schumaker and T. Sorokina. "Smooth macro-elements on Powell–Sabin-12 splits". *Mathematics of Computation* 75 (2006), pp. 711–726.

[Sta01] J. Stam. "Evaluation of Loop Subdivision Surfaces". *SIGGRAPH 99 Course Notes* (2001).

[Sta98] J. Stam. "Exact Evaluation of Catmull-Clark Subdivision Surfaces at Arbitrary Parameter Values". *Proceedings of the 25th Annual Conference on Computer Graphics and Interactive Techniques*. SIGGRAPH '98. 1998, pp. 395–404.

[Tak23] Thomas Takacs. "Approximation properties over self-similar meshes of curved finite elements and applications to subdivision based isogeometric analysis" (2023). arXiv: 2307.10403.

[TMH21] D. Toshniwal, B. Mourrain, and T.J.R. Hughes. "Polynomial spline spaces of non-uniform bi-degree on T-meshes: combinatorial bounds on the dimension". *Advances in Computational Mathematics* 47.1 (2021).

[TSH17] D. Toshniwal, H. Speleers, and T.J.R. Hughes. "Smooth cubic spline spaces on unstructured quadrilateral meshes with particular emphasis on extraordinary points: Geometric design and isogeometric analysis considerations". *Computer Methods in Applied Mechanics and Engineering* 327 (2017), pp. 411–458.

[TT23] T. Takacs and D. Toshniwal. "Almost-C^1 splines: Biquadratic splines on unstructured quadrilateral meshes and their application to fourth order problems". *Computer Methods in Applied Mechanics and Engineering* 403 (2023), p. 115640.

[Ver23] H.M. Verhelst, P. Weinmüller, A. Mantzaflaris, T. Takacs, and D. Toshniwal. *A comparison of smooth basis constructions for isogeometric analysis*. 2023. arXiv: 2309.04405.

[VZ01] L. Velho and D. Zorin. "4–8 Subdivision". *Computer Aided Geometric Design* 18.5 (2001), pp. 397–427.

[Wan01] R.-H. Wang. *Multivariate Spline Functions and Their Applications*. Kluwer Academic Publishers, 2001.

[Wan18] C. Wang, S. Xia, X. Wang, and X. Qian. "Isogeometric shape optimization on triangulations". *Computer Methods in Applied Mechanics and Engineering* 331 (2018), pp. 585–622.

[WS90] R.-H. Wang and X.-Q. Shi. "$S^{\mu}_{\mu+1}$ surface interpolations over triangulations". *Approximation, Optimization and Computing: Theory and Applications*. Ed. by A. G. Law and C. L. Wang. Elsevier Science Publishers B.V., 1990, pp. 205–208.

[Zor00] D. Zorin. "A method for analysis of C^1-continuity of subdivision surfaces". *SIAM Journal on Numerical Analysis* 37.5 (2000), pp. 1677–1708.

[ZQ19] M. Zareh and X. Qian. "Kirchhoff–Love shell formulation based on triangular isogeometric analysis". *Computer Methods in Applied Mechanics and Engineering* 347 (2019), pp. 853–873.

,