Subdivision and manifold techniques for isogeometric design and analysis of surfaces

Qiaoling Zhang

Department of Engineering
University of Cambridge

This dissertation is submitted for the degree of
Doctor of Philosophy

Pembroke College January 2019

Declaration

I hereby declare that except where specific reference is made to the work of others, the contents of this dissertation are original and have not been submitted in whole or in part for consideration for any other degree or qualification in this, or any other university. This dissertation is my own work and contains nothing which is the outcome of work done in collaboration with others, except as specified in the text and Acknowledgements. This dissertation contains fewer than 65,000 words including appendices, references, footnotes, tables and equations and has fewer than 150 figures.

Qiaoling Zhang
January 2019

Abstract

Design of surfaces and analysis of partial differential equations defined on them are of great importance in engineering applications, e.g., structural engineering, automotive and aerospace. This thesis focuses on isogeometric design and analysis of surfaces, which aims to integrate engineering design and analysis by using the same representation for both. The unresolved challenge is to develop a desirable surface representation that simultaneously satisfies certain favourable properties on meshes of arbitrary topology around the extraordinary vertices (EVs), i.e., vertices not shared by four quadrilaterals or three triangles. These properties include high continuity(geometric or parametric), optimal convergence in finite element analysis as well as simplicity in terms of implementation. To overcome the challenge, we further develop subdivision and manifold surface modelling techniques, and explore a possible scheme to combine the distinct appealing properties of the two. The unique advantages of the developed techniques have been confirmed with numerical experiments.

Subdivision surfaces generate smooth surfaces from coarse control meshes of arbitrary topology by recursive refinement. Around EVs the optimal refinement weights are application-dependent. We first review subdivision-based finite elements. We then proceed to derive the optimal subdivision weights that minimise finite element errors and can be easily incorporated into existing implementations of subdivision schemes to achieve the same accuracy with much coarser meshes in engineering computations. To this end, the eigenstructure of the subdivision matrix is extensively used and a novel local shape decomposition approach is proposed to choose the optimal weights for each EV independently.

Manifold-based basis functions are derived by combining differential-geometric manifold techniques with conformal parametrisations and the partition of unity method. This thesis derives novel manifold-based basis functions with arbitrary prescribed smoothness using quasi-conformal maps, enabling us to model and analyse surfaces with sharp features, such as creases and corners. Their practical utility in finite element simulation of hinged or rigidly joined structures is demonstrated with Kirchhoff-Love thin shell examples.

We also propose a particular manifold basis reproducing subdivision surfaces away from EVs, i.e., B-splines, providing a way to combine the appealing properties of subdivision (available in industrial software) for design and manifold basis (relatively new) for analysis.

I would like to dedicate this thesis to my parents and sister for their endless support, as well as my nephew, a cute and smart boy whom I missed most while being away from home.

Acknowledgements

I would like to thank everyone who has helped me either to secure the PhD offer here at Cambridge or to accomplish the PhD degree in four-year time full of joy.

First, I am grateful for my supervisor Dr Fehmi Cirak, who not only showed great encouragement while I had a hard time trying to pass the English requirement before admission but also provided enormous support during my whole PhD study. There are a lot of moments, too many to list here, when I sincerely feel myself is lucky to work with such a responsible, supportive, patient and passionate supervisor. What is really impressive and inspiring to me is his ability to pay attention to details in everything he does, even including reference letter writing, let alone research work and academic paper writing. I would never forget how much effort he has made for my PhD overrun funding as well as college Junior Research Fellowship applications. I still remember how much I was touched every time when I saw the email notification popping out late night implying that he was working hard on something that he could have left for me. Looking back to the last four years, I have learned a lot thanks to the patient supervision from Fehmi just as how he guides every student in our group. Not only would he often explain theories exceptionally well in detail but also sit down to help us debug. All of these merits of being a reputable supervisor are attributed to his passion for research and education.

I would also like to acknowledge Prof Malcolm Sabin, who contributed to our work on subdivision tuning, inspired me to great extent with his expertise in computer-aided geometric design, and provided insightful comments for this thesis as well.

In addition, I appreciate the comfortable working environment at CSMLab, where I enjoyed intellectually stimulating academic communication as well as fascinating casual chats with my lovely group mates. They are Xiao Xiao, Eky Febrianto, Ge Yin, Sumudu Herath, Kim Jie Koh, Adeeb Arif Kor, Arion Pons and Dr Musabbir Majeed. The friendship from Shan Zheng, Yanting Jin, Yuanyuan Liu, Gulzat Jaliel makes the time at Cambridge my lifelong treasure.

My acknowledgement also goes to Cambridge Trust for providing the full scholarship.

Last but not least, love from my parents, sister and my boyfriend Dr Thomas Kopsch makes me become who I am, a person who has learned to keep calm in bad times and feel thankful in good times.

Table of contents

List of figures

List of tables

Chapter 1

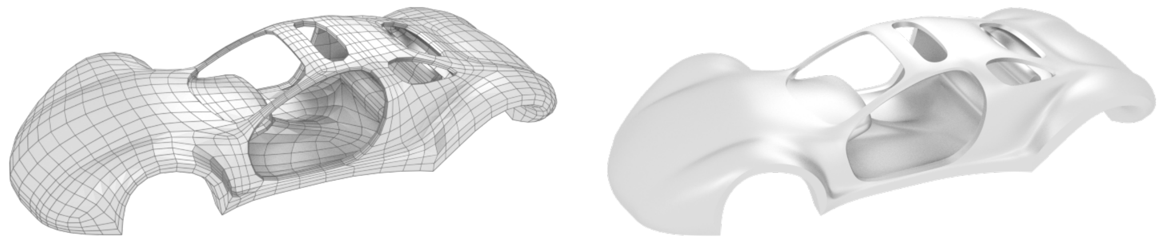
Introduction

1.1 Motivation

Performance-driven product design consists of two integral parts, i.e., computer-aided design (CAD) and finite element analysis (FEA). A geometry model is created in CAD software and a FEA simulation is run to predict the performance of the designed geometry. Based on the FEA result, the initial design will be modified iteratively to improve its performance. The integration of CAD and FEA systems will lead to a seamless design-analysis workflow and thus facilitate the overall product development cycle. The integration can be possibly achieved in an isogeometric design-analysis framework, where the same parametric representation used for geometric modelling in CAD will also be used to discretise physical fields in FEA. However, it is challenging to develop such a parametric representation that simultaneously satisfies all the desirable properties on meshes of arbitrary topology, such as high geometric or parametric continuity, optimal convergence in finite element analysis as well as simplicity in terms of implementation. This thesis focuses on the development of parametric representations for isogeometric design and analysis of surfaces, motivated by their wide applications in engineering, e.g., automotive, aerospace and structural engineering. We consider surfaces that are modelled from quadrilateral meshes of arbitrary topology and may include sharp features such as creased edges and corners, see Figure 1.1.

Isogeometric analysis

The integration of CAD and FEA is theoretically challenging but practically necessary. In industrial practice, the design and analysis software systems use incompatible parametric representations mainly because they were developed independently. Research about FEA, a widely used method for simulation, dated back to the early 1940s [1, 2], whereas it



(a) Control mesh **(b)** Manifold surface

Figure 1.1: A car with creased edges and sharp corners. (a) The control mesh has arbitrary topology and includes extraordinary vertices shared by other than four quadrilateral faces. (b) The creases and corners are faithfully reproduced using the new manifold-based basis functions proposed in Chapter 5.

was in the 1960s that the study of interactive free-form curve/surface design used in CAD software began [3]. The incompatibility between CAD and FEA systems requires a tedious meshing process to convert a geometry model created in CAD into a mesh suitable for FEA. To generate high-quality meshes for FEA is not only time-consuming but can also be challenging. To resolve the incompatibility and improve the efficiency of the design-analysis workflow, the concept of *isogeometric analysis* was proposed about a decade ago [4].

As envisioned in [4], isogeometric analysis aims to achieve a seamless engineering design-analysis workflow by representing geometry and discretising analysis models with the same kind of basis functions. Therefore, until recently isogeometric analysis was dominated by Non-Uniform Rational B-Splines (NURBS) basis functions, which are prevalent in present CAD systems. In order to represent the surface of parts with arbitrary topology, CAD systems resort to trimmed NURBS and boundary representations (B-Reps). As pointed out in the extensive review [5], trimming is much more complicated than most people think and provides a deceptively simple solution without addressing the root challenge. Furthermore, trimming generates non-watertight geometries which pose unique challenges in the analysis context. In order to represent surfaces of arbitrary topology without turning to trimming, unstructured meshes containing extraordinary vertices, that is, vertices inside the domain with other than four attached quadrilateral patches, are required.

In the community of geometric design, numerous techniques have been developed to provide smooth watertight representations around extraordinary vertices, including geometrically G^k and parametrically C^k continuous constructions, subdivision surfaces and also manifold-based surface constructions. These smooth parametrisation techniques developed to deal with extraordinary vertices provide promising candidate basis functions for isogeometric analysis. The application and further development of these techniques is currently a very active area of research in isogeometric analysis, see e.g. [6–11]. The search for easy to implement and optimally convergent schemes, especially with $C^{k\geq 2}$ and $G^{k\geq 2}$, is still open.

In this thesis, subdivision and manifold surface modelling techniques are investigated and further developed to improve their suitability for isogeometric analysis. A more detailed introduction of each technique can be found in Chapter 4 and Chapter 5. In a complementary line of research, there has been progress in isogeometric analysis of shells based on trimmed surfaces and weak enforcement of mechanical continuity conditions across patch boundaries [12, 13].

1.2 Contributions

This thesis has three main contributions to the progress of isogeometric analysis. The main obstacle faced by the realisation of isogeometric analysis in industrial standard is that the ideal basis functions suitable for both geometric design and simulation have not yet been developed. Each candidate method needs more or less further improvement. This thesis aims to improve the suitability of subdivision and manifold surface modelling techniques for isogeometric analysis. These are two promising candidate methods in our opinion, even though there are still remaining challenges.

Subdivision surfaces, as a generalisation of splines, were first introduced and studied in the late 1970s [14, 15]. Before the advent of isogeometric analysis, it had already been realised that subdivision surfaces provide also ideal basis functions for finite element analysis, in particular, of thin-shells [6, 16–18]. As known, around extraordinary vertices the subdivision weights need to be carefully tuned to achieve certain surface properties. However, it is mathematically impossible to simultaneously achieve all (second order) smoothness properties. Therefore, the optimal subdivision weights are application-dependent. While previous research works focus on subdivision tuning for geometry modelling, this thesis makes the first attempt to derive optimal subdivision weights for isogeometric analysis applications. As our computations confirm, the optimised subdivision weights yield a reduction of 50% and more in discretisation errors in the energy and L_2 norms. Although, as to be expected, the convergence rates are the same as for the classical Catmull-Clark weights, the convergence constants are improved. The optimised weights can be easily incorporated into existing implementations of Catmull-Clark subdivision to achieve the same accuracy with much coarser meshes in engineering computations. This work has been published in the journal *Computer-Aided Design* [11].

Manifold-based surface constructions have a rich history in geometric modelling [19–23]. Manifold-based basis functions for isogeometric analysis were first introduced about two years ago in [24]. The proposed $C^{k\geq 1}$ continuous manifold basis functions can represent parts with arbitrary topology and show optimal convergence rates in finite element computations.

This thesis extends previous work on smooth manifold basis functions [24] and derives novel manifold-based basis functions with arbitrary prescribed smoothness to represent surfaces with sharp features. This is achieved by proposing a new quasi-conformal mapping to parametrise the local manifold chart domains with arbitrary prescribed creases. This extension greatly widens the range of engineering geometries that can be modelled and analysed by manifold-based basis functions. This work has been presented at international conferences as well as invited workshops. A paper has been submitted to the journal *Computer Methods in Applied Mechanics and Engineering*.

This thesis also builds a connection between subdivision and manifold surface modelling techniques by proposing a particular manifold construction that is able to reproduce subdivision surfaces in regular region. This construction provides a foundation for a quasi-isogeometric design-analysis framework, where geometries can be designed using subdivision surfaces given their appealing properties in surface modelling and analysed using manifold representations for their optimal convergence in finite element analysis. Even though this idea differs from the conventional isogeometric paradigm, it is potentially useful to point out a direction towards the realisation of an integrated CAD-FEA system. This work has been included as a section of a paper to be published in the proceedings of IGAA 2018 conference.

1.3 Outline

Chapter 2 introduces the parametric geometry representations used in CAD with a focus on the underlying basis functions, which forms a basis of the research work presented in the following chapters. We start with univariate basis functions, including Bernstein polynomials for Bézier curves and B-spline basis functions for splines. Then we proceed to the subdivision construction for B-spline curves. In particular, we focus on subdivision of cubic B-spline curves to introduce the knot insertion algorithm and boundary conditions for B-splines.

Chapter 3 motivates all the concepts required to understand subdivision-based isogeometric analysis. To keep the discussion concise, one-dimensional problems are used as illustrative examples to demonstrate in detail how to use subdivision-generated basis functions to perform finite element analysis.

Chapter 4 presents the research work on subdivision surfaces. We investigate how to systematically optimise subdivision weights for isogeometric analysis applications. The proposed optimisation is demonstrated for Catmull-Clark subdivision scheme, the most widely used subdivision technique.

Chapter 5 presents the research work on manifold-based basis functions. We extend the smooth manifold basis functions proposed in [24] to model and perform isogeometric analysis on surfaces with arbitrary prescribed sharp features. To this end a new quasi-conformal map is proposed to parametrise the local manifold chart domains.

Chapter 6 discusses possible ways to construct a manifold representation that is able to reproduce subdivision surfaces in regular region, i.e., tensor-product B-splines. The discussion is restricted to the univariate setting without losing generality.

Chapter 7 summarises the thesis and points out several directions for future research.

Chapter 2

Basics of subdivision curves

This chapter provides basic materials that will help to understand subdivision curves and thus subdivision surfaces. We first review Bézier curves and their properties. We then proceed to B-splines by introducing their definition, de Boor's evaluation algorithm and Boehm's knot insertions algorithm. With the knot insertion algorithm in place, we introduce subdivision curves focusing on the univariate subdivision schemes that generate cubic B-splines. At the end of this chapter, boundary conditions and creases for subdivision curves are discussed. We will demonstrate how to use the introduced univariate basis functions constructed by subdivision for finite element analysis in the next chapter.

2.1 Bézier curves

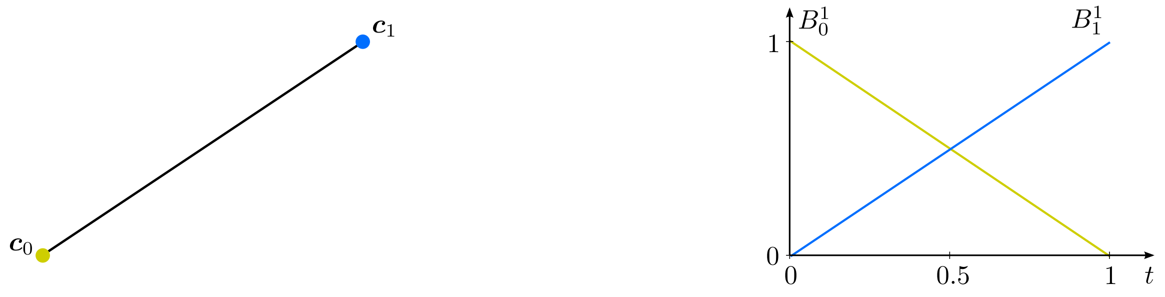
A Bézier curve is a parametric curve that is frequently used to model smooth curves. It can be constructed recursively by the de Casteljau algorithm, see e.g. [25, Chapter 3]. Alternatively, a Bézier curve $\boldsymbol{c}(t)$ can be explicitly represented as

$$\boldsymbol{c}(t) = \sum_{j=0}^{n} B_j^n(t)\boldsymbol{c}_j\,, \quad t \in [0,1]\,, \tag{2.1}$$

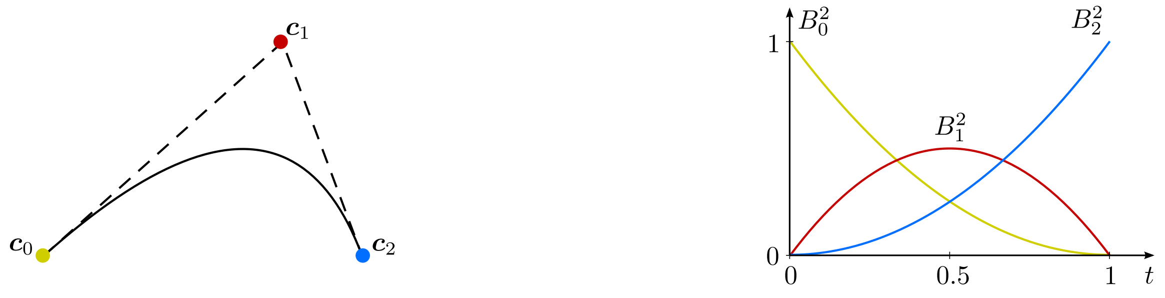
where $\boldsymbol{c}_j \in \mathbb{R}^d$ are the control points in real coordinate space of d dimensions and $B_j^n(t)$ are Bernstein polynomials of degree n defined as

$$B_j^n(t) = \binom{n}{j} t^j (1-t)^{n-j}\,, \quad j = 0,1,2,\cdots,n \tag{2.2}$$

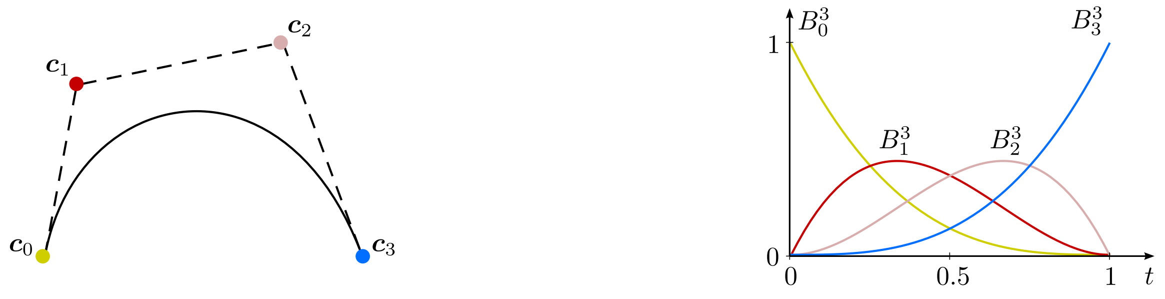
with $\binom{n}{j} = \frac{n!}{j!(n-j)!}$ denoting binomial coefficients. Figure 2.1 plots Bézier curves and Bernstein basis of different degrees, from linear to cubic. Bézier curves have many important properties making them a popular technique for geometric design. A detailed introduction of Bézier curve properties can be found in [25, Chapter 4 and 5]. In the following, we only summarise a few that are relevant to this work.



(a) Linear



(b) Quadratic



(c) Cubic

Figure 2.1: Left: Bézier curves (solid) of different degrees and their control polygons (dashed); Right: corresponding Bernstein basis $B_j^n(t)$ of degree $n \in \{1,2,3\}$ and index $j \in \{0,1,\cdots,n\}$.

Affine invariance

From (2.2), it is straightforward to verify that Bernstein polynomials form a partition of unity, i.e.,

$$\sum_{j=0}^{n} B_j^n(t) = \sum_{j}^{n} \binom{n}{j} t^j (1-t)^{n-j} = [(1-t)+t]^n \equiv 1\,, \tag{2.3}$$

leading to

$$\boldsymbol{A}\boldsymbol{c}(t) + \boldsymbol{v} = \boldsymbol{A}\sum_{j=0}^{n} B_j^n(t)\boldsymbol{c}_j + \boldsymbol{v}\sum_{j=0}^{n} B_j^n(t) = \sum_{j=0}^{n} B_j^n(t)(\boldsymbol{A}\boldsymbol{c}_j + \boldsymbol{v})\,. \tag{2.4}$$

This means that applying an affine transformation to a Bézier curve is equivalent to simply applying the affine map to its control points, which provides convenience for geometric modelling.

Convex hull property

Bézier curves always fall within the convex hull of their control points, see Figure 2.2. Given (2.2), it is clear that Bernstein polynomials are nonnegative for $t \in [0,1]$. Also, they have been shown to sum up to one in (2.3). As a result, the convex hull property follows.

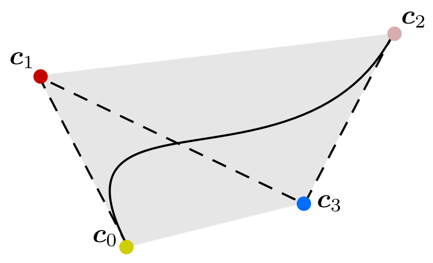


Figure 2.2: A cubic Bézier curves and the convex hull (shaded in grey) of its four control points.

Endpoint interpolation

A Bézier curve starts from its first control point and ends at its last control point, see the plots in Figure 2.1. From the definition of Bernstein basis (2.2), we have

$$B_j^n(0) = \begin{cases} 1 & \text{if } j = 0 \\ 0 & \text{else} \end{cases}, \quad \text{and} \quad B_j^n(1) = \begin{cases} 1 & \text{if } j = n \\ 0 & \text{else} \end{cases}, \tag{2.5}$$

which together with (2.1) proves the following endpoint interpolation property:

$$\boldsymbol{c}(0) = \boldsymbol{c}_0\,, \quad \boldsymbol{c}(1) = \boldsymbol{c}_n\,. \tag{2.6}$$

Derivatives

We first consider the derivatives of Bernstein polynomials,

$$\begin{aligned}\frac{\mathrm{d}B_j^n(t)}{\mathrm{d}t} &= \frac{\mathrm{d}}{\mathrm{d}t}\binom{n}{j} t^j(1-t)^{n-j} \\ &= \frac{n!}{(j-1)!(n-j)!}t^{j-1}(1-t)^{n-j} - \frac{n!}{j!(n-j-1)!}t^j(1-t)^{n-j-1} \\ &= n\left(B_{j-1}^{n-1}(t) - B_j^{n-1}(t)\right).\end{aligned} \tag{2.7}$$

Note that by default $B_j^n(t) = 0$ for $j \notin \{0, 1, \cdots, n\}$. By performing the differentiation repeatedly, we obtain the following general expression for r-th order derivatives:

$$\frac{\mathrm{d}^r B_j^n(t)}{\mathrm{d}t^r} = \frac{n!}{(n-r)!}\sum_{k=0}^{r}\binom{r}{k}(-1)^{r-k}B_{j-k}^{n-r}(t). \tag{2.8}$$

Now we are ready to derive the r-th order derivatives of Bézier curves, i.e.,

$$\begin{aligned}\frac{\mathrm{d}^r \boldsymbol{c}(t)}{\mathrm{d}t^r} &= \sum_{j=0}^{n}\boldsymbol{c}_j\frac{\mathrm{d}^r B_j^n(t)}{\mathrm{d}t^r} \\ &= \frac{n!}{(n-r)!}\sum_{j=0}^{n}\sum_{k=0}^{r}\binom{r}{k}(-1)^{r-k}B_{j-k}^{n-r}(t)\boldsymbol{c}_j \\ &= \frac{n!}{(n-r)!}\sum_{j=0}^{n-r}B_j^{n-r}(t)\underbrace{\sum_{k=0}^{r}\binom{r}{k}(-1)^{r-k}\boldsymbol{c}_{j+k}}_{\Delta^r\boldsymbol{c}_j} \\ &= \frac{n!}{(n-r)!}\sum_{j=0}^{n-r}B_j^{n-r}(t)\Delta^r\boldsymbol{c}_j\end{aligned} \tag{2.9}$$

where Δ^r is the r-th order difference operator, e.g,

$$\Delta^1\boldsymbol{c}_j = \boldsymbol{c}_{j+1} - \boldsymbol{c}_j, \quad \Delta^2\boldsymbol{c}_j = \boldsymbol{c}_{j+2} - 2\boldsymbol{c}_{j+1} + \boldsymbol{c}_j. \tag{2.10}$$

As shown in (2.9), the r-th order derivatives of degree-n Bézier curves can be interpreted as Bézier curves of degree $n-r$ with control points $\frac{n!}{(n-r)!}\Delta^r\boldsymbol{c}_j$. Recall the property of endpoint interpolation (2.5), the derivatives of Bézier curves at two ends are determined by the first and last well-defined r-th order difference, i.e.,

$$\frac{\mathrm{d}^r\boldsymbol{c}(0)}{\mathrm{d}t^r} = \frac{n!}{(n-r)!}\Delta^r\boldsymbol{c}_0, \quad \frac{\mathrm{d}^r\boldsymbol{c}(1)}{\mathrm{d}t^r} = \frac{n!}{(n-r)!}\Delta^r\boldsymbol{c}_{n-r}. \tag{2.11}$$

See (2.9) for the expression of the difference operator $\Delta^r \boldsymbol{c}_j$. This shows that the r-th order derivatives of Bézier curves at two ends only depend on the $r+1$ closest control points to each end. The first and second order derivatives are of particular interests in practice. Plugging (2.10) into (2.11) leads to

$$\begin{aligned} \frac{\mathrm{d}\boldsymbol{c}(0)}{\mathrm{d}t} &= n(\boldsymbol{c}_1 - \boldsymbol{c}_0)\,, & \frac{\mathrm{d}^2\boldsymbol{c}(0)}{\mathrm{d}t^2} &= n(n-1)(\boldsymbol{c}_2 - 2\boldsymbol{c}_1 + \boldsymbol{c}_0)\,, \\ \frac{\mathrm{d}\boldsymbol{c}(1)}{\mathrm{d}t} &= n(\boldsymbol{c}_n - \boldsymbol{c}_{n-1})\,, & \frac{\mathrm{d}^2\boldsymbol{c}(1)}{\mathrm{d}t^2} &= n(n-1)(\boldsymbol{c}_n - 2\boldsymbol{c}_{n-1} + \boldsymbol{c}_{n-2})\,. \end{aligned} \tag{2.12}$$

In particular, Bézier curves are tangential to the first segment $\boldsymbol{c}_0\boldsymbol{c}_1$ and last segment $\boldsymbol{c}_{n-1}\boldsymbol{c}_n$ of the control polygons, see Figure 2.1.

Splines in Bézier form

A spline is a piecewise polynomial, whose pieces meet with the highest possible continuity for the given degree. We can represent the individual pieces in the Bézier representation and this provides more flexible local control than one single Bézier segment. With the explicit expression for derivatives (2.11), it becomes straightforward to construct splines that are C^r-continuous at junction nodes in Bézier form. Consider two Bézier curves of the same degree n defined on two arbitrary consecutive parameter domains $[u_0, u_1]$ and $[u_1, u_2]$. To be able to use the derivative expressions we have obtained so far in the reference domain $t \in [0,1]$, we first map the arbitrary parameter domain $u \in [u_j, u_{j+1}]$ to the reference domain $t \in [0,1]$ with a linear transformation $t_j(u) = (u - u_j)/(u_{j+1} - u_j)$. As the result, the composed piecewise Bézier curve is defined as follows:

$$\boldsymbol{c}(u) = \begin{cases} \sum_{j=0}^{n} B_j^n(t)\boldsymbol{c}_j\,, & u_0 \le u \le u_1 \\ \sum_{j=0}^{n} B_j^n(t)\boldsymbol{c}_{j+n}\,, & u_1 \le u \le u_2 \end{cases} \quad \text{with} \quad t = \frac{u - u_j}{u_{j+1} - u_j}\,, \quad u \in [u_j, u_{j+1}]\,. \tag{2.13}$$

Then the derivatives for $u \in [u_j, u_{j+1}]$ are calculated using the chain rule, i.e.,

$$\frac{\mathrm{d}\boldsymbol{c}(u)}{\mathrm{d}u} = \frac{\mathrm{d}t}{\mathrm{d}u}\frac{\mathrm{d}\boldsymbol{c}(t)}{\mathrm{d}t} = \frac{1}{u_{j+1} - u_j}\frac{\mathrm{d}\boldsymbol{c}(t)}{\mathrm{d}t}\,, \tag{2.14}$$

which generalises to

$$\frac{\mathrm{d}^r\boldsymbol{c}(u)}{\mathrm{d}u^r} = \frac{1}{(u_{j+1} - u_j)^r}\frac{\mathrm{d}^r\boldsymbol{c}(t)}{\mathrm{d}t^r}\,. \tag{2.15}$$

Combining (2.12) and (2.15) gives the conditions that should be satisfied by the control points to make the composed curve defined in (2.13) C^2 continuous at the junction point u_1, i.e.,

$$\frac{\boldsymbol{c}_n - \boldsymbol{c}_{n-1}}{u_1 - u_0} = \frac{\boldsymbol{c}_{n+1} - \boldsymbol{c}_n}{u_2 - u_1}, \tag{2.16a}$$

$$\frac{\boldsymbol{c}_n - 2\boldsymbol{c}_{n-1} + \boldsymbol{c}_{n-2}}{(u_1 - u_0)^2} = \frac{\boldsymbol{c}_{n+2} - 2\boldsymbol{c}_{n+1} + \boldsymbol{c}_n}{(u_2 - u_1)^2}. \tag{2.16b}$$

As an illustrative example, Figure 2.3 shows two cubic Bézier curves attached together C^2 continuously. The first and second derivatives plotted in the same figure clearly indicate the C^2 continuity of the composite curve. Note that the derivatives of the x-component of the curve are not considered, because the chosen control points lead to $c_x(u) = u$ meaning that $c_x(u)$ is C^∞ continuous at the junction node $u_1 = 1$. The linear constraints of control points (2.16) effectively reduce the number of degrees of freedom. The same effect can be achieved by a basis transformation. This means a spline can be represented by unconstrained control points and the transformed basis functions, leading to the B-spline form. In practice, the B-spline form is easier to use since one can move each control point freely without breaking the continuity of the spline. We will introduce the B-spline basis in Section 2.2.

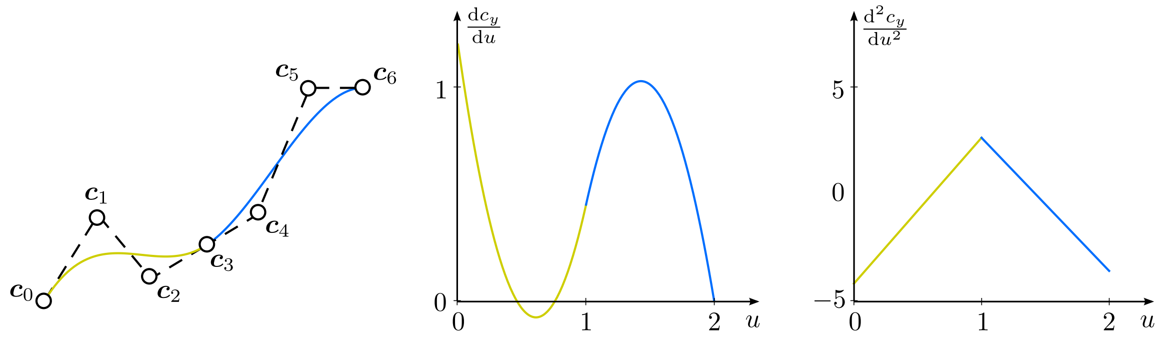


Figure 2.3: A C^2 continuous piecewise cubic curve composed of two Bézier pieces (left) and the derivatives of the y-component (right). The parameter domains are $[u_0, u_1] = [0, 1]$ and $[u_1, u_2] = [1, 2]$. The control points take for x-coordinates $[0, 1, 2, 3, 4, 5, 6]/3$ and y-coordinates $[0, 0.4, 0.1, 0.25, 0.4, 1, 1]$. The C^2 continuity requirements (2.16) are satisfied, leading to continuous derivatives up to second order.

Degree elevation

Any Bézier curve of degree n can also be written as a Bézier curve of degree $d \geq n$. This property is useful for the technique of creating piecewise Bézier surfaces by gluing several Bézier patches of different degrees together. This technique will be exploited in Chapter 5 for the modelling of manifold-based surfaces with sharp features.

To keep the shape of the represented curve unchanged while elevating the degree from n to $n+1$, the new control points $\boldsymbol{c}_j^{n+1}$ are determined as follows:

$$
\begin{aligned}
\boldsymbol{c}(t) &= [(1-t)+t]\sum_{j=0}^{n} B_j^n(t)\boldsymbol{c}_j^n \\
&= \sum_{j=0}^{n} \frac{n!}{j!(n-j)!} t^j (1-t)^{n-j+1}\boldsymbol{c}_j^n + \sum_{j=0}^{n} \frac{n!}{j!(n-j)!} t^{j+1}(1-t)^{n-j}\boldsymbol{c}_j^n \\
&= \sum_{j=0}^{n} \frac{n+1-j}{n+1} B_j^{n+1}(t)\boldsymbol{c}_j^n + \sum_{j=0}^{n} \frac{j+1}{n+1} B_{j+1}^{n+1}(t)\boldsymbol{c}_j^n \\
&= \sum_{j=0}^{n+1} \frac{n+1-j}{n+1} B_j^{n+1}(t)\boldsymbol{c}_j^n + \sum_{j=0}^{n+1} \frac{j}{n+1} B_j^{n+1}(t)\boldsymbol{c}_{j-1}^n \\
&= \sum_{j=0}^{n+1} B_j^{n+1}(t) \underbrace{\left(\frac{n+1-j}{n+1}\boldsymbol{c}_j^n + \frac{j}{n+1}\boldsymbol{c}_{j-1}^n \right)}_{\boldsymbol{c}_j^{n+1}} .
\end{aligned}
\tag{2.17}
$$

Note that the endpoint interpolation indicates the endpoints stay unchanged, i.e., $\boldsymbol{c}_0^{n+1} = \boldsymbol{c}_0^n$ and $\boldsymbol{c}_{n+1}^{n+1} = \boldsymbol{c}_n^n$, which are naturally implied in (2.17). Figure 2.4 illustrates the process of degree elevation by taking quadratic and cubic curves as examples. In each case, the new control points get closer to the curve compared with the control points at a lower degree. The degree elevation process can be applied repeatedly and the control points will converge to the curve represented.

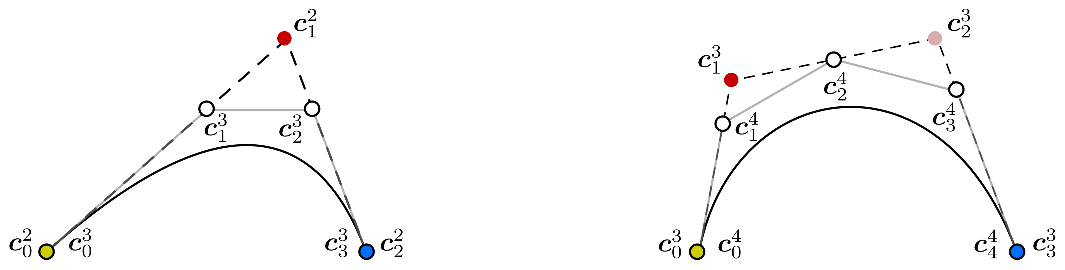


Figure 2.4: Degree elevation of Bézier curves. The new control points $\boldsymbol{c}_j^{n+1}$ are linear averages of the lower degree control points $\boldsymbol{c}_j^n$ according to (2.17). The new control polygon (grey solid) gets closer to the curve compared with the control polygon (dashed black) at a lower degree.

2.2 B-splines

2.2.1 Definition of B-spline basis functions

Consider a non-decreasing *knot vector* $U = \{u_0, u_1, \cdots, u_{n+1}\}$, the B-spline basis functions $B_j^n(u)$ of degree $n = 0$ are step functions, defined as:

$$B_j^0(u) = \begin{cases} 1\,, & \text{if } u_j \leq u < u_{j+1} \\ 0\,, & \text{otherwise} \end{cases}\,, \tag{2.18}$$

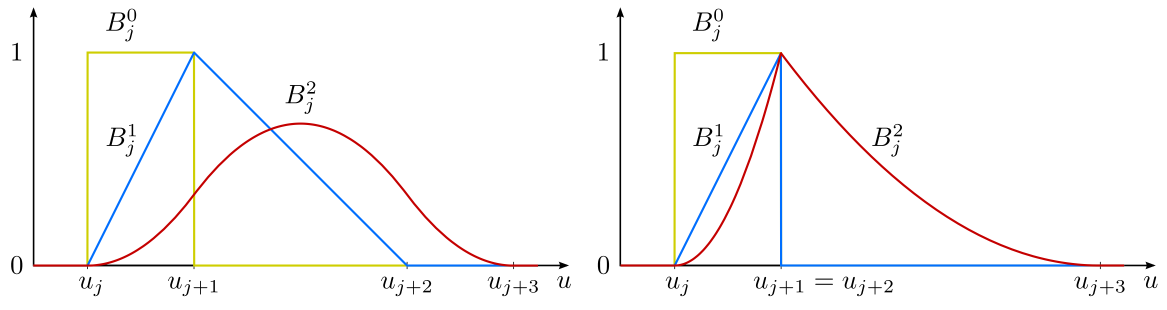
and B-splines of higher degrees can be constructed by the Cox-de Boor recursion formula, i.e.,

$$B_j^n(u) = \alpha_j^{n-1}(u) B_j^{n-1}(u) + \left(1 - \alpha_{j+1}^{n-1}(u)\right) B_{j+1}^{n-1}(u)\,, \tag{2.19a}$$

with

$$\alpha_j^n(u) = \begin{cases} \frac{u-u_j}{u_{j+n+1}-u_j}\,, & \text{if } u_{j+n+1} \neq u_j \\ 0\,, & \text{otherwise} \end{cases}\,. \tag{2.19b}$$

$\alpha_j^n(u)$ is a linear function increasing from 0 to 1 for $u \in [u_j, u_{j+n+1}]$, i.e. the domain where $B_j^n(u)$ is not zero. Figure 2.5 plots the B-splines $B_j^n(u)$ up to degree $n = 2$. I would like to point out that in this thesis we use B_j^n to denote both Bernstein polynomials and B-splines. As we shall see in a later section, Bernstein polynomials are particular instances of B-splines.



(a) Strictly increasing knots **(b)** Double knots at $u_{j+1} = u_{j+2}$

Figure 2.5: B-splines $B_j^n(u)$ of degree $n \in \{0, 1, 2\}$, defined over two different knot vectors. In (a), at the knot u_{j+1} B-splines $B_j^0(u)$, $B_j^1(u)$ and $B_j^2(u)$ are C^{-1}, C^0 and C^1 continuous, respectively. In (b), however, at the duplicated knot $u_{j+1} = u_{j+2}$ the B-spline $B_j^2(u)$ is only C^0 continuous and $B_j^1(u)$ only C^{-1} continuous.

Local support

The support of a function is the domain where it is not zero. Each degree-n B-spline $B^n_j(u)$ has a local support $[u_j, u_{j+n+1}]$, which consists of only $n+1$ knot intervals.

Smoothness

B-spline $B^n_j(u)$ is a piecewise polynomial of degree n, completely determined by the $n+2$ knots $\{u_j, u_{j+1}, \cdots, u_{j+n+1}\}$ as well as the prescribed degree n. Within each knot span, it is a degree-n polynomial and thus C^∞ continuous. At each knot, however, it is C^{n-m} continuous, where m denotes the multiplicity of the knot. This means for a knot vector without any repeated knots ($m = 1$) the B-splines of degree n reach the maximum continuity C^{n-1}. From Figure 2.5, we can notice that the knot repetition $u_{j+1} = u_{j+2}$ reduces the smoothness of the B-splines $B^1_j(u)$ and $B^2_j(u)$, whereas $B^0_j(u)$ is not affected by this duplication because the knot u_{j+2} is outside its support.

Uniform B-splines

A commonly used knot vector is $U = \{0, 1, 2, \cdots\}$, containing knots that are equally-spaced in the parametric space. This type of uniform knot vectors generate uniform B-splines, also known as *cardinal B-splines*, which are shifted instances of $B^n_0(u)$.

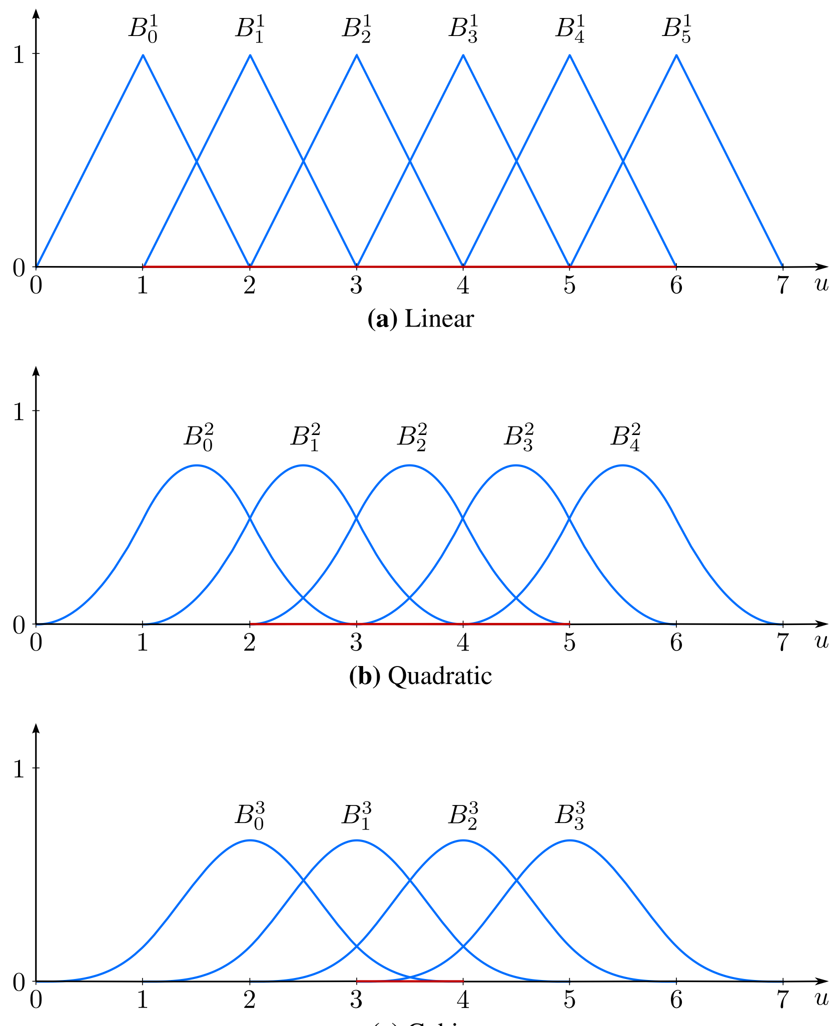
$$B^n_j(u) = B^n_0(u-j), \quad j \in \{0, 1, 2, \cdots\} \tag{2.20}$$

Figure 2.6 plots cardinal B-splines of different degrees from linear to cubic. Uniform B-spline basis functions are much simpler in terms of computations, compared with B-splines defined by non-uniform knots. The simplicity makes uniform B-splines popular in most implementations, such as approximation and finite element analysis.

Two-scale relation of uniform B-splines

Consider a uniform knot vector $U = \{0, 1, 2, \cdots\}$ and refine it by bisection, i.e., inserting new knots halfway between existing knots. The new knot vector is $\hat{U} = \{0, 1/2, 1, \cdots\}$. There is a scaling relation between the uniform B-splines defined on the initial knot vector U and those on the refined knot vector $\hat{U}$, i.e.

$$\hat{B}^n_{2j}(u) = B^n_j(2u) = B^n_0(2u-j), \tag{2.21}$$



(a) Linear

(b) Quadratic

(c) Cubic

Figure 2.6: B-splines $B_j^n(u)$ defined on a uniform knot vector $U = 0, 1, \cdots, z$. Note that given a uniform knot vector $U = \{0, 1, \cdots, z\}$ a complete and linearly independent piecewise polynomial basis is only formed over part of the parameter domain $n \leq u \leq z - n$ (highlighted in red) for B-splines of degree n.

where the factor 2 scales the support size of a B-spline at the finer level with respect to the initial support size. Moreover, the uniform B-splines of degree n defined on a knot vector U can be represented as a linear combination of B-splines defined on the bisected knot vector $\hat{U}$, i.e.,

$$B_j^n(u) = \sum_{k=0}^{n+1} S_{j,k}^n \hat{B}_{2j+k}^n(u), \quad \text{with} \quad S_{j,k}^n = \frac{1}{2^n} \binom{n+1}{k} \tag{2.22}$$

where $S_{j,k}^n$ denotes the subdivision matrix and its entries are determined as the usual binomial coefficients. (2.22) is known as refinability of uniform B-splines, which is one of the most important properties of B-splines related to its application in adaptive finite element analysis.

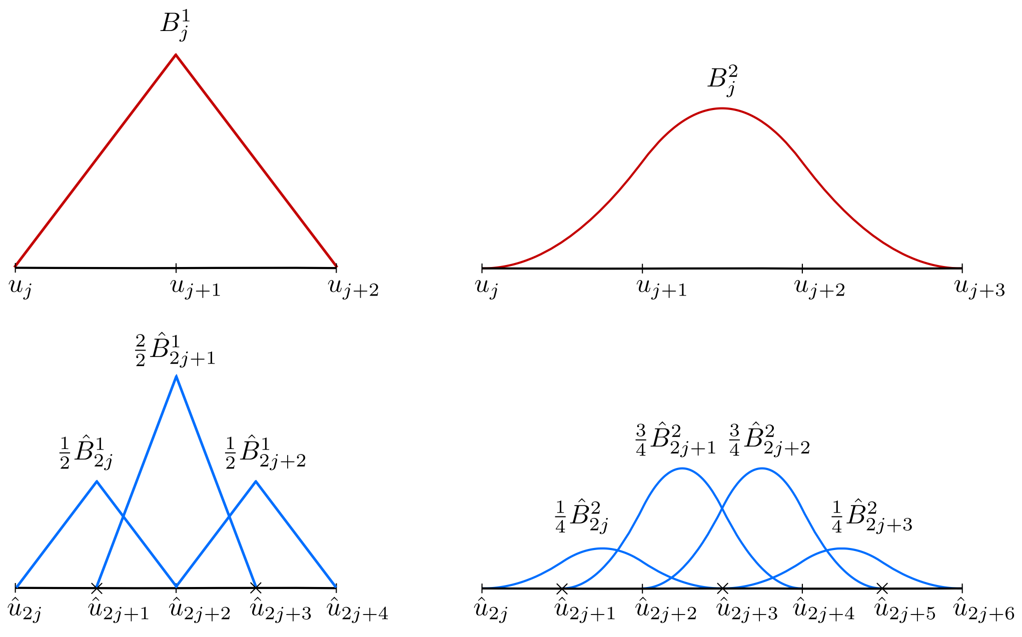
The $n+2$ B-splines $\hat{B}^n_{2j+k}$ for $k \in \{0,1,2,\cdots,n+1\}$ on the refined knot vector $\hat{U} = \{0,1/2,1,\cdots\}$ which can reproduce the B-spline B^n_j are called the children B-splines of B^n_j. For example, each linear B-spline $B^1_j(u)$ can be represented by its three children as

$$B^1_j(u) = \frac{1}{2}\left(\hat{B}^1_{2j}(u) + 2\hat{B}^1_{2j+1}(u) + \hat{B}^1_{2j+2}(u)\right) ,$$

and each quadratic B-spline $B^2_j(u)$ can be represented by its four children as

$$B^2_j(u) = \frac{1}{4}\left(\hat{B}^2_{2j}(u) + 3\hat{B}^2_{2j+1}(u) + 3\hat{B}^2_{2j+2}(u) + \hat{B}^2_{2j+3}(u)\right) .$$

Figure 2.7 plots a linear B-spline and a quadratic B-spline as well as their children B-splines.



(a) Linear **(b)** Quadratic

Figure 2.7: B-splines $B^n_j(u)$ defined on a knot vector U can be represented by a linear combination of their children B-splines $\hat{B}_{2j+k}(u)$ defined on the bisected knot vector $\hat{U}$.

2.2.2 Evaluation by de Boor's algorithm

Consider a knot vector $U = \{u_0, u_1, \cdots, u_n, \cdots, u_{n+k}, \cdots, u_{2n+k}\}$ defining a degree-n B-splines basis $B^n_j(u)$, $j \in \{0,1,\cdots,n+k-1\}$. The B-spline basis is complete over the domain $u \in [u_n, u_{n+k}]$ consisting of k intervals $[u_j, u_{j+1}]$, $j \in \{n,1,\cdots,n+k-1\}$. For example, in Figure 2.6 the knot vector is $U = \{u_0, u_1, \cdots, u_7\}$ and $k = 3$ for the quadratic $n = 2$ plot.

A B-spline curve of degree-n, which is essentially a piecewise polynomial consisting of k polynomial segments connected C^{n-1} continuously at the knots, can be represented as

$$\boldsymbol{c}(u) = \sum_{j=0}^{n+k-1} B_j^n(u)\boldsymbol{p}_j = \boldsymbol{B}^{\mathsf{T}}\boldsymbol{p}, \tag{2.23}$$

where $\boldsymbol{B}$ is a vector containing all the B-spline basis functions $B_j^n(u)$ and $\boldsymbol{p}$ a vector containing all control points $\boldsymbol{p}_j$. Since the B-spline basis functions $B_j^n(u)$ are uniquely determined by the knot vector U and the prescribed degree n, the B-spline curve $\boldsymbol{c}(u)$ of a given degree depends only on the knots and the control points $\boldsymbol{p}_j$.

The de Boor algorithm [26] points out an efficient way to evaluate the B-spline curve at any parameter value by recursive averages. It is considered as the generalisation of de Casteljau algorithm for Bézier curve evaluation to B-splines. Using de Boor's algorithm, we evaluate a B-spline recursively as follows:

$$\begin{aligned}
\boldsymbol{p}_j^0(u) &= \boldsymbol{p}_j, \quad j = 0, \dots, n, \\
\boldsymbol{p}_j^r(u) &= \frac{u_{j+n+1-r} - u}{u_{j+n+1-r} - u_j}\boldsymbol{p}_{j-1}^{r-1} + \frac{u - u_j}{u_{j+n+1-r} - u_j}\boldsymbol{p}_j^{r-1}, r = 1, \cdots, n, \quad j = r, \cdots, n, \\
\boldsymbol{c}(u) &= \boldsymbol{p}_n^n(u).
\end{aligned} \tag{2.24}$$

A graphical illustration of the recursive evaluation process given by de Boor's algorithm (2.24) is provided in Figure 2.8 taking cubic B-splines as an example.

$$\begin{tikzcd}
 & & & \boldsymbol{p}_3^3 & & & \\
 & & \boldsymbol{p}_2^2 \arrow[ur, "\frac{u_4-u}{u_4-u_3}"] & & \boldsymbol{p}_3^2 \arrow[ul, "\frac{u-u_3}{u_4-u_3}"'] & & \\
 & \boldsymbol{p}_1^1 \arrow[ur, "\frac{u_4-u}{u_4-u_2}"] & & \boldsymbol{p}_2^1 \arrow[ul, "\frac{u-u_2}{u_4-u_2}"'] \arrow[ur, "\frac{u_5-u}{u_5-u_3}"] & & \boldsymbol{p}_3^1 \arrow[ul, "\frac{u-u_3}{u_5-u_3}"'] & \\
\boldsymbol{p}_0^0 \arrow[ur, "\frac{u_4-u}{u_4-u_1}"] & & \boldsymbol{p}_1^0 \arrow[ul, "\frac{u-u_1}{u_4-u_1}"'] \arrow[ur, "\frac{u_5-u}{u_5-u_2}"] & & \boldsymbol{p}_2^0 \arrow[ul, "\frac{u-u_2}{u_5-u_2}"'] \arrow[ur, "\frac{u_6-u}{u_6-u_3}"] & & \boldsymbol{p}_3^0 \arrow[ul, "\frac{u-u_3}{u_6-u_3}"'] \\
\boldsymbol{p}_0 \arrow[u, dashed] & & \boldsymbol{p}_1 \arrow[u, dashed] & & \boldsymbol{p}_2 \arrow[u, dashed] & & \boldsymbol{p}_3 \arrow[u, dashed]
\end{tikzcd}$$

Figure 2.8: Illustration of de Boor's algorithm, taking as an example the evaluation of a cubic B-spline curve segment over the interval $[u_3, u_4]$.

2.2.3 Knot insertion by Boehm's algorithm

Knot insertion is the most important algorithm related to the application of B-splines in computer-aided geometric design. For example, it will help us to understand the subdivision technique in curve/surface modelling.

Given a knot vector $U = \{\cdots, u_j, u_{j+1}, u_{j+2}, \cdots\}$, a new knot vector $\hat{U}$ can be obtained by inserting a new knot $\hat{u}$ anywhere as long as the non-decreasing order is still maintained. Knot insertion involves representing the original B-spline curve defined by the knot vector U with the refined knot vector $\hat{U}$, i.e.,

$$\boldsymbol{c}(u) = \boldsymbol{B}^{\mathsf{T}}\boldsymbol{p} = \hat{\boldsymbol{B}}^{\mathsf{T}}\hat{\boldsymbol{p}}, \tag{2.25}$$

where $\boldsymbol{B}$ is a vector containing all the B-spline basis functions $B_j^n(u)$ defined over the original knot vector U and $\hat{\boldsymbol{B}}$ a vector of B-spline basis functions $\hat{B}_j^n(u)$ of the same degree but defined over the new knot vector $\hat{U} = U \cup \{\hat{u}\}$. Assume we insert a new knot $\hat{u}$ with $u_l \leq \hat{u} \leq u_{l+1}$, which results in a new knot vector $\hat{U}$ with knots

$$\hat{u}_j = u_j \quad \text{for} \quad j \leq l, \quad \hat{u}_{l+1} = \hat{u}, \quad \hat{u}_j = u_{j-1} \quad \text{for} \quad j \geq l+2. \tag{2.26}$$

According to Boehm's knot insertion algorithm [27], the B-spline basis functions before and after knot insertion are related by the following linear relation:

$$B_j^n(u) = \hat{\alpha}_j^n \hat{B}_j^n(u) + (1 - \hat{\alpha}_{j+1}^n)\hat{B}_{j+1}^n(u), \tag{2.27a}$$

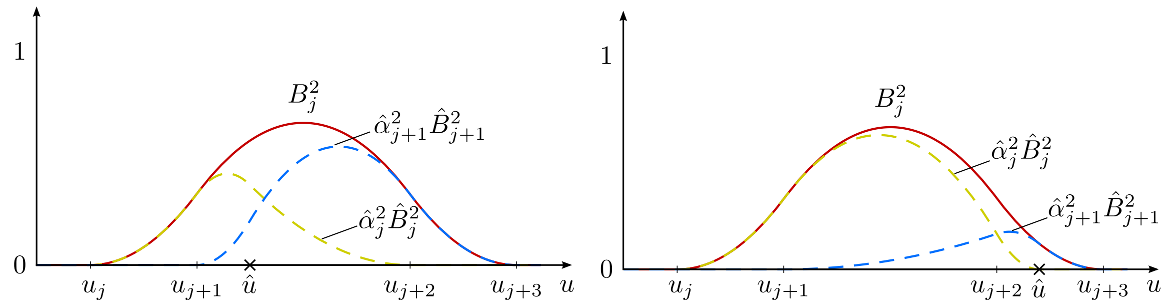
with

$$\hat{\alpha}_j^n = \begin{cases} 1, & j \leq l-n \\ \frac{\hat{u}-\hat{u}_j}{\hat{u}_{j+n+1}-\hat{u}_j} = \frac{\hat{u}-u_j}{u_{j+n}-u_j}, & l-n < j < l+1 \\ 0, & j \geq l+1 \end{cases}. \tag{2.27b}$$

Figure 2.9 takes quadratic B-splines as an example to illustrate the linear relation (2.27) due to knot insertion.

Given (2.25), it is clear that the relation between the B-splines $\boldsymbol{B}$ and $\hat{\boldsymbol{B}}$ also relates the control points $\hat{\boldsymbol{p}}$ after knot insertion to original control points $\boldsymbol{p}$. Boehm's insertion algorithm also gives

$$\hat{\boldsymbol{p}}_j = \hat{\alpha}_j^n \boldsymbol{p}_j + (1 - \hat{\alpha}_j^n)\boldsymbol{p}_{j-1}, \tag{2.28}$$



(a) New knot $\hat{u}$ inserted to the middle knot interval **(b)** New knot $\hat{u}$ inserted to the last knot interval

Figure 2.9: A quadratic B-spline $B_j^2(u)$ defined by the original knot vector U is a linear combination of the new quadratic B-splines $\hat{B}_j^2(u)$ and $\hat{B}_{j+1}^2(u)$ defined by the new knot vector $\hat{U} = U \cup \{\hat{u}\}$.

where the coefficients $\hat{\alpha}_j^n$ are computed as (2.27b). This means we can keep the shape of the represented B-spline curve during the knot insertion assuming the new control points are chosen as affine averages of original control points according to (2.28).

In Figure 2.10, we take a cubic B-spline as an example to illustrate Boehm's knot insertion algorithm (2.27) and (2.28). In this example, we insert a new knot $\hat{u}$ to the knot interval $[u_3, u_4]$ and consequently create a refined knot vector $\hat{U} = \{u_0, \cdots, u_3, \hat{u}, u_4, \cdots\}$. Notice the similarity between Figure 2.10a and the bottom part of the de Boor algorithm in Figure 2.8. In fact, it has been pointed out in [28] that Boehm's knot insertion algorithm is de Boor's algorithm. Boehm's knot insertion algorithm can be applied repeatedly to insert multiple knots one by one. As an application of Boehm's knot insertion algorithm, we are going to use repeated knot insertion to show the two-scale relation (2.22) for uniform cubic B-splines.

$$\begin{tikzcd}
\hat{\boldsymbol{p}}_0 & \hat{\boldsymbol{p}}_1 & & \hat{\boldsymbol{p}}_2 & & \hat{\boldsymbol{p}}_3 & \hat{\boldsymbol{p}}_4 & \cdots & \hat{\boldsymbol{p}}_{j+1} \\
 & & & & & & & \cdots & \\
\boldsymbol{p}_0 \arrow[uu, dashed] \arrow[uur, "\frac{u_4-\hat{u}}{u_4-u_1}"] & & \boldsymbol{p}_1 \arrow[uul, "\frac{\hat{u}-u_1}{u_4-u_1}"'] \arrow[uur, "\frac{u_5-\hat{u}}{u_5-u_2}"] & & \boldsymbol{p}_2 \arrow[uul, "\frac{\hat{u}-u_2}{u_5-u_2}"'] \arrow[uur, "\frac{u_6-\hat{u}}{u_6-u_3}"] & & \boldsymbol{p}_3 \arrow[uul, "\frac{\hat{u}-u_3}{u_6-u_3}"'] \arrow[uu, dashed] & \cdots & \boldsymbol{p}_j \arrow[uu, dashed]
\end{tikzcd}$$

(a) Control points before (bottom) and after (top) knot insertion

$$\begin{tikzcd}
\hat{B}_0^3(u) \arrow[dd, dashed] & \hat{B}_1^3(u) \arrow[ddl] \arrow[ddr] & & \hat{B}_2^3(u) \arrow[ddl] \arrow[ddr] & & \hat{B}_3^3(u) \arrow[ddl] \arrow[ddr] & \hat{B}_4^3(u) \arrow[dd, dashed] & \cdots & \hat{B}_{j+1}^3(u) \arrow[dd, dashed] \\
 & & & & & & & \cdots & \\
B_0^3(u) & & B_1^3(u) & & B_2^3(u) & & B_3^3(u) & \cdots & B_j^3(u)
\end{tikzcd}$$

(b) B-spline basis functions before (bottom) and after (top) knot insertion

Figure 2.10: Illustration of the Boehm algorithm for B-spline knot insertion, taking as an example inserting a new knot $\hat{u}$ into the interval $[u_3, u_4]$ and subdividing the cubic B-spline curve segment into two. Dashed arrow means weight is 1. Omitted weights in (b) are the same as in (a).

Two-scale relation as repeated knot insertion

In Section 2.2.1, we have mentioned the two-scale relation that a uniform B-spline of degree n can be reproduced by its $n+2$ children B-splines defined on the bisected knot vector. Now we are going to show the two-scale relation is a result of repeated knot insertion.

Let us consider a uniform knot vector $U = \{0,1,2,3,4\}$ which defines one cubic B-spline $B_0^3(u)$. Bisecting the knot vector means inserting four knots $\{\frac{1}{2}, \frac{3}{2}, \frac{5}{2}, \frac{7}{2}\}$. We can apply Boehm's knot insertion algorithm repeatedly to insert the new knots one by one. Figure 2.11 illustrates the procedure of repeated knot insertion. For notation simplicity, we denote each B-spline $B_j^3(u)$ by its five knots. For example, 01234 at the top of Figure 2.11 represents the initial uniform B-spline $B_0^3(u)$. We first insert the knot $\frac{3}{2}$ and end up with two B-splines $01\frac{3}{2}23$ and $1\frac{3}{2}234$. The diagram means

$$B^3_{01234}(u) = \frac{\frac{3}{2}-0}{3-0} B^3_{01\frac{3}{2}23}(u) + \frac{4-\frac{3}{2}}{4-1} B^3_{1\frac{3}{2}234}(u) = \frac{1}{2} B^3_{01\frac{3}{2}23}(u) + \frac{5}{6} B^3_{1\frac{3}{2}234}(u) \tag{2.29}$$

which is computed from (2.27). As shown in Figure 2.11, the new knot $\frac{5}{2}$ is inserted at the second step (from second to third row). After that, two knots $\frac{1}{2}$ and $\frac{7}{2}$ are inserted at the same time because they are so far away that the insertion of one knot does not interfere with the insertion of the other. From the tree structure in Figure 2.11, we can compute the contribution from each child B-spline to the original uniform cubic B-spline by multiplying the weights associated with relevant branches. For example, the contribution from the first child B-spline $\hat{B}_0^3(u)$ denoted by five knots $0\frac{1}{2}1\frac{3}{2}2$ is computed as

$$\frac{\frac{1}{2}-0}{2-0} \cdot \frac{\frac{5}{2}-0}{\frac{5}{2}-0} \cdot \frac{\frac{3}{2}-0}{3-0} = \frac{1}{8}. \tag{2.30}$$

As a result of the repeated knot insertion shown in Figure 2.11, we obtain

$$B_0^3(u) = \frac{1}{8}\left(\hat{B}_0^3(u) + 4\hat{B}_1^3(u) + 6\hat{B}_2^3(u) + 4\hat{B}_3^3(u) + \hat{B}_4^3(u)\right), \tag{2.31}$$

which agrees with the two-scale relation

$$B_j^n(u) = \sum_{k=0}^{n+1} S_{j,k}^n \hat{B}_{2j+k}^n(u), \quad \text{with} \quad S_{j,k}^n = \frac{1}{2^n}\binom{n+1}{k}. \tag{2.32}$$

Figure 2.12 plots a uniform cubic B-spline and its five children B-splines scaled with the coefficients given in (2.31).

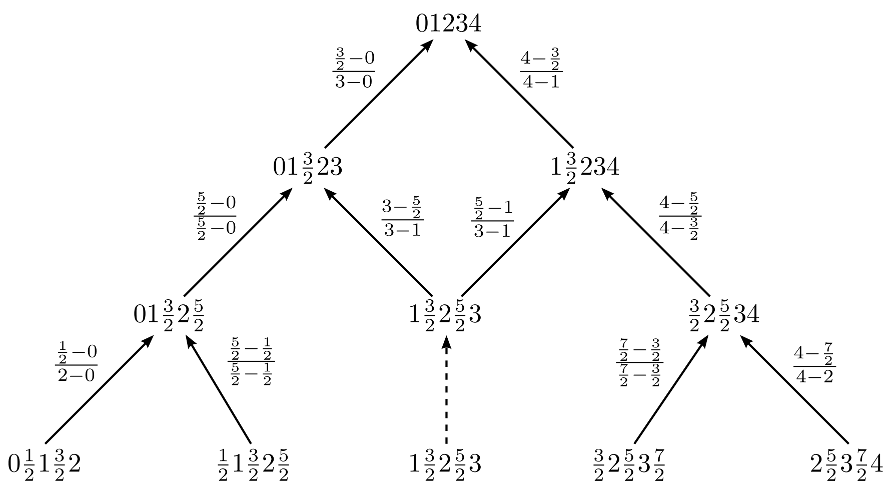

01234
$\frac{\frac{3}{2}-0}{3-0}$
$\frac{4-\frac{3}{2}}{4-1}$
$01\frac{3}{2}23$
$1\frac{3}{2}234$
$\frac{\frac{5}{2}-0}{\frac{5}{2}-0}$
$\frac{3-\frac{5}{2}}{3-1}$
$\frac{\frac{5}{2}-1}{3-1}$
$\frac{4-\frac{5}{2}}{4-\frac{3}{2}}$
$01\frac{3}{2}2\frac{5}{2}$
$1\frac{3}{2}2\frac{5}{2}3$
$\frac{3}{2}2\frac{5}{2}34$
$\frac{\frac{1}{2}-0}{2-0}$
$\frac{\frac{5}{2}-\frac{1}{2}}{\frac{5}{2}-\frac{1}{2}}$
$\frac{\frac{7}{2}-\frac{3}{2}}{\frac{7}{2}-\frac{3}{2}}$
$\frac{4-\frac{7}{2}}{4-2}$
$0\frac{1}{2}1\frac{3}{2}2$
$\frac{1}{2}1\frac{3}{2}2\frac{5}{2}$
$1\frac{3}{2}2\frac{5}{2}3$
$\frac{3}{2}2\frac{5}{2}3\frac{7}{2}$
$2\frac{5}{2}3\frac{7}{2}4$

Figure 2.11: Knot insertion algorithm applied repeatedly to insert new knots $\{\frac{1}{2},\frac{3}{2},\frac{5}{2},\frac{7}{2}\}$ to the uniform knot vector $U=\{0,1,2,3,4\}$. The new knot vector $\hat{U}=\{0,\frac{1}{2},1,\cdots,4\}$ defines five cubic B-splines $\hat{B}_j^3(u)$. All cubic B-splines are labelled by their five knots. The original B-spline and its five children B-splines defined on the bisected new knot sequence $\hat{U}$ are related by $B_0^3(u)=\frac{1}{8}\left(\hat{B}_0^3(u)+4\hat{B}_1^3(u)+6\hat{B}_2^3(u)+4\hat{B}_3^3(u)+\hat{B}_4^3(u)\right)$.

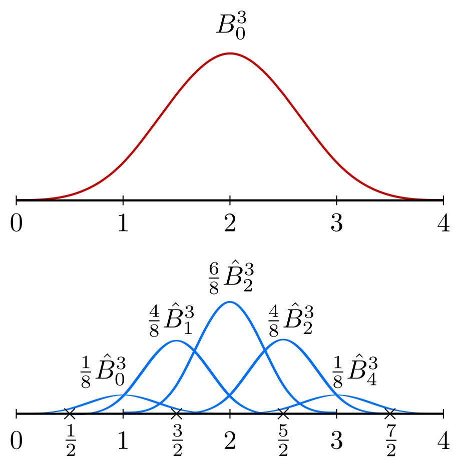


Figure 2.12: Illustration of the two-scale relation for uniform cubic B-splines, i.e., $B_0^3(u)=\frac{1}{8}\left(\hat{B}_0^3(u)+4\hat{B}_1^3(u)+6\hat{B}_2^3(u)+4\hat{B}_3^3(u)+\hat{B}_4^3(u)\right)$.

2.3 Subdivision curves

Subdivision curves, as the univariate counterpart of subdivision surfaces, are methods of representing smooth curves by specifying a coarse piecewise linear control polygon. The represented smooth curves can be calculated from the initial coarse control polygon as the limit curves of recursive subdivision of each polygon segment into two. The first algorithm for generating smooth curves through successive subdivision is known as Chaikin's corner-cutting algorithm [29], which has been realised to generate quadratic B-spline curves [30].

In fact, uniform B-splines of any degree n can be constructed as the limit curves of recursive Lane-Riesenfeld subdivision [31] with each subdivision step involving a refine stage followed by $n-1$ smooth stages.

In this section, we only introduce two subdivision schemes generating uniform and non-uniform cubic B-splines, respectively. They are the univariate instances of Catmull-Clark subdivision [14] and NURBS-compatible subdivision schemes [32].

Central-knot labelling

In previous sections, we label each B-spline by its first knot such that a linear B-spline $B_j^1(u)$ is defined by knots $\{u_j, u_{j+1}, u_{j+2}\}$, see Figure 2.7. In implementation of subdivision curves/surfaces as well as finite element analysis, however, it is more intuitive to label a B-spline of odd degree by its central knot such that $B_j^1(u)$ is define by knots $\{u_{j-1}, u_j, u_{j+1}\}$. In this case, the two-scale relation for linear uniform B-splines shown in Figure 2.7 becomes

$$B_j^1(u) = \frac{1}{2}\left(\hat{B}_{2j-1}^1(u) + 2\hat{B}_{2j}^1(u) + \hat{B}_{2j+1}^1(u)\right) . \tag{2.33}$$

The central-knot labelling is used in the rest of this thesis.

2.3.1 Subdivision generating uniform cubic B-splines

For uniform B-splines, the knots are equidistant in the parameter space. To keep the uniformity at each subdivision step we bisect the knot vector U by inserting a new knot in the middle of each knot interval. With the aforementioned central-knot labelling, the two-scale relation (2.22) in terms of uniform cubic B-splines becomes

$$B_j^3(u) = \frac{1}{8}\left(\hat{B}_{2j-2}^3(u) + 4\hat{B}_{2j-1}^3(u) + 6\hat{B}_{2j}^3(u) + 4\hat{B}_{2j+1}^3(u) + \hat{B}_{2j+2}^3(u)\right) . \tag{2.34}$$

As we only discuss subdivision for cubic B-splines, the superscript denoting degree is omitted for simplicity. Then the two scale-relation (2.34) can be written in a compact form as

$$\boldsymbol{B} = \boldsymbol{S}^{\mathsf{T}}\hat{\boldsymbol{B}}. \tag{2.35}$$

where $\boldsymbol{B}$ is a vector containing the B-spline basis functions before subdivision and $\hat{\boldsymbol{B}}$ containing the B-spline basis after subdivision. The matrix $\boldsymbol{S}$ stores all the coefficients in (2.34). Recall that subdivision keeps the represented B-spline curves unchanged, i.e.,

$$\boldsymbol{c}(u) = \boldsymbol{B}^{\mathsf{T}}\boldsymbol{p} = \hat{\boldsymbol{B}}^{\mathsf{T}}\boldsymbol{S}\boldsymbol{p} = \hat{\boldsymbol{B}}^{\mathsf{T}}\hat{\boldsymbol{p}}. \tag{2.36}$$

Given that $\hat{\boldsymbol{B}}$ represents a linearly independent B-spline basis, (2.36) results in

$$\hat{\boldsymbol{p}} = \boldsymbol{S}\boldsymbol{p}. \tag{2.37}$$

$$\boldsymbol{S} = \frac{1}{8}\begin{bmatrix} \ddots & & & & & \\ 1 & 6 & 1 & 0 & 0 & 0 \\ 0 & 4 & 4 & 0 & 0 & 0 \\ 0 & 1 & 6 & 1 & 0 & 0 \\ 0 & 0 & 4 & 4 & 0 & 0 \\ 0 & 0 & 1 & 6 & 1 & 0 \\ 0 & 0 & 0 & 4 & 4 & 0 \\ 0 & 0 & 0 & 1 & 6 & 1 \\ & & & & & \ddots \end{bmatrix} \tag{2.38}$$

Subdivision matrix: A matrix relates the B-splines and control points before and after refinement such that $\boldsymbol{B} = \boldsymbol{S}^{\mathsf{T}}\hat{\boldsymbol{B}}$ and $\hat{\boldsymbol{p}} = \boldsymbol{S}\boldsymbol{p}$.

Mask: Each column of the subdivision matrix contains the set of weights indicating the influence of a specific old control point on new ones, e.g., $[1\ 4\ 6\ 4\ 1]/8$ for cubic B-splines shown in Figure 2.13. In the case of subdivision of uniform B-splines of degree n, without considering the influence of boundaries, the mask is unique.

Stencils: Each row of the subdivision matrix contains the set of weights indicating the contributions from multiple old control points to a specific new control point, e.g., $[4\ 4]/8$ for new control points and $[1\ 6\ 1]/8$ for existing control points shown in Figure (2.14). The weights in each stencil sum up to one.

Notice that stencils in Figure 2.14 and the mask in Figure 2.13 express the same subdivision weights in (2.38). In terms of implementation, it is also common to apply one subdivision refinement of cubic uniform B-splines in two steps using a variant of stencils given in Figure 2.15. In this thesis, we use stencils or their derivatives to define a subdivision scheme.

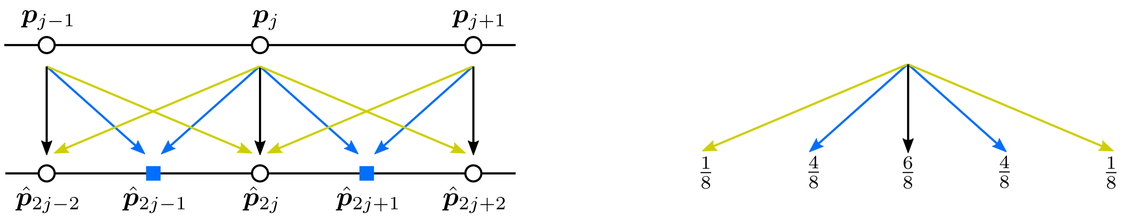
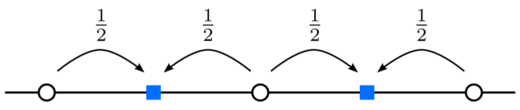
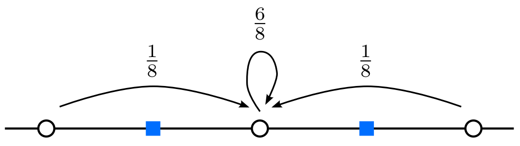


Figure 2.13: Graphical illustration of univariate mask for cubic uniform B-splines. The mask indicates the influence of each old control point (empty circle) on the new ones after subdivision.

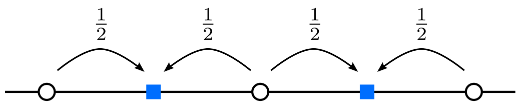


(a) Stencil for new face vertices

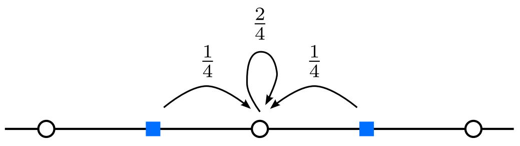


(b) Stencil for existing vertices

Figure 2.14: Graphical illustration of univariate stencils for cubic uniform B-splines. (a): Stencil for computing the positions of newly inserted control points (blue squares). (b) Stencil for computing the new positions of existing control points (empty circles).



(a) Step 1: insert new edge vertices



(b) Step 2: update existing vertices

Figure 2.15: Implementing one subdivision refinement of cubic B-splines in two steps: (a) new vertices are inserted to middle of each edge; (b) existing vertices are computed as weighted average of two adjacent new edge vertices and itself.

2.3.2 Subdivision generating non-uniform cubic B-splines

Though uniform B-splines are popular in most implementations, non-uniform knot vectors are needed to model sharp features or fine features. Non-uniform knots are also used to interpolate endpoints. A univariate subdivision scheme corresponding to non-uniform B-splines of any degree is proposed in [33]. It also provides the flexibility of selective knot insertion meaning that zero or one knot is inserted into any non-zero knot intervals in one subdivision step. One subdivision step is factorised to a refine stage and several smooth stages. Given the B-spline degree n, a refine stage and $\lfloor n/2 \rfloor$ smooth stages are needed to compute the new control points after subdivision. The derivation is related to the *blossom* or *polar form* of a degree-n polynomial $p(t)$, which is a unique symmetric multiaffine polynomial $B_p(t_1,t_2,\cdots,t_n)$ such that $B_p(t,\cdots,t) = p(t)$. Readers who are interested in the blossom form for B-splines are recommended to read [28] and the references therein. Here, we simply point out that the B-spline control point $\boldsymbol{p}_j$, which is corresponding to the B-spline basis $B_j^3(u)$ defined by five knots $\{u_{j-2},u_{j-1},u_j,u_{j+1},u_{j+2}\}$, is given by the blossom evaluated at three consecutive knots $B_p(u_{j-1},u_j,u_{j+1})$. During the subdivision refinement, the B-spline curve stays unchanged, meaning that the unique blossom form also stays unchanged. Therefore, each control point $\boldsymbol{p}_j$ is uniquely determined by the three knots $\{u_{j-1},u_j,u_{j+1}\}$, which we could use to label each control point by omitting the blossom symbol.

We follow the selective knot insertion subdivision algorithm given in [33] to derive the subdivision weights for cubic non-uniform B-splines. For cubic B-splines with $n = 3$, only one refine stage and one smooth stage are required. Consider a knot vector U of nine knots and insert two knots $\hat{u}_\alpha \in [u_3, u_4]$ and $\hat{u}_\beta \in [u_4, u_5]$ as below:

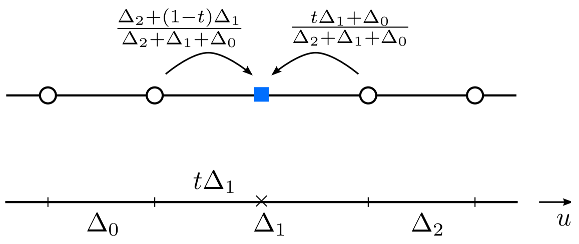
$$\begin{aligned} U &= \{u_0\ u_1\ u_2\ u_3 \qquad u_4 \qquad u_5\ u_6\ \ u_7\ u_8\} \\ \hat{U} \setminus U &= \{ \qquad\qquad\qquad \hat{u}_\alpha \qquad \hat{u}_\beta \qquad\qquad\quad \} \end{aligned}.$$

Figure 2.16 illustrates one subdivision step. At the refine stage, new control points denoted by $u_3\hat{u}_\alpha u_4$ and $u_4\hat{u}_\beta u_5$ are inserted as affine averages of the two adjacent control points. At this stage, new knots are only allowed to be inserted to the middle of three blossom entries. Now we have the correct number of control points but the knots used as blossom entries are not consistent with the refined knot vector $\hat{U}$. Therefore, at the smooth stage, we update with control points by inserting the knots as required, where we encounter three cases: a knot inserted on the left and a knot inserted on the right or two knots inserted on each side. Again, every knot insertion results in an affine average.

$$\begin{tikzcd}
 & u_1u_2u_3 & u_2u_3\hat{u}_\alpha & u_3\hat{u}_\alpha u_4 & \hat{u}_\alpha u_4\hat{u}_\beta & u_4\hat{u}_\beta u_5 & \hat{u}_\beta u_5u_6 & u_5u_6u_7 \\
\text{Smooth:} & & & & & & & \\
 & u_1u_2u_3 \arrow[uu, dashed] \arrow[uur, "\frac{u_4-\hat{u}_\alpha}{u_4-u_1}"] & u_2u_3u_4 \arrow[uu, "\frac{\hat{u}_\alpha-u_1}{u_4-u_1}"'] & u_3\hat{u}_\alpha u_4 \arrow[uu, dashed] \arrow[uur, "\frac{u_5-\hat{u}_\beta}{u_5-u_3}"] & u_3u_4u_5 \arrow[uu, "\frac{\hat{u}_\beta-\hat{u}_\alpha}{u_5-u_3}"] & u_4\hat{u}_\beta u_5 \arrow[uu, dashed] \arrow[uul, "\frac{\hat{u}_\alpha-u_3}{u_5-u_3}"'] & u_4u_5u_6 \arrow[uu, "\frac{u_7-\hat{u}_\beta}{u_7-u_4}"] & u_5u_6u_7 \arrow[uu, dashed] \arrow[uul, "\frac{\hat{u}_\beta-u_4}{u_7-u_4}"'] \\
\text{Refine:} & & & & & & & \\
 & \boldsymbol{p}_2 \arrow[uu, dashed] & \boldsymbol{p}_3 \arrow[uu, dashed] \arrow[uur, "\frac{u_5-\hat{u}_\alpha}{u_5-u_2}"] & & \boldsymbol{p}_4 \arrow[uul, "\frac{\hat{u}_\alpha-u_2}{u_5-u_2}"'] \arrow[uu, dashed] \arrow[uur, "\frac{u_6-\hat{u}_\beta}{u_6-u_3}"] & & \boldsymbol{p}_5 \arrow[uul, "\frac{\hat{u}_\beta-u_3}{u_6-u_3}"'] \arrow[uu, dashed] & \boldsymbol{p}_6 \arrow[uu, dashed]
\end{tikzcd}$$

Figure 2.16: Graphical illustration of the refine and smooth subdivision algorithm for selective knot insertions of non-uniform cubic B-splines. The weights for affine averages are marked on each branch. The dashed lines with omitted 1 as weights mean copy directly.

From the tree structure and the given weights in Figure 2.16, we can extract the subdivision weights for the computation of new edge vertices,such as $u_3\hat{u}_\alpha u_4$ and $u_4\hat{u}_\beta u_5$. As shown in Figure 2.17, the computation is regardless of the situation of knot insertion to adjacent edges. Instead, the position of a new control vertex only depends on the positions of two adjacent control points, the ratios of three relevant knot intervals and the location where the new knot is inserted. For the subdivision of uniform B-splines, knot intervals are all equal and each knot interval is bisected, which means $\Delta_1 = \Delta_2 = \Delta_3$ and $t = 0.5$. In this case, the weights in Figure 2.17 become $1/2$ and $1/2$, identical to the subdivision weights for cubic uniform B-splines in Figure 2.15a.



$\frac{\Delta_2+(1-t)\Delta_1}{\Delta_2+\Delta_1+\Delta_0}$
$\frac{t\Delta_1+\Delta_0}{\Delta_2+\Delta_1+\Delta_0}$
$t\Delta_1$
Δ_0
Δ_1
Δ_2
u

Figure 2.17: Graphical illustration of the computation of new control vertices, such as $u_3\hat{u}_\alpha u_4$ and $u_4\hat{u}_\beta u_5$ in Figure 2.16. For cubic B-splines, the position of a new control vertex (blue square) only depends on the positions of two adjacent control points (empty circles) and three relevant knot intervals, regardless of whether there are knot insertions at adjacent edges or not. Note that no knot will be inserted to zero knot interval, meaning that at least $\Delta_1 \neq 0$.

From Figure 2.16, we can also figure out how to compute the positions of existing vertices. Figure 2.18 summarises the computation in four cases as shown in Figure 2.16: (a) no knot inserted on either side, such as $u_1u_2u_3$ and $u_5u_6u_7$; (b) new knots inserted on both sides, such as $\hat{u}_\alpha u_4\hat{u}_\beta$; (c) new knot inserted only on the left, such as $\hat{u}_\beta u_5u_6$; (d) new knot inserted only on the right, such as $u_2u_3\hat{u}_\alpha$. Again, the computation of existing vertices given in Figure 2.18b is identical to the subdivision of uniform cubic B-splines in Figure 2.15b when $\Delta_0 = \Delta_1$ and $t_0 = t_1 = 0.5$.

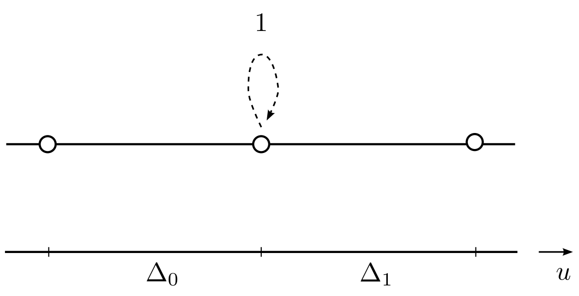
2.4 Boundary conditions and creases for subdivision curves

2.4.1 Boundary conditions

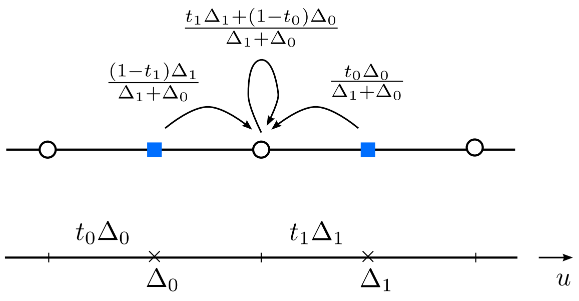
Uniform B-spline basis functions are much simpler in terms of computations, compared with B-splines defined by non-uniform knots. Also, uniform B-splines are adequate for most implementations, such as approximation and finite element analysis. However, uniform B-splines of degree $n \geq 2$ generally do not have the endpoint interpolation property. Figure 2.19a plots a cubic B-spline curve parametrised with uniform B-spline basis functions. It is clear that the endpoints are not interpolated by the B-spline curve. In this section, we discuss several techniques to create endpoint interpolation taking cubic B-splines as an example. See [34] for a summary of different creases and boundary conditions for subdivision curves.

Open knot vectors: Knot vectors are said to be open if the end knots have a multiplicity of $n+1$, where n is the degree of B-spline. An open knot vector looks like:

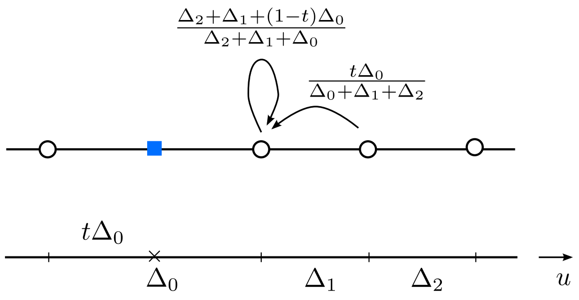
$$U = \{\underbrace{u_0, \cdots, u_0}_{n+1 \text{ times}}, u_1, \cdots, u_{l-1}, \underbrace{u_l, \cdots, u_l}_{n+1 \text{ times}}\}.$$



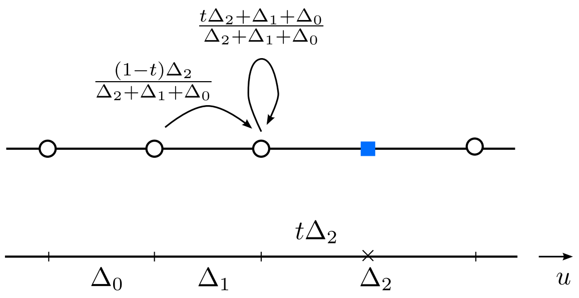
(a) No knot inserted either side


$\frac{t_1\Delta_1+(1-t_0)\Delta_0}{\Delta_1+\Delta_0}$
$\frac{(1-t_1)\Delta_1}{\Delta_1+\Delta_0}$
$\frac{t_0\Delta_0}{\Delta_1+\Delta_0}$
$t_0\Delta_0$
$t_1\Delta_1$
Δ_0
Δ_1
u

(b) New knots inserted both sides

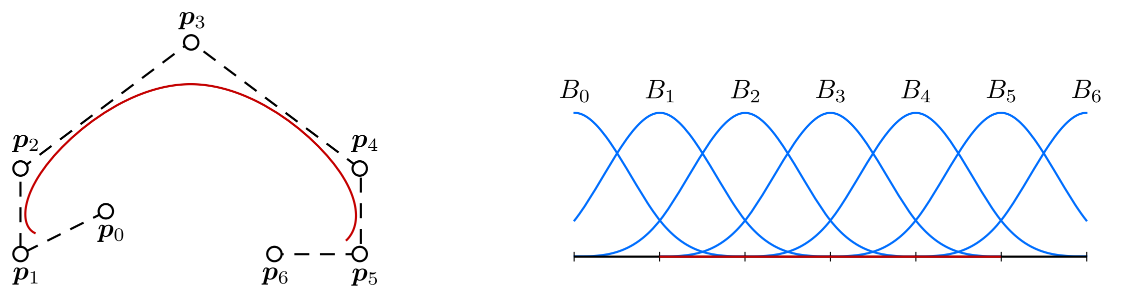

$\frac{\Delta_2+\Delta_1+(1-t)\Delta_0}{\Delta_2+\Delta_1+\Delta_0}$
$\frac{t\Delta_0}{\Delta_0+\Delta_1+\Delta_2}$
$t\Delta_0$
Δ_0
Δ_1
Δ_2
u

(c) New knot inserted left

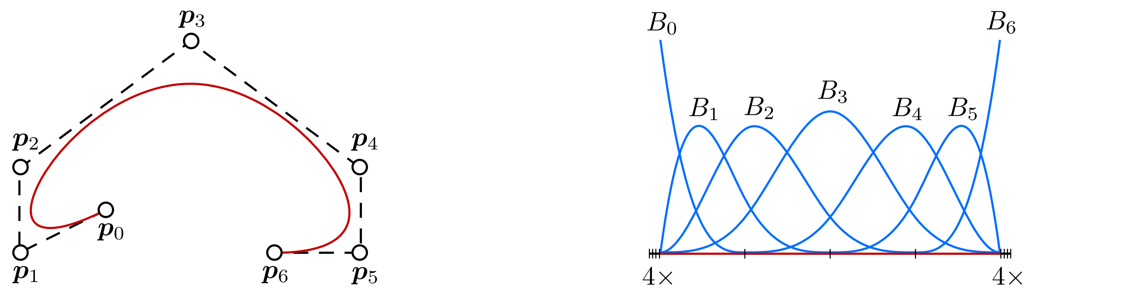

$\frac{t\Delta_2+\Delta_1+\Delta_0}{\Delta_2+\Delta_1+\Delta_0}$
$\frac{(1-t)\Delta_2}{\Delta_2+\Delta_1+\Delta_0}$
$t\Delta_2$
Δ_0
Δ_1
Δ_2
u

(d) New knot inserted right

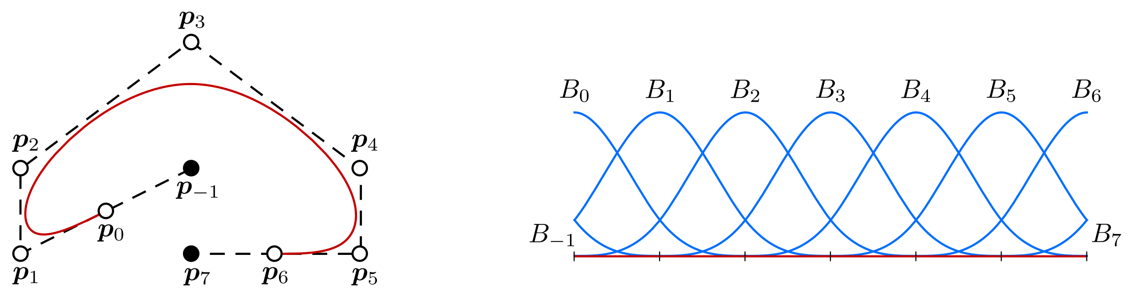
Figure 2.18: Graphical illustration of the computation of existing control vertices. For cubic B-splines, the selective knot insertions result in four different cases as shown in Figure 2.16: (a) no knot inserted on either side, such as $u_1u_2u_3$ and $u_5u_6u_7$; (b) new knots inserted on both sides, such as $\hat{u}_\alpha u_4 \hat{u}_\beta$; (c) new knot inserted only on the left, such as $\hat{u}_\beta u_5u_6$; (d) new knot inserted only on the right, such as $u_2u_3\hat{u}_\alpha$.



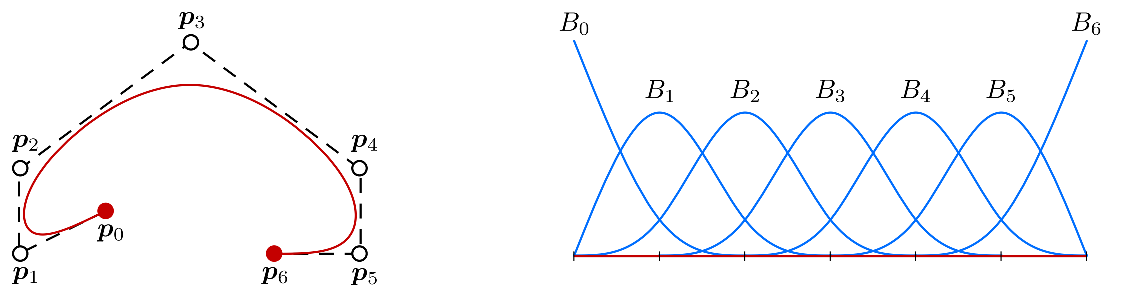
(a) Uniform knot sequence



(b) Open knot sequence



(c) Ghost points



(d) Modified subdivision rules

Figure 2.19: A comparison of various cubic splines (left) and the corresponding basis functions used to parametrise them. (a) A uniform knot sequence leads to a spline curve without end-interpolation properties. (b) An open knot sequence has a multiplicity of $n+1$ at the first and last knot values, generating a spline curve which interpolates the end points. (c) Ghost points (black solid dots), calculated as a linear combination of adjacent control points, are pre- and appended to create end-interpolation spline curve. In this example of cubic splines, only one ghost point at each end is needed, i.e., $\boldsymbol{p}_{-1} = 2\boldsymbol{p}_0 - \boldsymbol{p}_1$ and $\boldsymbol{p}_7 = 2\boldsymbol{p}_6 - \boldsymbol{p}_5$. (d) A modified subdivision rule shown in Figure 2.21 is applied to the end points (red solid dots). Their position is fixed during subdivision, leading to a limit curve interpolating the end points.

Figure 2.19b shows a cubic B-spline curve and the corresponding B-spline basis functions that are defined over an open knot vector $U = \{0,0,0,0,1,2,3,4,4,4,4\}$. The represented curve interpolates the endpoints. As a result of the multiplicity of end knots, the n B-spline basis functions close to each end become non-uniform. In this case, generating such non-uniform B-spline curves by recursive subdivision calls for the weights introduced in Section 2.3.2, see Figure 2.17 and Figure 2.18.

Consider a special open knot vector

$$U = \{\underbrace{0,\cdots,0}_{n+1\text{ times}},\underbrace{1,\cdots,1}_{n+1\text{ times}}\}$$

which has $2(n+1)$ knots and defines $n+1$ B-splines of degree n. These B-splines $B_j^n(u)$ form a complete polynomial basis over the parameter domain $u \in [0,1]$. In this special case, the B-splines $B_j^n(u)$ are actually identical to Bernstein polynomials of degree n, representing Bézier curves. Therefore, the technique of using open knot vectors to define B-splines interpolating at the endpoints is also referred to as Bézier end conditions.

Ghost points: An alternative technique to create a B-spline curve interpolating endpoints is to append ghost control points at both ends of the control polygon. The ghost points are set as linear combinations of control points to fulfil the endpoint interpolation. For example, for cubic B-spline curves one ghost point is needed at each end, see Figure 2.19c. Consider the condition of endpoint interpolation, we have

$$\boldsymbol{c}(u=0) = B_{-1}(0)\boldsymbol{p}_{-1} + B_0(0)\boldsymbol{p}_0 + B_1(0)\boldsymbol{p}_1 = \frac{1}{6}\boldsymbol{p}_{-1} + \frac{4}{6}\boldsymbol{p}_0 + \frac{1}{6}\boldsymbol{p}_1 \tag{2.39a}$$

$$\boldsymbol{c}(u=0) = \boldsymbol{p}_0 \tag{2.39b}$$

which leads to the following formula to compute the position of ghost point

$$\boldsymbol{p}_{-1} = 2\boldsymbol{p}_0 - \boldsymbol{p}_1 \,. \tag{2.40}$$

Similarly, the ghost point on the other end will be computed as

$$\boldsymbol{p}_7 = 2\boldsymbol{p}_6 - \boldsymbol{p}_5 \,. \tag{2.41}$$

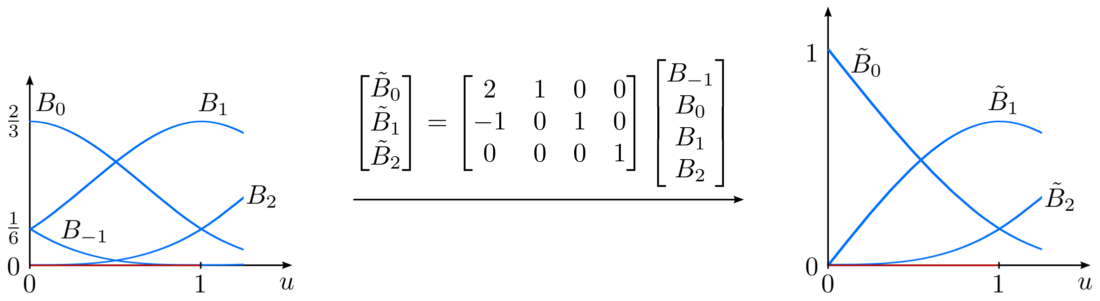


Figure 2.20: A basis transformation resulted from the ghost point constraint. For cubic B-spline, the ghost point is constrained to be $\boldsymbol{p}_{-1} = 2\boldsymbol{p}_0 - \boldsymbol{p}_1$, leading to the transformation shown in this plot. See (2.42) for the derivation of the transformation.

The constraint of ghost point is essentially a transformation of basis functions over the element adjacent to the end.

$$\begin{aligned}\boldsymbol{c}(u) &= B_{-1}(u)\boldsymbol{p}_{-1} + B_0(u)\boldsymbol{p}_0 + B_1(u)\boldsymbol{p}_1 + B_2(u)\boldsymbol{p}_2\\ &= B_{-1}(u)(2\boldsymbol{p}_0 - \boldsymbol{p}_1) + B_0(u)\boldsymbol{p}_0 + B_1(u)\boldsymbol{p}_1 + B_2(u)\boldsymbol{p}_2\\ &= \underbrace{[2B_{-1}(u) + B_0(u)]}_{\tilde{B}_0(u)}\boldsymbol{p}_0 + \underbrace{[B_1(u) - B_{-1}(u)]}_{\tilde{B}_1(u)}\boldsymbol{p}_1 + B_2(u)\boldsymbol{p}_2\end{aligned} \tag{2.42}$$

That is, for cubic B-splines, the effect of adding ghost point leads to the following basis transformation

$$\begin{bmatrix}\tilde{B}_0(u)\\ \tilde{B}_1(u)\\ \tilde{B}_2(u)\end{bmatrix} = \begin{bmatrix}2 & 1 & 0 & 0\\ -1 & 0 & 1 & 0\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}B_{-1}(u)\\ B_0(u)\\ B_1(u)\\ B_2(u)\end{bmatrix} \tag{2.43}$$

which is illustrated in Figure 2.20. Plug the original uniform cubic B-spline basis functions defined as

$$\begin{bmatrix}B_{-1}(u)\\ B_0(u)\\ B_1(u)\\ B_2(u)\end{bmatrix} = \frac{1}{6}\begin{bmatrix}1 & -3 & 3 & -1\\ 4 & 0 & -6 & 3\\ 1 & 3 & 3 & -3\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1\\ u\\ u^2\\ u^3\end{bmatrix} \tag{2.44}$$

into (2.43), we obtain the basis after transformation as follows:

$$\begin{bmatrix}\tilde{B}_0(u)\\ \tilde{B}_1(u)\\ \tilde{B}_2(u)\end{bmatrix} = \frac{1}{6}\begin{bmatrix}6 & -6 & 0 & 1\\ 0 & 6 & 0 & -2\\ 0 & 0 & 0 & 1\end{bmatrix}\begin{bmatrix}1\\ u\\ u^2\\ u^3\end{bmatrix} \tag{2.45}$$

which shows that in the transformed basis $\tilde{B}_j(u), j \in 0,1,2$ the quadratic term u^2 is missing. This means that adding ghost points leads to an incomplete cubic basis, which only has linear approximation order in the knot interval adjacent to endpoint. Adding ghost points does not influence the basis over the effective parameter domain away from the end. In fact, the same basis can be generated by applying modified subdivision rules to the end vertices, as shown in Figure 2.19d.

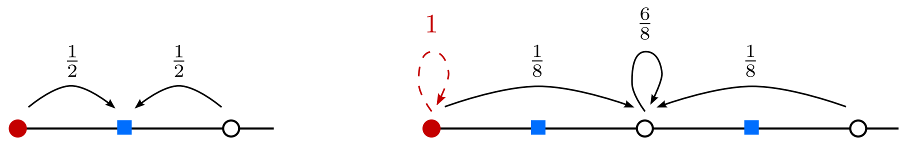
Modified subdivision rule: Another alternative technique to create B-spline curves that fulfil the endpoint interpolation property is to apply a modified subdivision rule to endpoints. This technique is used to generate the cubic B-spline curve shown in Figure 2.19d. For uniform cubic B-spline curves, the subdivision stencils are $[1/2, 1/2]$ for new vertices and $[1/8, 6/8, 1/8]$ for existing vertices (empty circles). In order to interpolate the endpoints, the stencil applied to the end vertices (red dots) is modified to be $[0,1,0]$, see Figure 2.21. Let us call this technique subdivision end conditions. Modifying the normal subdivision rule leads to a different limit curve, which effectively changes uniform B-spline basis to a transformed basis, see Figure 2.19d. The basis is in fact the same as the effective basis generated by adding ghost points. This is easy to verify. Compare Figure 2.19c and Figure 2.19d, except the ghost points all other control points are the same. When we apply the subdivision algorithm once, all control points between $\boldsymbol{p}_0$ and $\boldsymbol{p}_6$ should be the same. What remains in question is the position of endpoints. In fact, in both cases the position of the end control vertices will not change during subdivision. This is obvious when modified subdivision rule is applied. In the case of ghost points, it is also straightforward to show, i.e.,

$$\hat{\boldsymbol{p}}_0 = \begin{bmatrix} 1/8 & 6/8 & 1/8 \end{bmatrix} \begin{bmatrix} \boldsymbol{p}_{-1} \\ \boldsymbol{p}_0 \\ \boldsymbol{p}_1 \end{bmatrix} = \begin{bmatrix} 1/8 & 6/8 & 1/8 \end{bmatrix} \begin{bmatrix} 2\boldsymbol{p}_0 - \boldsymbol{p}_1 \\ \boldsymbol{p}_0 \\ \boldsymbol{p}_1 \end{bmatrix} = \boldsymbol{p}_0 \, . \tag{2.46}$$

Therefore we can conclude that the limit curves in both cases starting at $\boldsymbol{p}_0$ and ending at $\boldsymbol{p}_6$ are exactly the same. Consequently, adding ghost points and modifying subdivision rules result in the same basis transformation. The new basis lacks quadratic terms and is only able to reproduce linear functions.

2.4.2 Creases

The treatment of creases is analogous to boundaries as discussed in Section 2.4.1 in detail. In addition to the following two methods, repetition of control points is another way to create sharp creases [34], which also generates an incomplete basis. Repeating control points is not



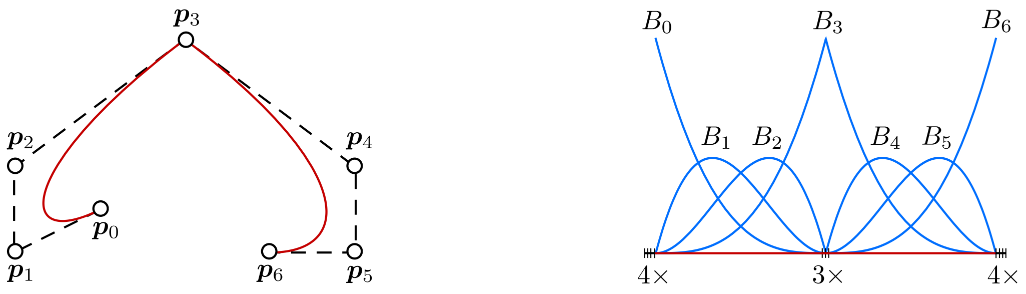
(a) New edge vertices **(b)** Existing vertices

Figure 2.21: Modified stencil $[0,1,0]$ is applied to the endpoint (red solid dot) to fix the endpoint at its initial position during subdivision. Consequently, the limit curve will interpolate the endpoints. All other existing vertices (empty circles) are computed using $[1/8, 6/8, 1/8]$ as normal and new edge vertices (blue squares) with $[1/2, 1/2]$.

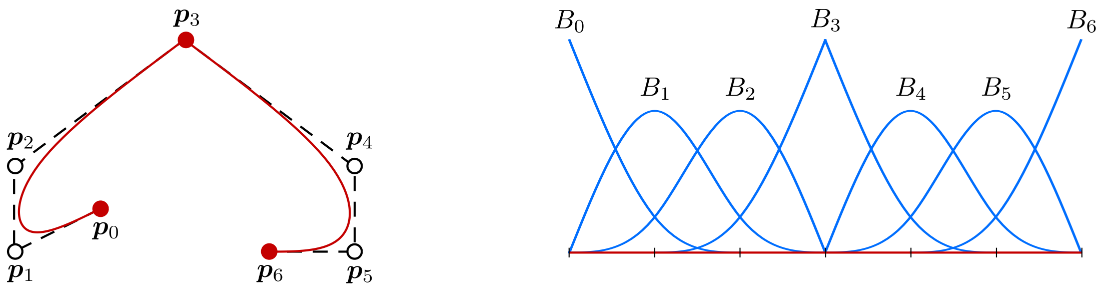
as convenient as the following two techniques in terms of modelling sharp creases in finite element analysis, because the zero element size leads to a singular mapping.

Multiple knots: One way to create degree-n B-spline basis which is only C^0 continuous at a certain interior knot is to repeat that knot to have multiplicity of n, see Figure 2.22a.

Sharp subdivision rules: As proposed in [35, 36], one can introduce sharp features, such as creases and corners, to subdivision surfaces by modifying the subdivision rules in the neighbourhood of a sharp feature. This means an alternative approach to model creases in subdivision curves is to apply modified subdivision rule at creased vertices. For example, for cubic uniform B-spline subdivision, subdivision stencil $[0,1,0]$ is used at the prescribed crease vertex to generate the B-spline curve in Figure 2.22b. As discussed for boundary conditions, modifying the subdivision rule at creases leads to an incomplete basis in the knot intervals adjacent to creases, where only linear functions can be reproduced rather than cubic.



(a) Nonuniform knot sequence



(b) Modified subdivision rule

Figure 2.22: A comparison of various cubic splines (left) and the corresponding basis functions used to parametrise them. (a) An open nonuniform knot sequence has a multiplicity of n at the prescribed crease vertex, generating a spline curve which interpolates the crease vertex $\boldsymbol{p}_3$. (b) A modified subdivision rule similar to Figure 2.21 is applied to the prescribed crease vertices (red solid dots). Their position is fixed during subdivision, leading to a limit curve interpolating the sharp crease vertex $\boldsymbol{p}_3$ and end points.

Chapter 3

Subdivision-based finite elements

In this chapter, we discuss the implementation aspects of using B-spline basis functions in finite element analysis (FEA). Even though our discussion takes as an example Poisson's equation, a second-order partial differential equation, all the basic ideas and concepts are readily extended to FEA of two- or three-dimensional problems described by higher-order partial differential equations.

From this chapter, we introduce the notation more commonly used in finite element analysis. u_i will no longer denote knots in the parameter domain as in the previous chapter. Instead, the knots will be denoted by ξ_i and u_i are the unknown coefficients in finite element approximation of the physical solution u, such as temperature in heat transfer or displacement in elasticity problems.

3.1 Review of finite element method

3.1.1 Strong form and weak form

Let us consider Poisson's equation. The strong form of this boundary value problem is defined as:

$$-\nabla^2 u = f \quad \text{in } \Omega \tag{3.1a}$$

$$u = \overline{u} \quad \text{on } \Gamma_D \tag{3.1b}$$

$$\nabla u \cdot \boldsymbol{n} = t \quad \text{on } \Gamma_N \tag{3.1c}$$

where f is a given source term and u the scalar solution field to be solved. On Dirichlet boundaries Γ_D the value of solution u is known, while on Neumann boundaries Γ_N the flux $\nabla u \cdot \boldsymbol{n}$ with $\boldsymbol{n}$ denoting the outward unit surface normal vector is prescribed instead. Note that

$\Gamma_D \cup \Gamma_N = \partial\Omega$ denotes the complete boundary of the domain Ω. The strong form indicates the valid solution to be twice differentiable.

The weak form of a differential problem is an integral form, from which finite element discretisation is derived. Thus, the first step of implementing FEA is to formulate a weak form equivalent to the strong form of the considered boundary value problem. This is achieved by multiplying the governing equation (3.1a) by an arbitrary test function w from an infinite dimensional space, and integrating over the computational domain Ω, i.e.,

$$-\int_\Omega w \nabla^2 u \, \mathrm{d}\Omega = \int_\Omega w f \, \mathrm{d}\Omega \, . \tag{3.2}$$

Note that the arbitrary test function is required to be zero where Dirichlet boundary conditions are given, i.e.,

$$w = 0 \quad \text{on} \, \Gamma_D \, . \tag{3.3}$$

Apply integration by parts to the term on the left in (3.2) and notice (3.3), we obtain

$$\int_\Omega \nabla w \cdot \nabla u \, \mathrm{d}\Omega = \int_\Omega w f \, \mathrm{d}\Omega + \int_{\Gamma_N} w \nabla u \cdot \boldsymbol{n} \, \mathrm{d}\Gamma \, , \tag{3.4}$$

which indicates that both the test function w and the solution u are only required to be once differentiable and their derivatives are square integrable. The second term on the right naturally allows us to apply the Neumann boundary condition, also called natural boundary conditions. Applying the Neumann boundary condition $\nabla u \cdot \boldsymbol{n} = t$ on Γ_N to (3.4) leads to the weak formulation corresponding to the PDE described in (3.1): Find $u \in \mathscr{X}_{\bar{u}}$ such that

$$\int_\Omega \nabla w \cdot \nabla u \, \mathrm{d}\Omega = \int_\Omega w f \, \mathrm{d}\Omega + \int_{\Gamma_N} w t \, \mathrm{d}\Gamma \quad \forall w \in \mathscr{X}_0 \, , \tag{3.5}$$

where the infinite dimensional function spaces $\mathscr{X}_{\bar{u}}$ and $\mathscr{X}_0$ are both subsets of the standard Hilbert space $\mathscr{H}^1$ restricted to certain boundary conditions, i.e.,

$$\begin{aligned} \mathscr{X}_{\bar{u}} &= \{ u \in \mathscr{H}^1(\Omega) \mid u = \bar{u} \text{ on } \Gamma_D \} \\ \mathscr{X}_0 &= \{ w \in \mathscr{H}^1(\Omega) \mid w = 0 \text{ on } \Gamma_D \} \end{aligned} \, . \tag{3.6}$$

3.1.2 Galerkin approximation

Let $\mathscr{X}^h_{\bar{u}} \subset \mathscr{X}_{\bar{u}}$ and $\mathscr{X}^h_0 \subset \mathscr{X}_0$ denote finite dimensional function spaces. Restricting the weak form (3.5) to the finite dimensional spaces leads to the Galerkin formulation: Find

$u \in \mathscr{X}_{\bar{u}}^h$ such that

$$\int_\Omega \nabla w^h \cdot \nabla u^h \, \mathrm{d}\Omega = \int_\Omega w^h f \, \mathrm{d}\Omega + \int_{\Gamma_N} w^h t \, \mathrm{d}\Gamma \quad \forall w^h \in \mathscr{X}_0^h, \tag{3.7}$$

where u^h is the best approximation of the continuous functions u in the considered finite dimensional space $\mathscr{X}_{\bar{u}}^h$. The approximation u^h is discretised as

$$u^h(\boldsymbol{x}) = \sum_{j=1}^{\mathrm{NP}} N_j(\boldsymbol{x}) u_j, \tag{3.8}$$

where $N_j(\boldsymbol{x})$ are basis functions (or shape functions), u_j unknown constants to be determined and NP denotes the number of nodes or degrees of freedom. In finite element analysis, the convention is to associate each basis function $N_j(\boldsymbol{x})$ and unknown constant u_j with one node in the finite element mesh. Similarly, the approximation of the test function can be represented as

$$w^h(\boldsymbol{x}) = \sum_{i=1}^{\mathrm{NP}} N_i(\boldsymbol{x}) w_i, \tag{3.9}$$

where w_i are arbitrary coefficients because the test function $w^h \in \mathscr{X}_0^h$ is an arbitrary smooth function. By substituting the discretisation (3.8) and (3.9) to the Galerkin formulation (3.7), we obtain

$$\sum_{i=1}^{NP} \sum_{j=1}^{NP} w_i \underbrace{\int_\Omega \nabla N_i \cdot \nabla N_j \, \mathrm{d}\Omega}_{K_{ij}} \; u_j = \sum_{i=1}^{NP} w_i \underbrace{\int_\Omega N_i f \, \mathrm{d}\Omega + \int_{\Gamma_N} N_i t \, \mathrm{d}\Gamma}_{b_i}. \tag{3.10}$$

Given that w_i are arbitrary coefficients, the above equation leads to a system of linear equations

$$\boldsymbol{K}\boldsymbol{u} = \boldsymbol{b}, \tag{3.11}$$

where $\boldsymbol{K}$ is a $\mathrm{NP} \times \mathrm{NP}$ matrix, $\boldsymbol{u}$ is a column vector of size NP containing all the unknown constants u_j and the right hand side vector $\boldsymbol{b}$ is also a column vector of size NP. The entries of matrix $\boldsymbol{K}$ and vector $\boldsymbol{b}$ are calculated as integrals defined in (3.10). The integrals are computed in each element resulting in element matrix and element vector, which are then assembled together to give the global matrix $\boldsymbol{K}$ and global vector $\boldsymbol{b}$.

Remember that in the process of deriving the weak formulation the Dirichlet boundary condition has not be applied yet. This means the linear system (3.11) is under-determined. To obtain a well-defined system, Dirichlet boundary conditions should be enforced using auxiliary techniques. For interpolating basis functions, such as Lagrange polynomials, certain nodal values u_j can be simply determined by the Dirichlet boundary conditions. Remove these "predetermined" degrees of freedom and then solve the reduced system of linear equations for remaining unknowns. For non-interpolating basis functions, such as uniform B-splines, one can use standard techniques used for constrained optimisation problems, including Lagrange multiplier, penalty method, Nitsche method and so on. We will discuss this in more details later using an illustrative example. After the linear equations are solved to determine unknown constants u_j, the finite element solution u^h to the strong formulation (3.1) is obtained according to (3.8).

3.1.3 Isoparametric mapping

In practice, the integrals are computed in each element and summed together to give the matrix $\boldsymbol{K}$ and vector $\boldsymbol{b}$. For example, one entry of matrix $\boldsymbol{K}$ can be computed as

$$K_{ij} = \int_{\Omega} \nabla N_i(\boldsymbol{x}) \cdot \nabla N_j(\boldsymbol{x}) \, \mathrm{d}\Omega = \sum_e \underbrace{\int_{\Omega^e} \nabla N_I(\boldsymbol{x}) \cdot \nabla N_J(\boldsymbol{x}) \, \mathrm{d}\Omega^e}_{K^e_{IJ}} . \tag{3.12}$$

where the local index I, J are introduced. The basis functions used in finite elements usually have local support, meaning that each basis function is zero in most elements. Therefore, when computing the integrals in one element, which leads to an element matrix $\boldsymbol{K}^e$, we only need to consider a few basis functions that are non-zero over the considered element. At the stage of assembling element matrix $\boldsymbol{K}^e$ to global matrix $\boldsymbol{K}$, a mapping to associate the local index with the global index is needed.

However, the integral in physical domain is inefficient and can be rather difficult for complex geometries. Furthermore, the basis functions, such as B-splines, are normally given in a (global) parameter space, making it natural to use parametric form to represent everything, including the test function w and the solution field u. B-splines when used for curve/surface modelling are defined on structured topological space, where a global parametrisation is possible. However, when it comes to unstructured meshes on top of which subdivision surfaces are defined, a global parametrisation is not possible and not needed either. In general, a local parametrisation provides more flexibility and greatly simplifies the integral computing. This is exactly the spirit of *isoparametric mapping*, which is often used in finite element practice.

In the context of isoparametric mapping, a local parameter domain, known as *reference element*, is defined and a local mapping is created for each finite element. For example, for B-spline based FEA, it is convenient to define the reference element as a unit square $\square := [0,1] \times [0,1] \ni \boldsymbol{\eta} = (\eta^1, \eta^2)$ in bivariate case. In each element, we have the following local parametrisation:

$$u^h(\boldsymbol{\eta}) = \sum_J N_J(\boldsymbol{\eta}) u_J , \quad \boldsymbol{\eta} \in [0,1] \times [0,1] , \tag{3.13}$$

$$\boldsymbol{x}(\boldsymbol{\eta}) = \sum_J N_J(\boldsymbol{\eta}) \boldsymbol{x}_J , \quad \boldsymbol{\eta} \in [0,1] \times [0,1] , \tag{3.14}$$

where J is the local index of all the basis functions which are non-zero in the considered element. With the isoparametric mapping, the element integral is transformed to reference element, i.e.,

$$K^e_{IJ} = \int_{\square} \nabla N_I(\boldsymbol{\eta}) \cdot \nabla N_J(\boldsymbol{\eta}) \left| \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\eta}} \right| \mathrm{d}\boldsymbol{\eta} , \tag{3.15}$$

where the Jacobian matrix is computed as

$$\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\eta}} = \begin{bmatrix} \frac{\partial x^1}{\partial \eta^1} & \frac{\partial x^2}{\partial \eta^1} \\ \frac{\partial x^1}{\partial \eta^2} & \frac{\partial x^2}{\partial \eta^2} \end{bmatrix} = \begin{bmatrix} \frac{\partial N_0(\boldsymbol{\eta})}{\partial \eta^1} & \cdots & \frac{\partial N_J(\boldsymbol{\eta})}{\partial \eta^1} & \cdots \\ \frac{\partial N_0(\boldsymbol{\eta})}{\partial \eta^2} & \cdots & \frac{\partial N_J(\boldsymbol{\eta})}{\partial \eta^2} & \cdots \end{bmatrix} \begin{bmatrix} x^1_0 & x^2_0 \\ \vdots & \vdots \\ x^1_J & x^2_J \\ \vdots & \vdots \end{bmatrix} , \tag{3.16}$$

and the derivative is computed as

$$\nabla N_I(\eta) = \begin{bmatrix} \frac{\partial N_I}{\partial x^1} \\ \frac{\partial N_I}{\partial x^2} \end{bmatrix} = \left(\frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\eta}} \right)^{-1} \begin{bmatrix} \frac{\partial N_I}{\partial \eta^1} \\ \frac{\partial N_I}{\partial \eta^2} \end{bmatrix} . \tag{3.17}$$

In finite elements, the integrals are evaluated by numerical integration techniques, such as Gaussian quadrature, i.e.,

$$K^e_{IJ} = \sum_{i=1}^{m} w_i \nabla N_I(\boldsymbol{\eta}_i) \cdot \nabla N_J(\boldsymbol{\eta}_i) \left| \frac{\partial \boldsymbol{x}}{\partial \boldsymbol{\eta}} \right|_{\boldsymbol{\eta}=\boldsymbol{\eta}_i} , \tag{3.18}$$

where $(w_i, \boldsymbol{\eta}_i)$ are the weights and points given by Gaussian quadrature rules for m integration points. Note that normally the basis functions defined over the reference element, e.g., uniform B-spline basis functions and Lagrange polynomials, are universally invariant for all elements. Therefore, we only need to evaluate the basis functions $N_J(\boldsymbol{\eta}_i)$ and their derivatives $\nabla N_J(\boldsymbol{\eta}_i)$ once at each Gauss point. This explains why the element integral evaluation becomes more efficient with the help of isoparametric mapping. When non-uniform or locally modified uniform B-spline basis functions are used in FEA, subdivision can be a powerful technique for efficient basis function evaluation. We will cover the evaluation of B-spline basis in Section 3.2.3.

3.2 When B-splines are used as basis functions

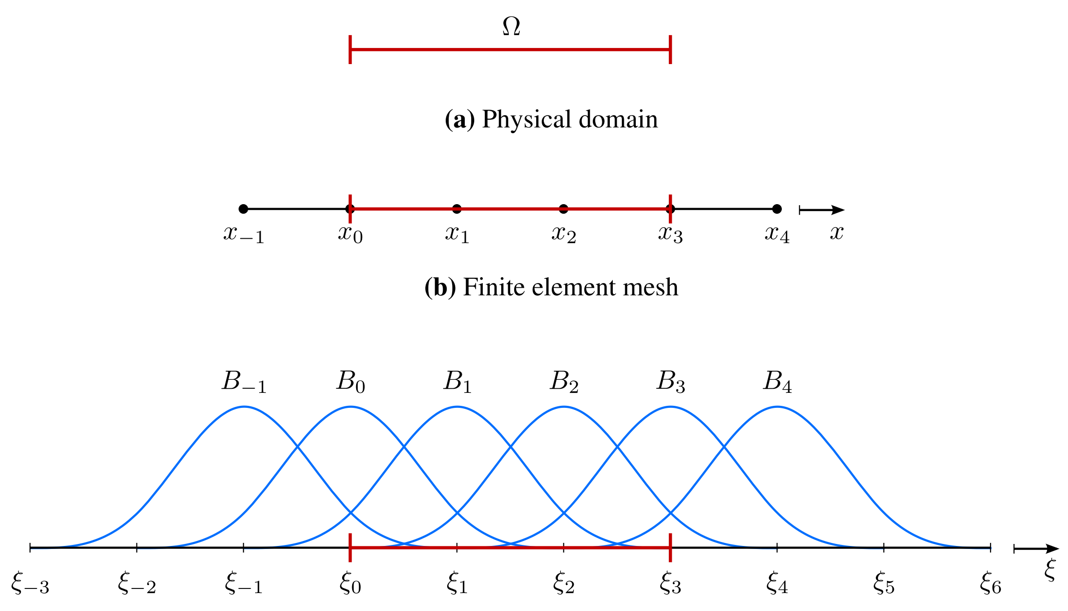
3.2.1 Concept of effective domain and ghost cells

Comparing FEA using Lagrange basis with FEA using B-spline basis, a notable difference is the concept of the effective domain. Recall that each B-spline basis of degree n is defined by $n+2$ knots and supported in $n+1$ knot intervals in the parameter domain. This makes the n knot intervals close to the boundary of the parameter domain ineffective due to the lack of a complete B-spline basis. Therefore, the effective parameter domain consists of knots which are at least n cells away from the boundaries. The physical domain Ω, where the PDE under consideration is defined, only corresponds to the effective parameter domain. Unlike FEA with Lagrange basis, in which case the finite element mesh is boundary-fitted to the physical domain Ω, the finite element mesh for B-spline based FEA is chosen to be larger than the physical domain to have enough nodes to accommodate all B-spline basis functions that contribute to the computation. To illustrate the concept of effective domain, a one-dimensional example is shown in Figure 3.1. It is a topological sketch only, meaning that the knots ξ_j in parameter space and nodes x_j in physical space do not have to be uniformly distributed as they look in Figure 3.1. The finite element mesh has five elements and six nodes. However, the physical domain $\Omega := [0,L]$ corresponds only to the effective computational domain $[x_0,x_3]$, meaning that $x_0 = 0$ and $x_3 = L$. The nodes $x_j \notin \Omega$ are called ghost nodes. The elements containing ghost nodes are called ghost cells. Generally, for B-spline basis of degree n, the finite element mesh contains $\lfloor \frac{n}{2} \rfloor$ layers of ghost cells surrounding the computational domain.

If interpolating basis (e.g., non-uniform B-spline basis defined on open knot vectors) is chosen, the ghost nodes are placed on the boundaries of computational domain, i.e., $x_{-1} = 0$ and $x_4 = L$. However, when non-interpolating basis (e.g. uniform B-spline of degree $n \geq 2$) is used for FEA, extra attention should be paid to ensure the computational domain boundaries are interpolated. In the given example, we require $x_0 = x(\xi_0) = \sum_j B_j(\xi_0)x_j$ and $x_3 = x(\xi_3) = \sum_j B_j(\xi_3)x_j$, so ghost nodes x_{-1} and x_4 in the finite element mesh are set at $x_{-1} = 2x_0 - x_3$ and $x_4 = 2x_3 - x_2$, as discussed in Section 2.4.1.

3.2.2 Methods to enforce boundary conditions

In principle, all the techniques introduced in Section 2.4.1 can be employed to interpolate boundaries of the computational domain as well as to interpolate the Dirichlet boundary conditions given in the strong formulation. In practice, especially for FEA of higher-dimensional problems, adding ghost points is the least favourable approach, because it



(a) Physical domain

(b) Finite element mesh

(c) Parameter domain

Figure 3.1: A topological sketch of computational domain, finite element mesh and the parameter domain for B-spline based FEA. In this plot, cubic B-splines are used as the basis functions. Each cubic B-spline basis $B_j(\xi)$ is defined by five knots $\{\xi_{j-2}, \xi_{j-1}, \xi_j, \xi_{j+1}, \xi_{j+2}\}$ and supported in four knot spans in the parameter domain. Each basis B_j is associated with its central knot ξ_j and a corresponding node x_j in the finite element mesh. The finite element mesh is normally larger than the computational domain Ω where the PDE is defined. In the finite element mesh, the number of nodes are equal to the number of B-spline basis, whereas the computational domain only corresponds to the effective parameter/physical domain (marked in red) where the basis is complete.

involves many constraints of the degrees of freedom u_j. In contrast, open knot vectors and modified subdivision rules lead to interpolating basis functions needing no constraints of the degrees of freedom. In the case of open knot vectors, ghost cells are still included in the finite element mesh but the ghost nodes can be placed on the boundaries of computational domain. When modified subdivision rules are applied at the boundaries, no ghost cells are needed but the resulting B-spline basis is incomplete in the elements close to the boundaries, see the discussion in Section 2.4.1.

In addition to creating interpolating basis, alternatively auxiliary techniques can be used to enforce Dirichlet boundary conditions in FEA where non-interpolating basis, e.g., uniform B-spline, is used. Amongst many, penalty method and Nitsche method as well as their variants are most common.

Penalty method

Taking the aforementioned Poisson's equation as an example, using Penalty method to weakly enforce Dirichlet boundary conditions leads to the following weak formulation [37]: Find $u \in \mathscr{X}$ such that

$$\int_{\Omega} \nabla w \cdot \nabla u \, \mathrm{d}\Omega + \underline{\gamma \int_{\Gamma_D} wu \, \mathrm{d}\Gamma} = \int_{\Omega} wf \, \mathrm{d}\Omega + \int_{\Gamma_N} wt \, \mathrm{d}\Gamma + \underline{\gamma \int_{\Gamma_D} w\bar{u} \, \mathrm{d}\Gamma} \quad \forall w \in \mathscr{X} , \tag{3.19}$$

where the penalty parameter γ needs to be large enough. Compared with the weak formulation in (3.5), two more terms (marked with underlines) are added in (3.19). Notice that the function space $\mathscr{X} = \mathscr{H}^1(\Omega)$ for w, u are now not restricted to any boundary conditions, such as (3.6).

Penalty method is easy to implement but is known to be variationally inconsistent with the strong formulation. A detailed prior error estimation in [37] shows that in order for (3.19) to yield finite element errors converging optimally or sub-optimally, the penalty parameter needs to be sufficiently large $\gamma \geq 1/h^k$, where h is the element size and k depends on the optimal approximation order. However, such a sufficiently large penalty parameter γ can lead to an ill-conditioned stiffness matrix $\boldsymbol{K}$.

Nitsche method

Nitsche method is an alternative technique to enforce Dirichlet boundary conditions weakly. The resulted weak formulation is variationally consistent with the strong form [38]. That is, Find $u \in \mathscr{X}$ such that

$$\begin{aligned} &\int_{\Omega} \nabla w \cdot \nabla u \, \mathrm{d}\Omega - \underline{\int_{\Gamma_D} w \nabla u \cdot \boldsymbol{n} \, \mathrm{d}\Gamma} - \underline{\int_{\Gamma_D} u \nabla w \cdot \boldsymbol{n} \, \mathrm{d}\Gamma} + \gamma \int_{\Gamma_D} wu \, \mathrm{d}\Gamma \\ = &\int_{\Omega} wf \, \mathrm{d}\Omega + \int_{\Gamma_N} wt \, \mathrm{d}\Gamma - \underline{\int_{\Gamma_D} \bar{u} \nabla w \cdot \boldsymbol{n} \, \mathrm{d}\Gamma} + \gamma \int_{\Gamma_D} w\bar{u} \, \mathrm{d}\Gamma \quad \forall w \in \mathscr{X} , \end{aligned} \tag{3.20}$$

which can be proved to be variationally consistent with the strong form (3.1) regardless of the choice of the parameter γ. However, the parameter γ needs to satisfy stability conditions under which the coercivity of the derived bilinear form is guaranteed. For example, as derived in [38], the stability condition requires $\gamma > C$, where C is a mesh-dependent constant and can be estimated as the maximum eigenvalue of a generalised eigenvalue problem. In fact, one can choose a valid parameter $\gamma > C$ in a wide range such that the Nitsche weak formulation (3.20) does not lead to ill-conditioned matrices. Same as penalty method, there are no boundary condition restrictions on the function space $\mathscr{X} = \mathscr{H}^1(\Omega)$ for w, n. Compared with the penalty weak formulation (3.19), three more terms (marked with underlines) are added to the Nitsche weak formulation (3.20). This formulation is referred to as the symmetric

Nitsche formulation. Due to the similarities between the penalty (3.19) and Nitsche (3.20) formulations, some people regard Nitsche method as variationally consistent penalty method, but the parameter γ in Nitsche formulation functions more like a stability parameter rather than penalty parameter. Other variants of Nitsche method exist, such as non-symmetric parameter-free Nitsche formulation [39].

3.2.3 Evaluation of B-spline basis by subdivision

Uniform B-splines

Uniform B-splines are B-splines that are defined on uniformly spaced knots. As a result, a uniform B-spline basis functions are just shifted instances of each other, see Figure 2.6. This means, for all finite elements, the basis functions look the same in the reference element. For example, in the case of uniform cubic B-spline basis, there are four non-zero basis functions over each reference element, see Figure 3.2. The four cubic B-spline pieces which contribute to this considered element are computed as

$$\begin{bmatrix} B_0(\eta) \\ B_1(\eta) \\ B_2(\eta) \\ B_3(\eta) \end{bmatrix} = \begin{bmatrix} \frac{1}{6} & 0 & 0 & 0 \\ \frac{2}{3} & \frac{2}{3} & \frac{1}{3} & \frac{1}{6} \\ \frac{1}{6} & \frac{1}{3} & \frac{2}{3} & \frac{2}{3} \\ 0 & 0 & 0 & \frac{1}{6} \end{bmatrix} \begin{bmatrix} (1-\eta)^3 \\ 3\eta(1-\eta)^2 \\ 3\eta^2(1-\eta) \\ \eta^3 \end{bmatrix} . \tag{3.21}$$

With isoparametric mapping, in each element the geometry and the solution are represented as

$$u^h(\eta) = \sum_{J=0}^{3} B_J(\eta) u_J , \quad \eta \in [0,1] , \tag{3.22}$$

$$x(\eta) = \sum_{J=0}^{3} B_J(\eta) x_J , \quad \eta \in [0,1] , \tag{3.23}$$

where x_J and u_J with the local index $J \in \{0,1,2,3\}$ denote the coordinate and the degree of freedom that are associated to the supporting nodes of the element. For example, in Figure 3.1, the four supporting nodes of element $\Omega^e := [\xi_0, \xi_1]$ are nodes with global index $j \in \{-1,0,1,2\}$, while for element $\Omega^e := [\xi_1, \xi_2]$ they are nodes $j \in \{0,1,2,3\}$. In implementation, the indices of supporting nodes are stored in each element instance. When cubic B-spline basis is used, the supporting nodes are all the nodes in the one-ring neighbourhood consisting of all the elements which share at least one node with the considered element.

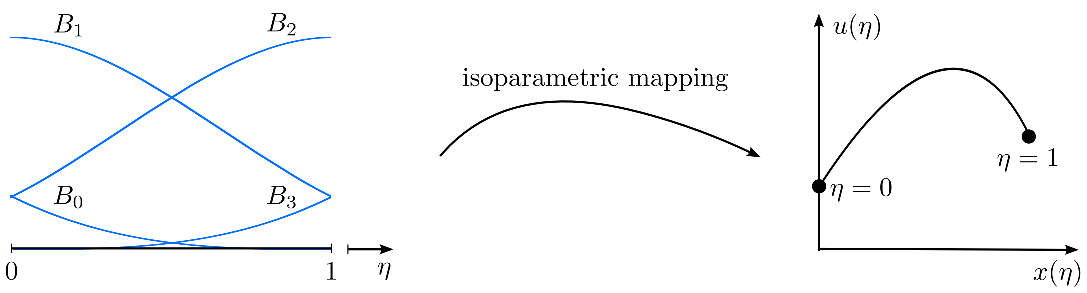


Figure 3.2: Isoparametric mapping from a reference element $\Box := [0,1] \ni \eta$ to the physical space $x(\eta) = \sum_{J=0}^{3} B_J(\eta) x_J$ and the solution field $u(\eta) = \sum_{J=0}^{3} B_J(\eta) u_J$ using uniform cubic B-spline as basis.

Irregular B-splines

In this section, we will demonstrate how subdivision can be used to evaluate irregular B-splines. This evaluation algorithm is just a particular one-dimensional instance of Jos Stam's algorithm for subdivision surface evaluation. Here, we call all except uniform B-splines irregular. The irregularity can be a result of non-uniform knots or locally modified subdivision rules, see Figure 3.3.

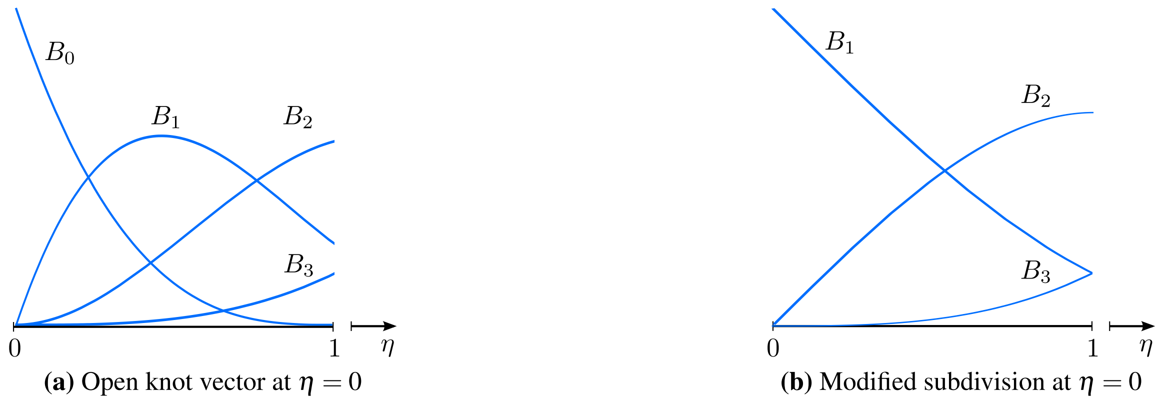


Figure 3.3: Irregular B-splines resulted from a non-uniform knot vector and a modification of the subdivision weights as shown in Figure 2.21. Note that in (b) the basis function $B^0(\eta) = 0$, meaning that the leftmost node in the one-ring neighbourhood of the considered element does not have influence in this element.

Using subdivision as an efficient technique to evaluate irregular B-splines is motivated by two observations. One observation is that a B-spline basis function $B_j(\xi)$ is a B-spline curve given the control coefficients as $p_i = \delta_{ij}$, i.e.,

$$c(\xi) = \sum_i B_i(\xi) p_i = \sum_i B_i(\xi) \delta_{ij} = B_j(\xi). \tag{3.24}$$

Recall that if we keep refining the control polygon by subdivision algorithm in the limit the control polygon will converge to the represented B-spline curve. Subdivision is a linear operator mapping the control points to a finer level,

$$\boldsymbol{p}^1 = \boldsymbol{S}\boldsymbol{p}\,, \tag{3.25}$$

where $\boldsymbol{p}$ is a vector of initial control points and $\boldsymbol{p}_1$ denotes control points after one subdivision refinement, referred to as control points at subdivision level $\ell = 1$. The convention is to denote initial control points $\boldsymbol{p} := \boldsymbol{p}^0$ meaning subdivision level $\ell = 0$. More generally, we have

$$\boldsymbol{p}^{\ell+1} = \boldsymbol{S}\boldsymbol{p}^{\ell} = \boldsymbol{S}^{\ell+1}\boldsymbol{p}^0\,, \tag{3.26}$$

and in the limit

$$\boldsymbol{p}^{\infty} = \boldsymbol{S}^{\infty}\boldsymbol{p}^0\,. \tag{3.27}$$

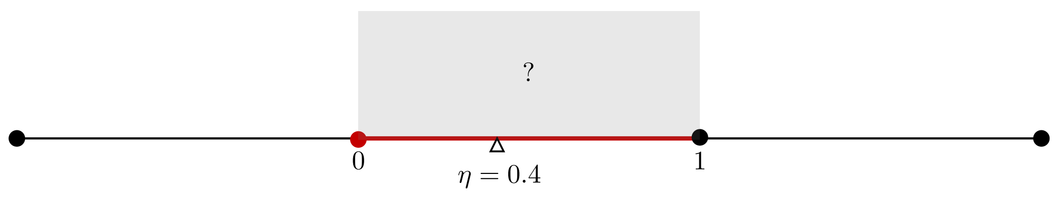
To represent a B-spline basis function $B_j(\xi)$, we have determined the initial control coefficients $\boldsymbol{p}^0$. In this case, the limit curve $\boldsymbol{p}^{\infty}$, which is the B-spline basis function itself, totally depends on the subdivision matrix $\boldsymbol{S}$. This means the subdivision weights inherit all information that is needed to define a B-spline basis function.

Given that the represented curve stays unchanged during subdivision, we have

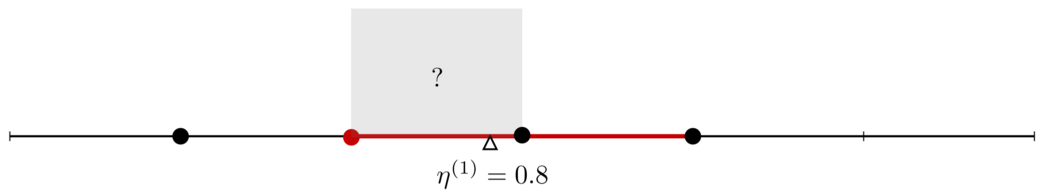
$$B_j(\xi) = c(\xi) = \boldsymbol{B}^{(0)^{\mathsf{T}}}(\xi)\boldsymbol{p}^0 = \boldsymbol{B}^{(\ell)^{\mathsf{T}}}(\xi)\boldsymbol{p}^{\ell} \tag{3.28}$$

implying that the B-spline basis function $B_j(\xi)$ can be evaluated at any subdivision level ℓ. We can guarantee that the basis functions $B^l_j(\xi)$ defined on a finer level (on a refined knot vector) will be easier to evaluate than initial B-splines $B^0_j(\xi)$. This is because uniform subdivision, i.e., bisecting non-zero knot intervals, will create a region with more and more uniformly distributed knots. In other words, if we apply subdivision a sufficient number of times, the finer level basis functions $B^l_j(\xi)$ become uniform B-splines and are trivial to evaluate.

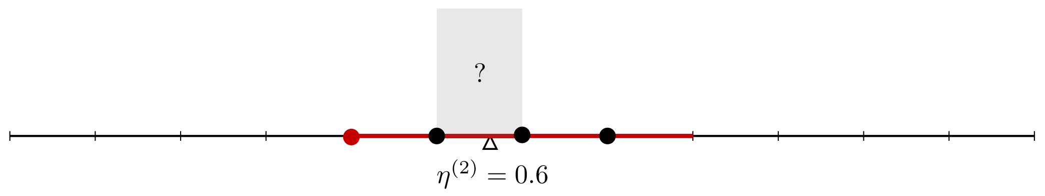
As an illustrative example of the process of evaluation by subdivision, let us consider the value of the basis $B_1(\eta)$ shown in Figure 3.3b at $\eta = 0.4$. Figure 3.4 provides a graphical illustration of the subdivision process, from which it is clear that three cycles of subdivision refinements are needed in order to embed the target parameter point $\eta = 0.4$ in an element influenced by uniform B-splines only. Table 3.1 lists the computation of the control point values at each subdivision level until level $l = 3$ where only uniform B-spline basis functions contribute to the target element. The subdivision weights used can be found in Figure 2.21.



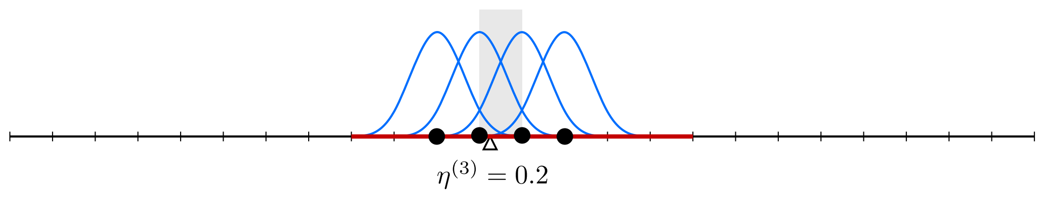
(a) Initial knot vector



(b) Once subdivided knot space



(c) Twice subdivided knot space



(d) Three times subdivided knot space

Figure 3.4: Graphical illustration of the process of evaluation in a reference element influenced by the modified subdivision rules applied at the prescribed sharp vertex (marked in red). The red line marks a knot span in the initial knot space. At every subdivision step, a new knot is inserted to the middle of a non-zero knot span, creating two equidistant knot spans. This means the B-spline basis functions defined by knots located within the same initial knot span will be uniform B-splines, e.g., the four B-splines plotted on level three. The grey window at each level denotes the element which contains the target parameter where the B-spline basis is to be evaluated. Note that at any subdivision level we parametrise each knot span to a unit length, i.e.,$\eta^{(\ell)} \in [0, 1]$.

Table 3.1: Numerical illustration of evaluation of irregular B-spline basis by subdivision, see Figure 3.4 for a graphical illustration.

Level	Parameter	Uniform B-splines?	Control points update
0	$\eta^{(0)} = 0.4$	No	$\boldsymbol{p}^0 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$
1	$\eta^{(1)} = 0.8$	No	$\boldsymbol{p}^1 = \frac{1}{8}\begin{bmatrix} 4 \\ 8 \\ 4 \\ 1 \end{bmatrix} = \frac{1}{8}\begin{bmatrix} 4 & 4 & 0 & 0 \\ 0 & 8 & 0 & 0 \\ 0 & 4 & 4 & 0 \\ 0 & 1 & 6 & 1 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}$
2	$\eta^{(2)} = 0.6$	No	$\boldsymbol{p}^2 = \frac{1}{64}\begin{bmatrix} 64 \\ 48 \\ 33 \\ 20 \end{bmatrix} = \frac{1}{8^2}\begin{bmatrix} 0 & 8 & 0 & 0 \\ 0 & 4 & 4 & 0 \\ 0 & 1 & 6 & 1 \\ 0 & 0 & 4 & 4 \end{bmatrix}\begin{bmatrix} 4 \\ 8 \\ 4 \\ 1 \end{bmatrix}$
3	$\eta^{(3)} = 0.2$	Yes	$\boldsymbol{p}^3 = \frac{1}{512}\begin{bmatrix} 385 \\ 324 \\ 266 \\ 212 \end{bmatrix} = \frac{1}{8^3}\begin{bmatrix} 1 & 6 & 1 & 0 \\ 0 & 4 & 4 & 0 \\ 0 & 1 & 6 & 1 \\ 0 & 0 & 4 & 4 \end{bmatrix}\begin{bmatrix} 64 \\ 48 \\ 33 \\ 20 \end{bmatrix}$

The initial control coefficients prescribed for $B_1(\eta)$ are

$$\boldsymbol{p}^0 = [0,1,0,0]^\mathsf{T}\,, \tag{3.29}$$

resulting in the following control coefficients after three times of subdivision:

$$\boldsymbol{p}^3 = \frac{1}{512}[385,324,266,212]^\mathsf{T}\,. \tag{3.30}$$

In addition, given (3.21) it is straightforward to evaluate the B-spline basis on level 3 for $\eta^{(3)} \in [0,1]$

$$\boldsymbol{B}^{(3)}(\eta) = \begin{bmatrix} B_0^{(3)}(\eta) \\ B_1^{(3)}(\eta) \\ B_2^{(3)}(\eta) \\ B_3^{(3)}(\eta) \end{bmatrix} = \frac{1}{6}\begin{bmatrix} 1 & 0 & 0 & 0 \\ 4 & 4 & 2 & 1 \\ 1 & 2 & 4 & 4 \\ 0 & 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} (1-\eta)^3 \\ 3\eta(1-\eta)^2 \\ 3\eta^2(1-\eta) \\ \eta^3 \end{bmatrix}, \tag{3.31}$$

where the superscript level number in $\eta^{(3)}$ is omitted for brevity. It should be clear that the input parameter η for B-spline $\boldsymbol{B}^{(\ell)}(\eta)$ is always $\eta^{(\ell)}$, which is the parameter value mapped recursively from the initial parameter value $\eta^{(0)}$ according to

$$\eta^{(\ell+1)} = \begin{cases} 2\eta^{(\ell)} & \text{if} \quad \eta^{(\ell)} \in [0,0.5] \\ 2\eta^{(\ell)} - 1 & \text{if} \quad \eta^{(\ell)} \in (0.5,1] \end{cases} \quad \text{for} \quad \ell = 0,1,\cdots\,. \tag{3.32}$$

This effectively transforms each element on level ℓ, see the grey windows in Figure 3.4, to a reference parameter domain $\square := [0,1] \ni \eta^{(\ell)}$. Table 3.1 also lists the parameter transformation during the subdivision process. Now we are ready to compute $B_1(\eta)$ according to (3.28), i.e.,

$$B_1(\eta) = \begin{bmatrix} B_0 & B_1 & B_2 & B_3 \end{bmatrix}\begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} B_0^{(3)} & B_1^{(3)} & B_2^{(3)} & B_3^{(3)} \end{bmatrix}\begin{bmatrix} 385 \\ 324 \\ 266 \\ 212 \end{bmatrix}/512\,, \tag{3.33}$$

Plug in $\eta^{(3)} = 0.2$ into (3.31), we obtain

$$\begin{bmatrix} B_0^{(3)} & B_1^{(3)} & B_2^{(3)} & B_3^{(3)} \end{bmatrix} = \frac{1}{750}\begin{bmatrix} 64 & 473 & 212 & 1 \end{bmatrix} \quad \text{at} \quad \eta^{(3)} = 0.2\,. \tag{3.34}$$

Finally substituting (3.34) into (3.33) gives the value of the irregular B-spline basis $B_1(\eta)$ resulted from modified subdivision rules given in Figure 2.21,

$$B_1(\eta) = \frac{229}{375} \quad \text{at} \quad \eta = 0.4\,. \tag{3.35}$$

In fact, in Section 2.4.1 when we discuss different techniques to create the boundary interpolation for subdivision curves, we have mentioned that applying modified subdivision rules given in Figure 2.21 will result in basis functions equivalent to (2.45), repeated here,

$$\begin{bmatrix} B_1(\eta) \\ B_2(\eta) \\ B_3(\eta) \end{bmatrix} = \frac{1}{6} \begin{bmatrix} 6 & -6 & 0 & 1 \\ 0 & 6 & 0 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ \eta \\ \eta^2 \\ \eta^3 \end{bmatrix}, \tag{3.36}$$

which also gives the same value as (3.35) without surprise. In fact, (3.33) is valid not only for $\eta = 0.4$ used in this example but also for any $\eta \in [0.375, 0.5]$. This is because $\eta \in [0.375, 0.5]$ will end up in the same target element at subdivision level $\ell = 3$, thus requiring the same evaluation process. In practice, we take the initial coefficients $\boldsymbol{p}^0$ as an identity matrix of size $n+1$ to evaluate the $n+1$ non-zero B-spline basis in the considered element, which can be easily verified analogous to (3.28), i.e.,

$$\boldsymbol{B}^{(0)^{\mathsf{T}}}(\eta) = \boldsymbol{c}(\eta) = \boldsymbol{B}^{(0)^{\mathsf{T}}}(\eta)\boldsymbol{I} = \boldsymbol{B}^{(\ell)^{\mathsf{T}}}(\eta^{(\ell)})\boldsymbol{p}^{\ell}\,. \tag{3.37}$$

The algorithm of evaluation by subdivision demonstrated here can be applied to evaluate non-uniform cubic B-spline basis functions as well, in which case the subdivision weights for non-uniform knot vectors should be used, see Figure 2.17 and Figure 2.18. However, it is not too difficult to derive the closed form functions for non-uniform B-spline basis or modified B-spline basis, which can be tabulated for evaluation use. The real power of this evaluation algorithm is that it provides the very first approach to evaluate subdivision surfaces in irregular region around extraordinary vertices [40], where a closed form is not trivial to derive.

3.3 Examples

Now we have all techniques in place to perform FEA using B-splines as basis functions. As an illustrative example, we consider the following one-dimensional Poisson's equation:

$$-\frac{\mathrm{d}^2 u}{\mathrm{d}x^2} = f\,, \quad x \in [0,1] \tag{3.38a}$$

$$u(0) = \overline{u}_0\,, \quad u(1) = \overline{u}_1\,. \tag{3.38b}$$

Only Dirichlet boundary conditions are included because Neumann boundary conditions are natural boundary conditions, which are easier to enforce compared with Dirichlet boundary conditions. In order to validate the finite element solution u^h, we compare it with the analytical solution u. To this end, we prescribe the analytical solution to be $u = u(x)$ by computing the boundary conditions from $\overline{u}_0 = u(0)$, $\overline{u}_1 = u(1)$ and the source term from $f = -\,\mathrm{d}^2 u/\,\mathrm{d}x^2$.

The finite element discretisation error is measured in both L_2 norm and energy norm, computed as

$$\begin{aligned} ||u^h - u||_{L_2} &= \left(\int_\Omega (u^h - u)^2 \,\mathrm{d}\Omega\right)^{\frac{1}{2}}\,, \\ ||u^h - u||_E &= \left(\int_\Omega (\nabla u^h - \nabla u)^2 \,\mathrm{d}\Omega\right)^{\frac{1}{2}}\,, \end{aligned} \tag{3.39}$$

which are evaluated by Gauss integration, same as integrals in the weak formulation. More often we look at the normalised errors, i.e.,

$$\frac{||u^h - u||_{L_2}}{||u||_{L_2}} \quad \text{and} \quad \frac{||u^h - u||_E}{||u||_E}\,. \tag{3.40}$$

For second-order PDEs, such as Poisson's equation, if the solution is approximated using cubic B-spline basis functions, prior error estimations indicate that the L_2 norm error $||u^h - u||_{L_2}$ converges optimally at order of 4 and the energy norm error $||u^h - u||_E$ at order of 3.

3.3.1 Penalty method vs Nitsche method

As the first example, we prescribe the analytical solution to be

$$u = \sin\left(\frac{\pi}{2}x\right)\,,$$

which means the boundary conditions are set to be $\overline{u}_0 = 0$, $\overline{u}_1 = 1$ and the source term

$$f = -\mathrm{d}^2 u/\mathrm{d}x^2 = \frac{\pi^2}{4}\sin\left(\frac{\pi}{2}x\right) .$$

We have mentioned two methods to enforce Dirichlet boundary conditions in a weak form in Section 3.2.2. In this example, we compare the penalty method and Nitsche method. Uniform cubic B-splines are chosen as the basis functions. The element integrals are evaluated by 3 Gauss quadrature points, considering that integration of a polynomial function of degree p needs at least $\lceil (p+1)/2 \rceil$ Gauss points. The computational domain is discretised with elements of a uniform size $h = 1/N$, where the number of elements $N \in \{4, 8, 16, 32, 64, 128\}$. Remember that for cubic B-splines the finite element mesh contains one layer of ghost cells, meaning that the meshes used in our computation contain $N+2$ elements uniformly distributed in the extended domain $[0-h, 1+h]$.

Figure 3.5 illustrates that the finite element solution is very sensitive to the choice of the parameter γ when penalty method is used. As discussed in Section 3.2.2, the parameter γ in the penalty weak formulation (3.19) needs to be sufficiently large to make sure the finite element solution satisfies the Dirichlet boundary conditions. However, as shown in Figure 3.5, when $\gamma \geq 10^{10}$ the linear system becomes ill-conditioned and overall $\gamma = 10^9$ is the best choice in this particular example.

However, the solution is less sensitive to the choice of the parameter γ when the Nitsche formulation is applied, see Figure 3.6. As discussed in Section 3.2.2, Nitsche formulation is variationally consistent with the strong form regardless of the choice of γ. However, as a stability parameter, there is a minimum threshold for γ to guarantee the coercivity of the linear equation system $\boldsymbol{Ku} = \boldsymbol{b}$, see [38]. A local or global eigenvalue problem can be solved to provide a good estimate for the minimum value of a valid stability parameter γ. Numerical experiments show that in this example $\gamma \geq 10^3$ achieves stability for all element sizes $h = 1/N$ with $N \in \{4, 8, 16, 32, 64, 128\}$.

The convergence plots for penalty method and Nitsche method are put together in Figure 3.7 at three different parameter values $\gamma = 10^3, 10^6, 10^9$. Clearly, at $\gamma = 10^3, 10^6$ Nitsche method leads to optimally converged solutions whereas the solution generated by penalty method does not converge. However, at $\gamma = 10^9$ both Nitsche and penalty methods lead to converged solutions but they deviate from the optimal convergence as the element size h decreases. This one-dimensional example demonstrates in general Nitsche method is more robust than penalty method for Dirichlet boundary conditions enforcement. But we would like to point out that one might find penalty method works fine in some specific situations.

Due to its simplicity, penalty method is still commonly used in many implementations to enforce different types of constraints.

3.3.2 Different end conditions

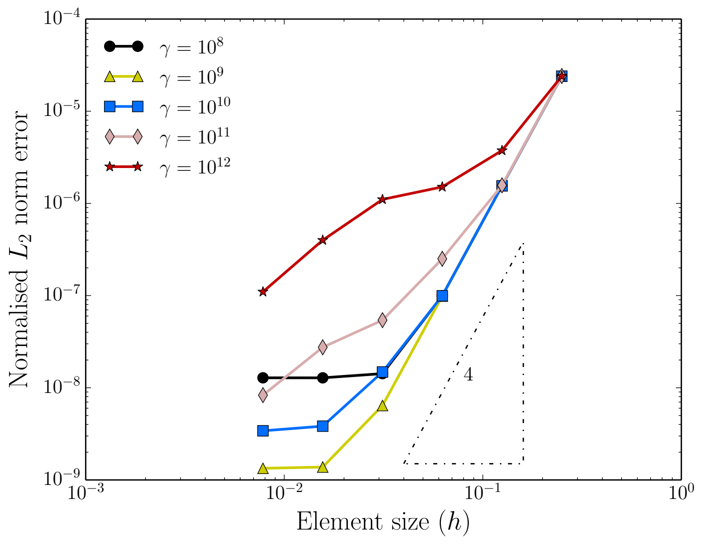
In this section, we compare three different types of cubic B-spline basis generated by different end conditions, including B-spline defined by open knot vectors (Bézier end conditions), B-spline generated by modified subdivision rules at boundaries (subdivision end conditions), and uniform B-splines. Just as in the previous example, the element integrals are evaluated by 3 Gauss quadrature points and the computational domain is discretised with elements of a uniform size $h = 1/N$, where the number of elements $N \in \{4, 8, 16, 32, 64, 128\}$. In the case of Bézier end conditions and uniform B-splines, the finite element mesh contains one layer of ghost cells. For uniform B-splines, the mesh consists of $N+3$ nodes uniformly distributed at $x_j \in \{-1, 0, 1, 2, \cdots, N-2, N-1, N, N+1\}h$. For Bézier end conditions, the finite element mesh is uniform in the middle but non-uniform close to boundaries, i.e., $x_j \in \{0, \frac{1}{3}, 1, 2, \cdots, N-2, N-1, N-\frac{1}{3}, N\}h$, which gives a linear parametrisation of $x(\eta)$ with Jacobian being constant. In the case of subdivision boundary conditions, no ghost cells are needed and the finite element mesh simply consists of $N+1$ uniformly distributed nodes $x_j \in \{0, 1, 2, \cdots, N-2, N-1, N\}h$. In fact, regardless of the basis functions, in one-dimensional problems it is straightforward to apply Dirichlet boundary conditions, which are simply described by linear constraints. For instance, consider the finite element solution represented by

$$u^h(\eta) = \sum_{j=0}^{N+2} B_j(\eta) u_j \tag{3.41}$$

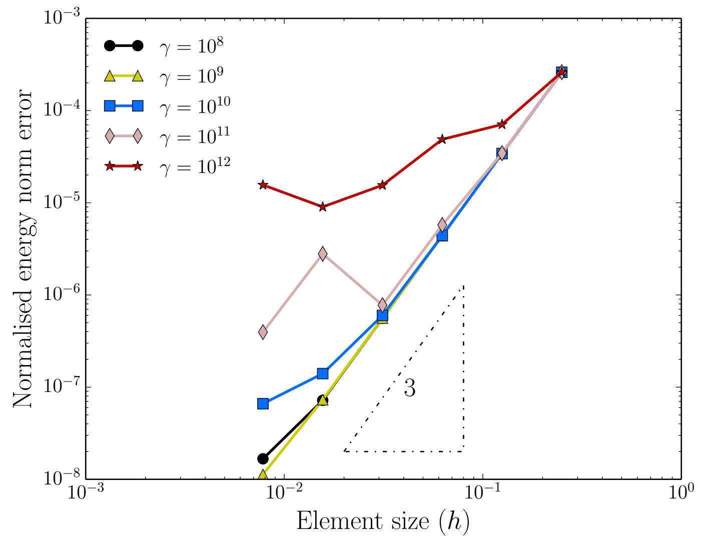
with unknowns u_j to be determined. The left boundary $x = 0$ refers to $\eta = 0$ in the first element, over which only the first four B-splines are non-zero. Therefore, the Dirichlet boundary condition $u = \overline{u}$ at $x = 0$ requires

$$u^h(x=0) = \sum_{j=0}^{3} B_j(\eta = 0) u_j = \overline{u}. \tag{3.42}$$

The linear constraints can be applied by standard techniques, such as Lagrange multipliers or rows/columns elimination. In this example, the Dirichlet boundary conditions are enforced in a strong form with the help of Lagrange multipliers for all different types of B-spline basis functions. For one-dimensional problems, Dirichlet boundary conditions can be easily applied in a strong form. In two- or three-dimensional problems, however, it is not always as straightforward as one-dimensional examples. In B-spline based FEA, Dirichlet boundary

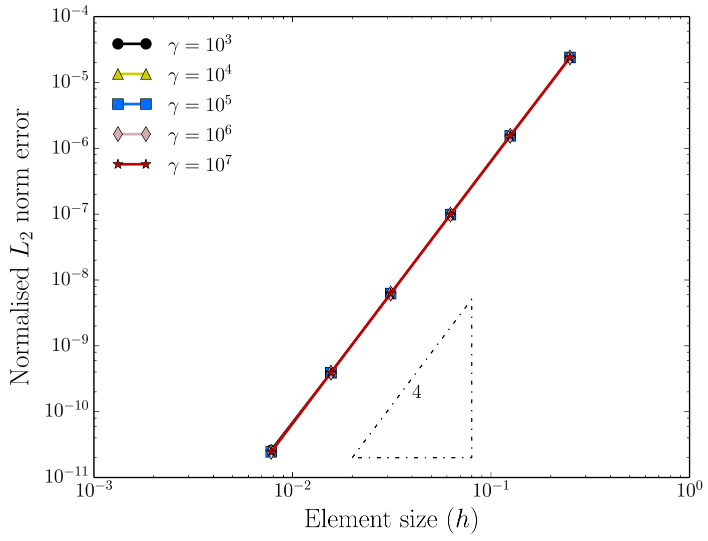


(a) L_2 norm error

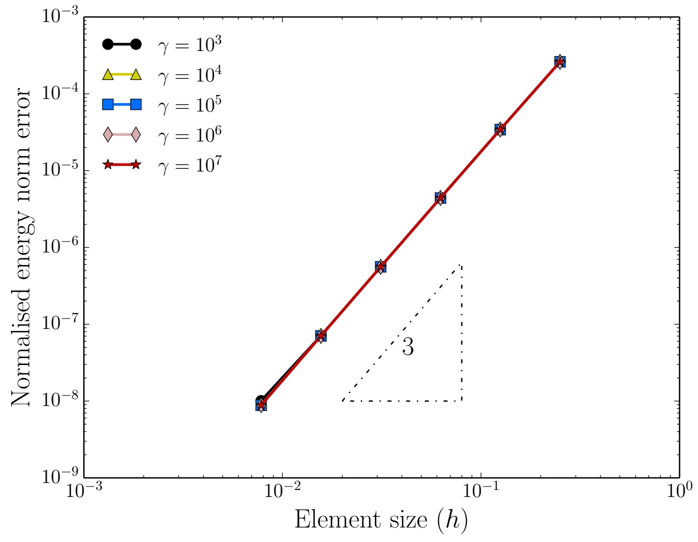


(b) Energy norm error

Figure 3.5: Uniform cubic B-splines using penalty method. Influence of penalty parameters on the convergence of L_2 and energy norm errors. As expected, when penalty parameter is too small or too large, the finite element solution does not converge. This is because the weak formulation is only variationally consistent for sufficiently large penalty parameter γ which unfortunately results in ill-conditioned matrix. The numerical experiments show that $\gamma = 10^9$ is the best choice in this particular example when taking the finite element errors in both L_2 and energy norms into consideration.

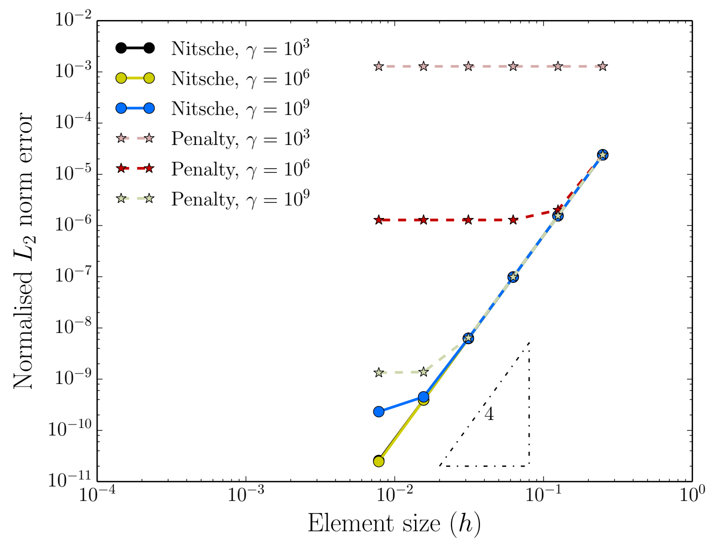


(a) L_2 norm error

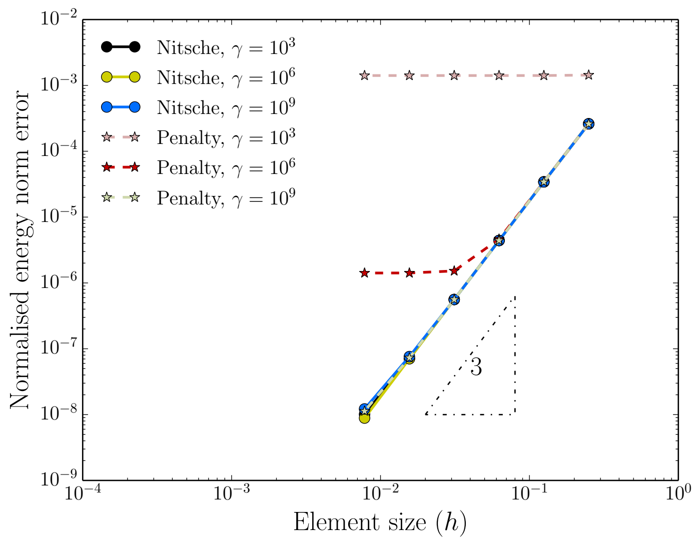


(b) Energy norm error

Figure 3.6: Uniform cubic B-splines using Nitsche method. The convergence of L_2 and energy norm errors is not sensitive to the value of stability parameter γ. Our numerical experiments indicate that in this example Nitsche formulation with $\gamma \geq 1000$ leads to optimally converged finite element solution. But when $\gamma \geq 10^9$ the stiffness matrix $\boldsymbol{K}$ becomes ill-conditioned and the finite element solution will deviate from the optimal convergence as the element size decreasing, see Figure 3.7.



(a) L_2 norm error



(b) Energy norm error

Figure 3.7: Uniform cubic B-splines using penalty method or Nitsche method. Nitsche method yields optimal convergence rates for both L_2 and energy norm errors when the parameter $\gamma = 10^3, 10^6$. However, the solution obtained by using penalty method deviates from the optimal convergence regardless of the parameter γ, see Figure 3.5 as well.

conditions are often enforced in a weak form by Nitsche method. In this example, we compare the results of weakly and strongly enforced Dirichlet boundary conditions to validate Nitsche method.

Example 1

Same as previous example, we prescribe the analytical solution to be

$$u = \sin\left(\frac{\pi}{2}x\right) .$$

Figure 3.8 plots both L_2 norm and energy norm errors of the finite element solutions obtained by using different B-splines together with either Lagrange multipliers or Nitsche method. From Figure 3.8, we see that weak enforcement of Dirichlet boundary conditions by Nitsche method achieves the same accuracy as the strong enforcement by Lagrange multipliers. In both cases, optimal convergence rates are achieved for uniform B-splines and non-uniform B-splines with Bézier end conditions. However, as expected, for subdivision end conditions the convergence rates are not optimal given the prescribed solution $u(x) = \sin(\pi x/2)$. This is because the prescribed solution has non-zero second order derivative on the boundary $x = 1$ whereas the incomplete basis in the boundary elements resulted from subdivision end conditions lacks quadratic term and leads to zero second order derivative, see the detailed discussion in Section 2.4.1.

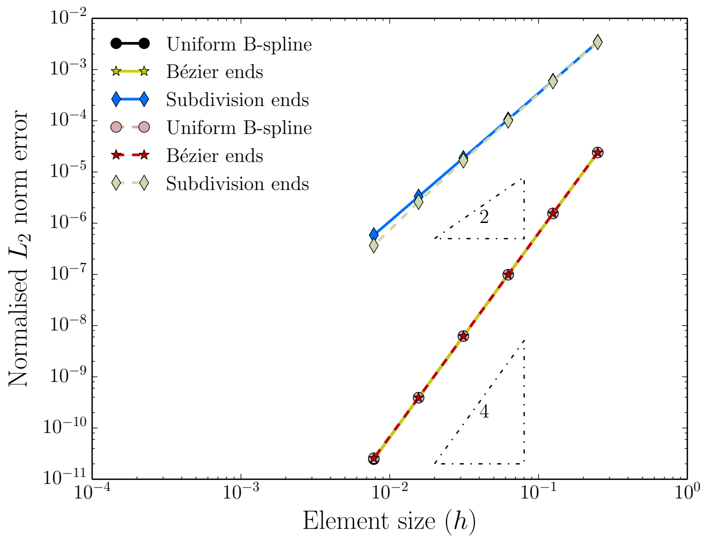
Example 2

As the second example, we prescribe the analytical solution to be

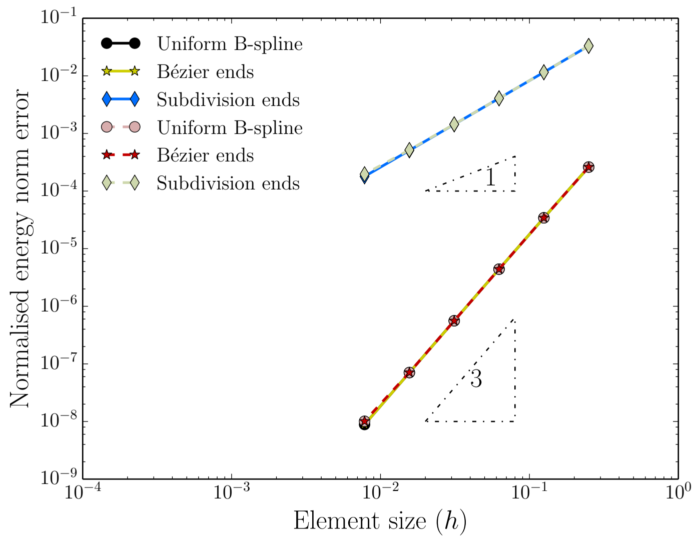
$$u = \sin(\pi x) ,$$

which has zero values and zero second order derivatives on both ends $x = 0$ and $x = 1$.

Again, we compare the performance of different bases. Figure 3.9 plots both L_2 norm and energy norm errors of the finite element solutions obtained by using different B-splines together with either Lagrange multipliers or Nitsche method. This time optimal convergence rates are achieved for the basis generated by subdivision end conditions as well. This is because the prescribed solution has zero second order derivative at both boundaries $x = 0$ and $x = 1$, in which case the missing quadratic term in the basis does not influence the convergence rates. This example shows that subdivision end conditions are not always bad.

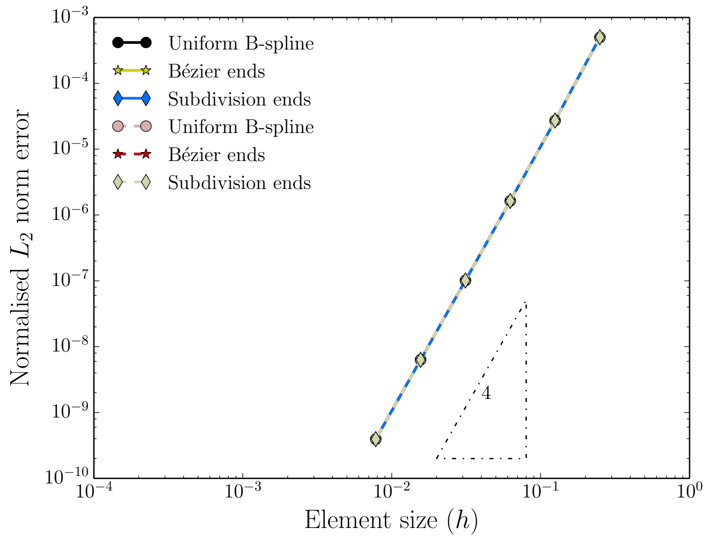


(a) L_2 norm error

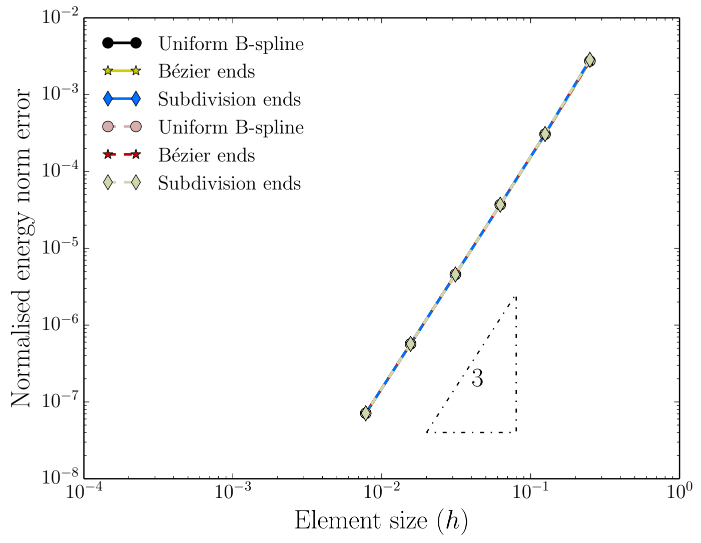


(b) Energy norm error

Figure 3.8: Convergence of finite element solutions obtained by using different B-spline basis together with Lagrange multipliers (solid) or Nitsche method (dashed). Weak enforcement of Dirichlet boundary conditions by Nitsche method achieves the same accuracy as strong enforcement by Lagrange multipliers. The convergence rates for uniform B-splines and non-uniform B-splines with Bézier end conditions are optimal. However, as expected, for subdivision end conditions the convergence rates are not optimal given the prescribed solution $u(x) = \sin(\pi x/2)$. This is because the prescribed solution has non-zero second order derivative on the boundary $x = 1$ whereas the incomplete basis in the boundary elements resulted from subdivision end conditions lacks quadratic term and leads to zero second derivatives, see the detailed discussion in Section 2.4.1.



(a) L_2 norm error



(b) Energy norm error

Figure 3.9: Convergence of finite element solutions obtained by using different B-spline basis together with Lagrange multipliers (solid) or Nitsche method (dashed). Again, weak enforcement of Dirichlet boundary conditions by Nitsche method achieves the same accuracy as strong enforcement by Lagrange multipliers. This time, however, optimal convergence rates are also achieved for subdivision end conditions given the prescribed solution $u(x) = \sin(\pi x)$. This is because the prescribed solution has zero second order derivative on both boundaries $x = 0$ and $x = 1$, in which case the missing quadratic term in the boundary elements does no harm to the convergence rates.

Chapter 4

Subdivision surfaces with optimised refinement weights

Subdivision curves/surfaces are methods to represent smooth curves/surfaces by coarse control polygons/meshes. The represented curves/surfaces are calculated by recursive subdivision refinements of the initial control polygons/meshes. In previous chapters, taking univariate subdivision as an example we have demonstrated how subdivision construction can be used to evaluate the B-spline basis functions in finite element analysis. In this chapter, bivariate subdivision schemes, i.e., subdivision surfaces, will be used for finite element analysis. For structured quadrilateral meshes, i.e, each vertex is shared by four faces, tensor-product of univariate B-splines gives B-spline surfaces. Quadrilateral subdivision schemes generalise tensor-product B-spline surfaces to unstructured meshes containing *extraordinary vertices*, i.e., vertices shared by other than four faces. Near extraordinary vertices, the optimal values of subdivision weights are application-dependent. This chapter presents our work on how to optimise subdivision weights for isogeometric analysis applications [11].

Subdivision surfaces provide an elegant isogeometric analysis framework for geometric design and analysis of partial differential equations defined on surfaces. They are already a standard in high-end computer animation and graphics and are becoming available in a number of geometric modelling systems for engineering design, including Catia, PTC Creo and Autodesk Fusion 360. The subdivision refinement rules are usually adapted from knot insertion rules for B-splines. The quadrilateral Catmull-Clark scheme considered in this work is equivalent to cubic B-spline surfaces away from extraordinary, or irregular, vertices with other than four adjacent elements. Around extraordinary vertices the surface consists of a nested sequence of smooth B-spline patches which join C^1 continuously at the point itself. As known from geometric design literature, the subdivision weights can be optimised so that the surface quality is improved by minimising short-wavelength surface oscillations around extraordinary vertices. We use the related techniques to determine weights that minimise finite element discretisation errors as measured in the thin-shell energy norm. The optimisation problem is formulated over a characteristic domain and the errors in approximating cup- and saddle-like quadratic shapes obtained from eigenanalysis of the subdivision matrix are minimised. In finite element analysis the optimised subdivision weights for either cup- or saddle-like shapes are chosen depending on the shape of the solution field around an extraordinary vertex. As our computations confirm, the optimised subdivision weights yield a reduction of 50% and more in discretisation errors in the energy and L_2 norms. Although, as to be expected, the convergence rates are the same as for the classical Catmull-Clark weights, the convergence constants are improved.

4.1 Introduction

As a generalisation of splines, subdivision surfaces can provide watertight representations for geometries with arbitrary topology. Before the advent of isogeometric analysis, it had already been realised that subdivision surfaces provide also ideal basis functions for finite element analysis, in particular, of thin-shells [6, 16–18], see also more recent work [41, 42].

Subdivision schemes for generating smooth surfaces were first described in the late 1970s as an extension of low degree B-splines to control meshes with non-tensor-product connectivity [14, 15]. In subdivision a geometry is described with a control mesh and a limiting process of repeated refinement. For parts of the mesh containing only regular vertices, with each adjacent to four quadrilateral faces, the refinement rules are adapted from knot insertion rules for B-splines. For the remaining parts with extraordinary vertices the refinement rules are chosen such that they yield in the limit a smooth surface. Subdivision refinement is a linear mapping of coordinates of the coarse control mesh to the coordinates of the refined mesh with a subdivision matrix. Hence, the local limit surface properties can be inferred from the eigenstructure of the subdivision matrix after a discrete Fourier transform [15, 43]. The C^1 continuity of the surface and its curvature behaviour at the extraordinary vertex depend on eigenvalues and the ordering, i.e. Fourier indices, of the corresponding eigenvectors. In turn, both depend on the coefficients of the subdivision matrix that encodes the specific refinement rules applied.

As known, around extraordinary vertices short-wavelength surface oscillations, i.e. ripples, may occur irrespective of C^1 continuity and boundedness of curvature [44, 45]. There have been many attempts to improve the fairness of subdivision surfaces, that is, to minimise curvature variations, by carefully tuning the refinement rules, earlier works include [46, 47]. More recently, in [48] the refinement rules for Catmull-Clark and other quadrilateral schemes have been optimised such that the variation of the Gaussian curvature is minimised while ensuring bounded curvatures. Different from the direct search method used in [48], the refinement rules can also be obtained from a nonlinear constrained optimisation problem. In [49] such a procedure is applied to triangular Loop and $\sqrt{3}$-subdivision schemes with a multi-objective cost function comprised of terms penalising divergence of curvatures and aiming local quadratic precision. In [50] a fairness increasing cost function containing the third derivatives of the surface in combination with C^1 continuity and bounded curvature constraints is optimised.

In the present work, we optimise the subdivision refinement rules so that their approximation properties are improved when used in finite element analysis of thin-shells. Thin-shells are prevalent in many engineering applications, most prominently in aerospace, automotive and structural engineering, and are equivalent to thin-plates when their unstressed geometry is planar [51]. The thin-shell energy functional and weak form depend on the second order derivatives of the stressed surface. Consequently, it is crucial to reduce any short-wavelength oscillations in the subdivision surface. As the included examples demonstrate, meshes with extraordinary vertices usually lead to lower convergence rates than meshes with tensor-product connectivity. For obtaining the improved isogeometric analysis adapted refinement rules we postulate a constrained optimisation problem with a cost function measuring the errors in approximating cup- and saddle-like quadratic shapes. Three of the weights in the Catmull-Clark subdivision scheme around an extraordinary vertex are chosen as degrees of freedom for optimisation. As constraints the C^1 continuity of the surface is strictly enforced and bounded curvatures are enforced as long as non-negative real weights are feasible. The eigenstructure of the subdivision matrix is extensively used in formulating the optimisation problem as usual in previous related work [52, Chapter 4,5] and [53, Chapter 15]. We compute the eigenvalues and eigenvectors numerically after applying a discrete Fourier transform that exploits the local circular symmetry around the extraordinary vertex. The local parametrisation of the subdivision surface required for evaluating the finite element integrals and the cost function is obtained with the algorithm proposed in [40]. Two sets of optimised weights for cup- and saddle-like shapes are obtained. The weights for finite element analysis are chosen depending on the dominant shape of the solution field around an extraordinary vertex.

For completeness, we note that subdivision is not the only approach for creating smooth surfaces on arbitrary connectivity control meshes. Over the years numerous C^k and G^k smooth constructions with $k \geq 1$ have been proposed, too many to name here. The search for sufficiently flexible smooth surface representations, especially with $C^{k\geq 2}$ and $G^{k\geq 2}$, is still open. It is worth mentioning that none of the existing constructions is widely used in commercial CAD systems. This may well be because their implementation is too complicated. The application of basis functions resulting from smooth constructions for isogeometric analysis is currently a very active area of research. For instance, the utility of G^k constructions with NURBS has recently been explored in [7, 9, 54]. Alternatively, C^k constructions relying on manifold-based surface constructions [19, 21, 24] and constructions relying on singular parametrisations [55–57] have also been investigated. Some of these schemes are able to provide optimal convergence rates.

The outline of this chapter is as follows. In Section 4.2 the Catmull-Clark subdivision is introduced, with a review of the relevant theory on eigenanalysis of the subdivision matrix. Specifically, the necessary conditions for C^1 smoothness and boundedness of the curvature are motivated, and the local parametrisation of subdivision surfaces using the characteristic map is introduced. These are all classical results and concepts which are mostly unknown in isogeometric analysis. In Section 4.3 the proposed constrained optimisation problem and its numerical solution are discussed. Two sets of subdivision weights are derived that minimise the thin-plate energy norm errors in approximating locally cup- and saddle-like shapes. Subsequently, it is shown how a finite element solution can be locally decomposed into cup- and saddle-like components. Depending on this decomposition and the following choice of optimal weights, a second more accurate finite element analysis can be performed. In Section 4.4 the proposed approach is applied to transversally loaded thin-plate problems using meshes with extraordinary vertices and the convergence of the errors in L_2 and energy norms is reported.

4.2 Catmull-Clark subdivision surfaces

4.2.1 Refinement weights and the subdivision matrix

Catmull-Clark subdivision is a generalisation of cubic tensor-product B-splines to unstructured meshes [14]. On non-tensor-product meshes the number of faces connected to a vertex, i.e. valence v, can be different from four. The vertices with $v \neq 4$ are referred to as *extraordinary* or *star vertices*. During subdivision refinement each quadrilateral face of the control mesh is split into four faces and the coordinates of the old and new control vertices are computed with the subdivision weights given in Figure 4.1. The weights in each of the three diagrams have to be normalised so that they add up to one. The unnormalised weights assigned to the extraordinary vertex (empty circle) are denoted by α, β and γ respectively. For $v = 4$ and bivariate cubic B-splines the three weights take the values $\alpha = 8$, $\beta = 1$ and $\gamma = 1$, which after normalisation convey the same information as tensor-product of the univariate subdivision stencils shown in Figure 2.14. The new vertices introduced by the subdivision process are all regular (with $v = 4$) and the total number of irregular vertices in the mesh remains constant. That is, the irregular vertices are more and more surrounded by regular vertices.

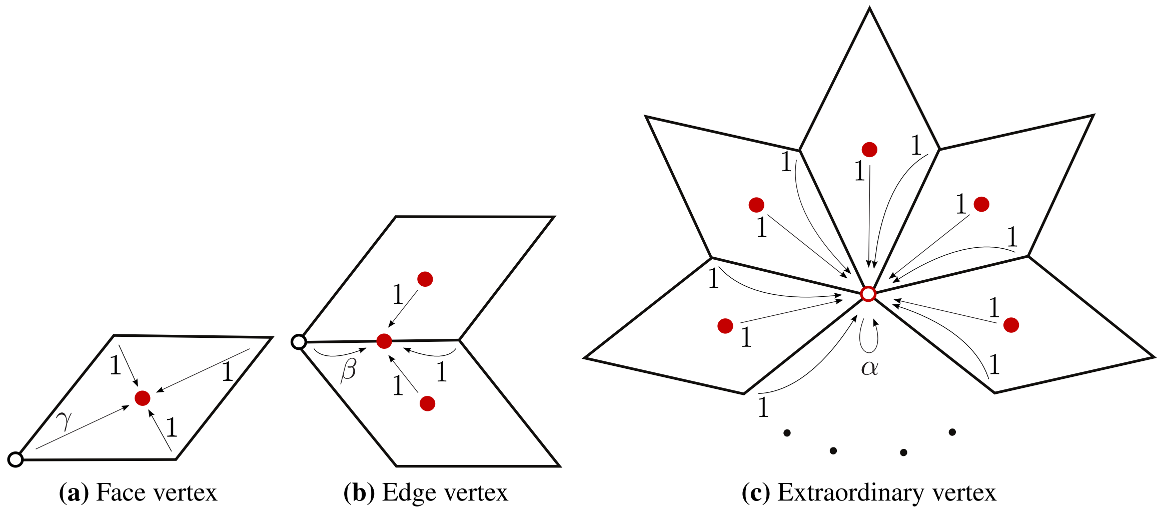


Figure 4.1: Subdivision weights for the Catmull-Clark scheme with the empty circle denoting the extraordinary vertex. The weights in each of the three diagrams have to be normalised so that they add up to one. For Catmull-Clark scheme the three weights take the values $\alpha = v(v-2)$, $\beta = 1$ and $\gamma = 1$, where v is the valence.

In order to study the smoothness behaviour of subdivision surfaces near an extraordinary vertex, it is sufficient to consider only the vertices in its immediate vicinity. A 1-neighbourhood of a vertex is formed by the union of faces that contain the vertex. The n-neighbourhood is defined recursively as the union of all 1-neighbourhoods of the $(n-1)$-neighbourhood vertices. It is assumed that the considered n-neighbourhood has only one single extraordinary vertex located at its centre. The n-neighbourhood control vertices $\boldsymbol{p}^\ell$ at the refinement level ℓ are mapped to control vertices $\boldsymbol{p}^{\ell+1}$ with the subdivision matrix $\boldsymbol{S}$,

$$\boldsymbol{p}^{\ell+1} = \boldsymbol{S}\boldsymbol{p}^\ell . \tag{4.1}$$

The square subdivision matrix $\boldsymbol{S}$ can be readily derived from the weights indicated in Figure 4.1. The control point coordinates at level ℓ are arranged in this form

$$\boldsymbol{p}^\ell = \begin{bmatrix} p^\ell_{1x} & p^\ell_{1y} & p^\ell_{1z} \\ p^\ell_{2x} & p^\ell_{2y} & p^\ell_{2z} \\ \vdots & \vdots & \vdots \end{bmatrix} \tag{4.2}$$

with each row containing the coordinates of one control point $\boldsymbol{p}_j^\ell \in \mathbb{R}^3$ with the index j.

4.2.2 Eigendecomposition of the subdivision matrix

For the tuning approach to be introduced in Section 4.3, it is necessary to consider the 3-neighbourhood around an extraordinary vertex, see Figure 4.2. The 3-neighbourhood consists

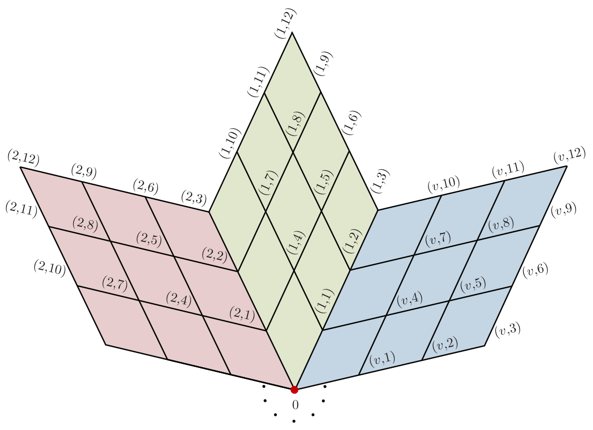


Figure 4.2: Three-rings of faces around an extraordinary vertex with valence v and the numbering of the vertices. In the index pair (s,a) the first is the segment number and the second is the vertex number.

of v segments with each segment containing $12v$ vertices, excluding the extraordinary vertex with index 0. Hence, there are $(12v+1)$ vertices so that the subdivision matrix has the dimensions $(12v+1)\times(12v+1)$. In establishing the subdivision matrix it is assumed that the index pair (s,a) is converted to a scalar index as $a+12(v-s)$. The eigenvalues and eigenvectors of $\boldsymbol{S}$ are closely related to the smoothness and other properties of the subdivision surface. The eigendecomposition of the asymmetric subdivision matrix $\boldsymbol{S}$ reads

$$\boldsymbol{S} = \sum_j \lambda_j \boldsymbol{r}_j \otimes \boldsymbol{l}_j \tag{4.3}$$

with

$$(\boldsymbol{S}-\lambda_j\boldsymbol{I})\boldsymbol{r}_j = \boldsymbol{0}, \quad \boldsymbol{l}_j^\mathsf{T}(\boldsymbol{S}-\lambda_j\boldsymbol{I}) = \boldsymbol{0}^\mathsf{T} \quad \text{and} \quad \langle \boldsymbol{l}_j, \boldsymbol{r}_k\rangle = \begin{cases} 1 & \text{if } j=k \\ 0 & \text{if } j\neq k \end{cases}, \tag{4.4}$$

where λ_j are the eigenvalues and $\boldsymbol{r}_j$, $\boldsymbol{l}_j$ are the right and left eigenvectors respectively. Throughout this chapter it is assumed that the eigenvalues are sorted in descending order with largest being λ_0. The subdivision matrix $\boldsymbol{S}$ has a cyclical structure due to the cyclic symmetry of the weights given in Figure 4.1. We assume that the vertices in the 3-neighbourhood are enumerated according to Figure 4.2.

Because of the cyclical structure, the eigendecomposition of $\boldsymbol{S}$ can be best computed with a discrete Fourier transform (DFT). As pioneered in [15], DFT is crucial in identifying the different geometric shapes described by the different eigenvectors. For transforming $\boldsymbol{S}$ the following extended DFT matrix is considered:

$$\boldsymbol{F} = \frac{1}{\sqrt{v}} \begin{bmatrix} 1 & \mathbf{0}^\mathsf{T} & \mathbf{0}^\mathsf{T} & \mathbf{0}^\mathsf{T} & \cdots & \mathbf{0}^\mathsf{T} \\ \mathbf{0} & \boldsymbol{I} & \boldsymbol{I} & \boldsymbol{I} & \cdots & \boldsymbol{I} \\ \mathbf{0} & \boldsymbol{I} & \omega\boldsymbol{I} & \omega^2\boldsymbol{I} & \cdots & \omega^{-1}\boldsymbol{I} \\ \mathbf{0} & \boldsymbol{I} & \omega^2\boldsymbol{I} & \omega^4\boldsymbol{I} & \cdots & \omega^{-2}\boldsymbol{I} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbf{0} & \boldsymbol{I} & \omega^{-1}\boldsymbol{I} & \omega^{-2}\boldsymbol{I} & \cdots & \omega\boldsymbol{I} \end{bmatrix} \tag{4.5}$$

with the complex number $\omega = \exp(i2\pi/v)$ where $i = \sqrt{-1}$, the identity matrix $\boldsymbol{I}$ of size (12×12) and the zero vector $\mathbf{0}$ of size 12. The first row and column of $\boldsymbol{F}$ have been introduced for the extraordinary vertex. In obtaining (4.5) the standard relations $\omega^{v+k} = \omega^k$ and $\omega^{v-k} = \omega^{-k} = \overline{\omega}^k$ with the complex conjugate $\overline{\omega} = \exp(-i2\pi/v)$ are used. The inverse transform $\boldsymbol{F}^{-1}$ is obtained by replacing ω with its complex conjugate $\overline{\omega}$. The subdivision matrix is Fourier transformed according to

$$\hat{\boldsymbol{S}} = \boldsymbol{F}\boldsymbol{S}\boldsymbol{F}^{-1}, \tag{4.6}$$

leading to a block diagonal matrix

$$\hat{\boldsymbol{S}} = \begin{bmatrix} \hat{\boldsymbol{S}}^{(0,0)} & & & & \\ & \hat{\boldsymbol{S}}^{(1,1)} & & & \\ & & \hat{\boldsymbol{S}}^{(2,2)} & & \\ & & & \ddots & \\ & & & & \hat{\boldsymbol{S}}^{(v-1,v-1)} \end{bmatrix}, \tag{4.7}$$

where the blocks $\hat{\boldsymbol{S}}^{(m,m)}$ are of size 13×13 for $m = 0$ and of size 12×12 for $m \neq 0$. Due to its block-diagonal structure the eigendecomposition of the transformed matrix

$$\hat{\boldsymbol{S}} = \sum_j \lambda_j \hat{\boldsymbol{r}}_j \otimes \hat{\boldsymbol{l}}_j \tag{4.8}$$

can be more readily determined. Namely, it is sufficient to consider the eigenvalue problems for each of the v blocks $\hat{\boldsymbol{S}}^{(m,m)}$ separately, i.e.,

$$\left(\hat{\boldsymbol{S}}^{(m,m)} - \lambda_n^{(m,m)} \boldsymbol{I}\right) \hat{\boldsymbol{r}}_n^{(m,m)} = \boldsymbol{0}\,, \tag{4.9}$$

where the eigenvalues within each block are also sorted in descending order, i.e, $\lambda_0^{(m,m)}$ is the largest eigenvalue in the block $\hat{\boldsymbol{S}}^{(m,m)}$. The eigenvalues λ_j and eigenvectors $\hat{\boldsymbol{r}}_j$ and $\hat{\boldsymbol{l}}_j$ are the union of all the eigenvalues and the block-wise eigenvectors. In obtaining the eigenvectors $\hat{\boldsymbol{r}}_j$ and $\hat{\boldsymbol{l}}_j$, each of size $12v+1$, the corresponding block-wise vectors $\hat{\boldsymbol{r}}_n^{(m,m)}$ and $\hat{\boldsymbol{l}}_n^{(m,m)}$ are suitably padded with zeros. The vectors $\hat{\boldsymbol{r}}_n^{(m,m)}$ and $\hat{\boldsymbol{l}}_n^{(m,m)}$ are of size 13 for $m = 0$ and of size 12 for $m \neq 0$. Moreover, the subdivision matrix $\boldsymbol{S}$ and its Fourier transform $\hat{\boldsymbol{S}}$ have the same eigenvalues λ_j and their eigenvectors are related by

$$\boldsymbol{r}_j = \boldsymbol{F}^{-1}\hat{\boldsymbol{r}}_j \quad \text{and} \quad \boldsymbol{l}_j = \boldsymbol{F}^{-1}\hat{\boldsymbol{l}}_j\,. \tag{4.10}$$

Each block $\hat{\boldsymbol{S}}^{(m,m)}$ corresponds to a specific rotational frequency $\omega_f = 2\pi m/v$. As pointed out, the eigenvectors $\hat{\boldsymbol{r}}_j$ and $\hat{\boldsymbol{l}}_j$ can have non-zero entries only in the components corresponding to a specific $\hat{\boldsymbol{r}}_n^{(m,m)}$ and $\hat{\boldsymbol{l}}_n^{(m,m)}$. Hence, the transformation of $\hat{\boldsymbol{r}}_j$ and $\hat{\boldsymbol{l}}_j$ according to (4.10) yields always a column of $\boldsymbol{F}^{-1}$ each of which corresponds to a specific rotational frequency[1]. To this end, recall the Euler identity

$$\omega^{ms} = e^{i2\pi ms/v} = \cos(2\pi ms/v) + i\sin(2\pi ms/v)\,. \tag{4.11}$$

Hence, for a fixed angular frequency $\omega_f = 2\pi m/v$ the vectors $\boldsymbol{r}_j$ and $\boldsymbol{l}_j$ will assign each control vertex (s,a) with a fixed index a, see Figure 4.2, a value that oscillates with the angular frequency $2\pi m/v$ while circumnavigating the extraordinary vertex by incrementing $s \in \{1,\cdots,v\}$.

Furthermore, for geometric interpretation of the eigendecomposition it is helpful to realise that most of the eigenvalues λ_j have the multiplicity of two. That is, the eigenvalues $\lambda_n^{(m,m)}$ and $\lambda_n^{(v-m,v-m)}$ are identical in the blocks $m \geq 1$. The corresponding eigenvectors $\boldsymbol{r}_j$ and $\boldsymbol{l}_j$ have the same eigenfrequency because the columns m and $v-m$ of the DFT matrix $\boldsymbol{F}^{-1}$ are the complex conjugates of each other.

[1]The first columns and rows of $\boldsymbol{F}$ and $\boldsymbol{F}^{-1}$ are assigned to the extraordinary vertex and do not represent harmonics.

4.2.3 Limit analysis and smoothness

The eigenvalues λ_j and eigenvectors $\hat{\boldsymbol{r}}_j$ and $\hat{\boldsymbol{l}}_j$ of the Fourier transformed subdivision matrix $\hat{\boldsymbol{S}}$ have to satisfy certain conditions for a subdivision scheme leading to a smooth well-defined surface, see [15, 43, 44]. To understand this, consider the projection of control mesh vertex coordinates at subdivision level $\ell = 0$ into the eigenspace of the subdivision matrix, using the orthogonality of left and right eigenvectors (4.4),

$$\boldsymbol{p}^{\ell=0} = \boldsymbol{r}_0\langle\boldsymbol{l}_0,\boldsymbol{p}^0\rangle + \boldsymbol{r}_1\langle\boldsymbol{l}_1,\boldsymbol{p}^0\rangle + \boldsymbol{r}_2\langle\boldsymbol{l}_2,\boldsymbol{p}^0\rangle + \cdots + \boldsymbol{r}_{12v}\langle\boldsymbol{l}_{12v},\boldsymbol{p}^0\rangle\,, \tag{4.12}$$

where each of the scalar products $\langle\ ,\ \rangle$ yield a row vector with 3 components. Subdividing the 3-neighbourhood in the eigenspace, while considering the eigendecomposition (4.4), gives

$$\boldsymbol{S}\boldsymbol{p}^0 = \lambda_0\boldsymbol{r}_0\langle\boldsymbol{l}_0,\boldsymbol{p}^0\rangle + \lambda_1\boldsymbol{r}_1\langle\boldsymbol{l}_1,\boldsymbol{p}^0\rangle + \lambda_2\boldsymbol{r}_2\langle\boldsymbol{l}_2,\boldsymbol{p}^0\rangle + \cdots + \lambda_{12v}\boldsymbol{r}_{12v}\langle\boldsymbol{l}_{12v},\boldsymbol{p}^0\rangle\,. \tag{4.13}$$

Hence, the repeated subdivision of the 3-neighbourhood can be simply achieved with

$$\boldsymbol{S}^\ell\boldsymbol{p}^0 = \lambda_0^\ell\boldsymbol{r}_0\langle\boldsymbol{l}_0,\boldsymbol{p}^0\rangle + \lambda_1^\ell\boldsymbol{r}_1\langle\boldsymbol{l}_1,\boldsymbol{p}^0\rangle + \lambda_2^\ell\boldsymbol{r}_2\langle\boldsymbol{l}_2,\boldsymbol{p}^0\rangle + \cdots + \lambda_{12v}^\ell\boldsymbol{r}_{12v}\langle\boldsymbol{l}_{12v},\boldsymbol{p}^0\rangle\,. \tag{4.14}$$

From this equation it is evident that the properties of a subdivision surface are widely governed by the eigenstructure of the subdivision matrix. The subdivision matrix $\boldsymbol{S}$ is a stochastic matrix, i.e. only positive entries and each row adds up to 1, so that its largest eigenvalue is $\lambda_0 = 1$ and the components of the corresponding eigenvector $\boldsymbol{r}_0$ are all equal to 1. In the limit $\ell \to \infty$ all control vertices converge to $\langle\boldsymbol{l}_0,\boldsymbol{p}^0\rangle$. The first term in (4.14) can be eliminated by translating the initial control vertex coordinates by $-\boldsymbol{r}_0\langle\boldsymbol{l}_0,\boldsymbol{p}^0\rangle$. Without loss of generality, in the following we assume that the coordinate system for 3-neighbourhood has been chosen so that the first term in (4.14) is zero, that is,

$$\boldsymbol{S}^\ell\boldsymbol{p}^0 = \lambda_1^\ell\boldsymbol{r}_1\langle\boldsymbol{l}_1,\boldsymbol{p}^0\rangle + \lambda_2^\ell\boldsymbol{r}_2\langle\boldsymbol{l}_2,\boldsymbol{p}^0\rangle + \cdots + \lambda_{12v}^\ell\boldsymbol{r}_{12v}\langle\boldsymbol{l}_{12v},\boldsymbol{p}^0\rangle\,. \tag{4.15}$$

For a (symmetric) C^1-continuous subdivision surface the subdominant eigenvalues λ_1 and λ_2 have to satisfy the following relationship:

$$\lambda_1 = \lambda_2 > \lambda_3\,. \tag{4.16}$$

In addition, the corresponding eigenvectors $\boldsymbol{r}_1$, $\boldsymbol{l}_1$, $\boldsymbol{r}_2$ and $\boldsymbol{l}_2$ have to come from the eigendecomposition of the blocks $\hat{\boldsymbol{S}}^{(1,1)}$ and $\hat{\boldsymbol{S}}^{(v-1,v-1)}$ [43, 44]. As discussed in Section 4.2.2, owing to the symmetry properties of the Fourier transformation, $\hat{\boldsymbol{S}}^{(1,1)}$ and $\hat{\boldsymbol{S}}^{(v-1,v-1)}$ have the same eigenvalues, and the eigenvectors $\boldsymbol{r}_2$ and $\boldsymbol{l}_2$ are the complex conjugates of $\boldsymbol{r}_1$ and $\boldsymbol{l}_1$. All the four eigenvectors $\boldsymbol{r}_1$, $\boldsymbol{l}_1$, $\boldsymbol{r}_2$ and $\boldsymbol{l}_2$ are usually complex and have the angular frequency $\omega_f = 2\pi/v$. A set of real eigenvectors each of size $12v+1$ representing vertex values can be obtained as the linear combination of the complex ones, e.g., with $\frac{1}{2}(\boldsymbol{r}_1 + \boldsymbol{r}_2)$ and $\frac{1}{2i}(\boldsymbol{r}_1 - \boldsymbol{r}_2)$ where $i = \sqrt{-1}$. To avoid a proliferation of symbols we will use the same symbols for the so-computed real and complex eigenvectors.

A necessary condition for the C^2-continuity of a subdivision surface (with no artificial flat spots) is that the subsubdominant eigenvalues satisfy

$$\lambda_3 = \lambda_1^2, \quad \lambda_4 = \lambda_1^2, \quad \lambda_3 = \lambda_4 = \lambda_5 > \lambda_6 \tag{4.17}$$

and the corresponding eigenvectors come from the eigendecomposition of the blocks $\hat{\boldsymbol{S}}^{(0,0)}$, $\hat{\boldsymbol{S}}^{(2,2)}$ and $\hat{\boldsymbol{S}}^{(v-2,v-2)}$ [15, 44]. Remember that $\lambda_4 = \lambda_5$ is naturally satisfied due to the duplicity of eigenvalues from blocks $\hat{\boldsymbol{S}}^{(m,m)}$ and $\hat{\boldsymbol{S}}^{(v-m,v-m)}$ when $m \geq 1$, as mentioned in Section 4.2.2.

4.2.4 Characteristic map

As first proposed in [43] the characteristic map provides a means for parametrisation of the surface generated by a subdivision scheme. Parametrisation of the subdivision surface, at least a local one, is essential in order to associate the so far discrete representation based on control vertices with a continuous differentiable representation. This is, for instance, required for finite element analysis using subdivision surfaces. The characteristic map is defined using the two real right eigenvectors $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ corresponding to the subdominant eigenvalue $\lambda_1 = \lambda_2$. The characteristic control mesh shown in Figure 4.3 representing the 3-neighbourhood around an extraordinary vertex has the coordinates

$$\boldsymbol{p}_c^0 = \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{0} \end{bmatrix}, \tag{4.18}$$

where the third *out-of-plane* coordinate is chosen as $\boldsymbol{0}$. As discussed in Section 4.2.2, recall that the two eigenvectors $\boldsymbol{r}_1$ and $\boldsymbol{r}_2$ have the angular frequency $\omega_f = 2\pi/v$ and have been chosen so that they are orthogonal in the plane spanned by the corresponding two complex eigenvectors. This can be done without loss of generality because the subdivision construction is invariant under affine transformations. Hence, the coordinates of control vertices (s,a) oscillate with $\cos(2\pi s/v)$ in the horizontal direction and with $\sin(2\pi s/v)$ in the vertical direction leading to the shown characteristic control mesh in Figure 4.3.

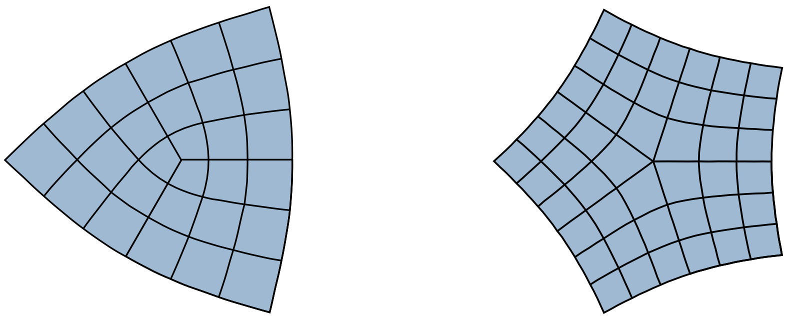


Figure 4.3: Characteristic control mesh of Catmull-Clark scheme for valence $v = 3$ (left) and $v = 5$ (right).

As suggested in [43], the planar surface described by the characteristic control mesh can be used for the parametrisation of subdivision surfaces. To this end, first consider the subdivision refinement of the characteristic mesh. According to (4.15) and the orthogonality of left and right eigenvectors (4.4), the subdivision refinement of the characteristic mesh simply yields a scaled version of the same mesh:

$$\boldsymbol{p}_c^\ell = \boldsymbol{S}^\ell \boldsymbol{p}_c^0 = \begin{bmatrix} \lambda_1^\ell \boldsymbol{r}_1 & \lambda_2^\ell \boldsymbol{r}_2 & \boldsymbol{0} \end{bmatrix} . \tag{4.19}$$

Hence, the refined control mesh is simply obtained by scaling the control mesh by $\lambda_1 = \lambda_2$. Repeated subdivision yields repeated scaling of the control mesh. This combined with the fact that the Catmull-Clark scheme leads to bivariate cubic B-splines in patches with only ordinary vertices is used for parametrising the subdivision surface. During subdivision refinement each patch is split into four patches. In particular, in the patches adjacent to the extraordinary vertex three of the created patches have only regular vertices and can be parametrised with bivariate cubic B-splines.

With repeated refinement more and more of the subdivision surface can be parametrised with cubic B-splines. A practical algorithm for efficient implementation of this parametrisation has been introduced in [40]. Without going into details we define the bijective characteristic map

$$\chi : (\boldsymbol{\eta}, s) \in (\Omega, s) \mapsto \boldsymbol{\xi} \in \Omega_\chi \tag{4.20}$$

with $\boldsymbol{\eta} = (\eta_1, \eta_2)$ and $\boldsymbol{\xi} = (\xi_1, \xi_2)$, which maps a set of square domains (Ω, s) with $s \in \mathbb{N}^+$ representing the faces in the control mesh into the characteristic domain Ω_χ. The smooth parametrisation provided by the characteristic map χ is illustrated in Figure 4.4. With the subdivision basis functions $\boldsymbol{N}(\boldsymbol{\eta}, s)$, consisting of cubic B-splines and obtained according

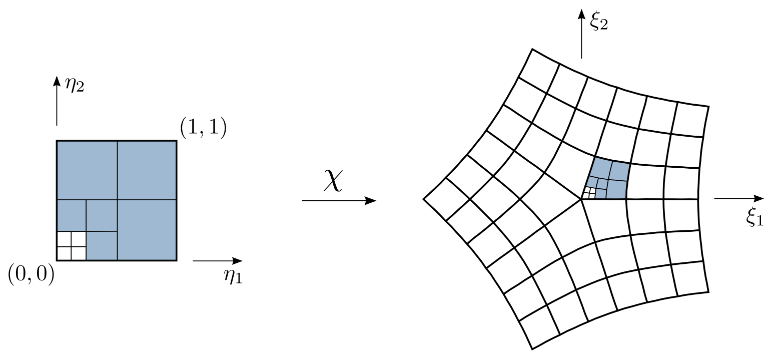


Figure 4.4: Characteristic map from a unit square to the characteristic domain.

to [40], the characteristic map can be written as

$$\boldsymbol{\xi} = \chi(\boldsymbol{\eta}, s) = \boldsymbol{N}^{\mathsf{T}}(\boldsymbol{\eta}, s) \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 \end{bmatrix} . \tag{4.21}$$

For brevity, in the following we omit the face index in the basis function $\boldsymbol{N}(\boldsymbol{\eta}, s)$ and write $\boldsymbol{N}(\boldsymbol{\eta})$. In Section 3.2.3, taking the univariate basis as an example, we have demonstrated the whole procedure of using Stam's algorithm [40] to evaluate the basis functions $\boldsymbol{N}(\eta)$ in "irregular" region with non-uniform knots. The same process is readily extended to evaluate the bivariate basis function $\boldsymbol{N}(\boldsymbol{\eta})$ around extraordinary vertices. In this case, the number of supporting control points for each regular cubic B-spline patch (shaded in blue in Figure 4.4) is sixteen and the subdivision weights given in Figure 4.1 should be used.

4.3 Optimisation of subdivision weights

We aim to modify the subdivision weights α, β and γ of the Catmull-Clark scheme, see Figure 4.1, to improve its approximation properties when used in finite element analysis. As known in CAD, not all parameters yield visually appealing surfaces even when they give C^1-continuous surfaces, for a quantitative analysis see [44]. Small surface oscillations, i.e. ripples, appear when the represented surface is not planar.

4.3.1 Preliminaries

First, we consider the representation of a polynomial scalar field $u(\xi_1, \xi_2)$ over the characteristic domain Ω_χ. It is assumed that the scalar field is given in the form

$$\begin{aligned} u(\xi_1,\xi_2) &= c_0 + c_1\xi_1 + c_2\xi_2 + c_3(\xi_1^2+\xi_2^2) + c_4(\xi_1^2-\xi_2^2) + c_5(2\xi_1\xi_2) + \dots \\ &= c_0u_0 + c_1u_1(\xi_1) + c_2u_2(\xi_2) + c_3u_3(\xi_1,\xi_2) \\ &\quad + c_4u_4(\xi_1,\xi_2) + c_5u_5(\xi_1,\xi_2) + \dots , \end{aligned} \tag{4.22}$$

where $c_j \in \mathbb{R}$ and the functions u_j on the second line are introduced for notational convenience. The chosen functions u_0, $u_1(\xi_1)$, $u_2(\xi_2)$, $u_3(\xi_1,\xi_2)$, $u_4(\xi_1,\xi_2)$ and $u_5(\xi_1,\xi_2)$ can represent all quadratics and their choice will be discussed further below. The approximation of u_j over Ω_χ can be studied by comparing with it the limit surface resulted from the control points

$$\boldsymbol{p}^0_{u_j} = \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & u_j(\boldsymbol{r}_1,\boldsymbol{r}_2) \end{bmatrix} , \tag{4.23}$$

where the third coordinate is a vector formed by the scalar function $u_j(\xi_1,\xi_2)$ evaluated at the vertex locations $[\boldsymbol{r}_1\ \boldsymbol{r}_2]$, row by row. The linear functions $u_1(\xi_1)$ and $u_2(\xi_2)$ can be exactly represented so that we are mainly concerned about the quadratic terms $u_3(\xi_1,\xi_2)$, $u_4(\xi_1,\xi_2)$ and $u_5(\xi_1,\xi_2)$.

The specific form of the quadratic functions in (4.22) is motivated by the eigenstructure of the subdivision matrix $\boldsymbol{S}$, see Section 4.2.2. Specifically, the control point values $\boldsymbol{r}_3$, $\boldsymbol{r}_4$ and $\boldsymbol{r}_5$ and the corresponding control point values $u_3(\boldsymbol{r}_1,\boldsymbol{r}_2)$, $u_4(\boldsymbol{r}_1,\boldsymbol{r}_2)$ and $u_5(\boldsymbol{r}_1,\boldsymbol{r}_2)$ have matching angular frequencies over the 3-neighbourhood of the extraordinary vertex[2]. It is straightforward to confirm the orthogonality relations

$$\langle u_j(\boldsymbol{r}_1,\boldsymbol{r}_2), \boldsymbol{l}_k\rangle = \begin{cases} \neq 0 & \text{for } j = k \\ = 0 & \text{for } j \neq k \end{cases} \quad \text{with } j,k \in \{3,4,5\} . \tag{4.24}$$

According to (4.12), the projection of the control vertex coordinates $\boldsymbol{p}^0_{u_j}$ into the eigenspace of the subdivision matrix $\boldsymbol{S}$, while neglecting the terms with higher orders than quadratic, yields

$$\boldsymbol{p}^0_{u_j} = \begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_j\langle \boldsymbol{l}_j, u_j(\boldsymbol{r}_1,\boldsymbol{r}_2)\rangle \end{bmatrix} . \tag{4.25}$$

[2]In order for the phase to match, the indexing of the vertices has to begin along the edge aligned with the ξ_1-axis.

With the eigendecomposition (4.4) the subdivision refinement of this control mesh gives

$$\boldsymbol{p}_{u_j}^{\ell} = \boldsymbol{S}^{\ell}\boldsymbol{p}_{u_j}^{0} = \begin{bmatrix} \lambda_1^{\ell}\boldsymbol{r}_1 & \lambda_2^{\ell}\boldsymbol{r}_2 & \lambda_j^{\ell}\boldsymbol{r}_j\langle \boldsymbol{l}_j, u_j(\boldsymbol{r}_1,\boldsymbol{r}_2)\rangle \end{bmatrix}, \tag{4.26}$$

That is, the subdivision refinement of the first two components yields the characteristic domain and the third component yields the graph of the surface $u_j^h(\xi_1,\xi_2)$ approximating $u_j(\xi_1,\xi_2)$, see Figure 4.5. The corresponding limit surface $u_j^h(\xi_1,\xi_2)$ has, according to (4.21), the following form:

$$\begin{bmatrix} \boldsymbol{\xi} & u_j^h \end{bmatrix} = \boldsymbol{N}^{\mathsf{T}}(\boldsymbol{\eta})\begin{bmatrix} \boldsymbol{r}_1 & \boldsymbol{r}_2 & \boldsymbol{r}_j\langle \boldsymbol{l}_j, u_j(\boldsymbol{r}_1,\boldsymbol{r}_2)\rangle \end{bmatrix}. \tag{4.27}$$

To compare the shapes $u_j^h(\xi_1,\xi_2)$ and $u_j(\xi_1,\xi_2)$ quantitatively, we introduce the thin-plate energy norm

$$\|u\|_e^2 = \int_{\Omega}\left(\frac{\partial^2 u}{\partial \xi_1^2}+\frac{\partial^2 u}{\partial \xi_2^2}\right)^2 - 2(1-\mu)\left(\frac{\partial^2 u}{\partial \xi_1^2}\frac{\partial^2 u}{\partial \xi_2^2} - \left(\frac{\partial^2 u}{\partial \xi_1 \partial \xi_2}\right)^2\right)\mathrm{d}\Omega, \tag{4.28}$$

with the Poisson ratio $\mu = 0.3$. Moreover, the necessary conditions for C^2-continuity given

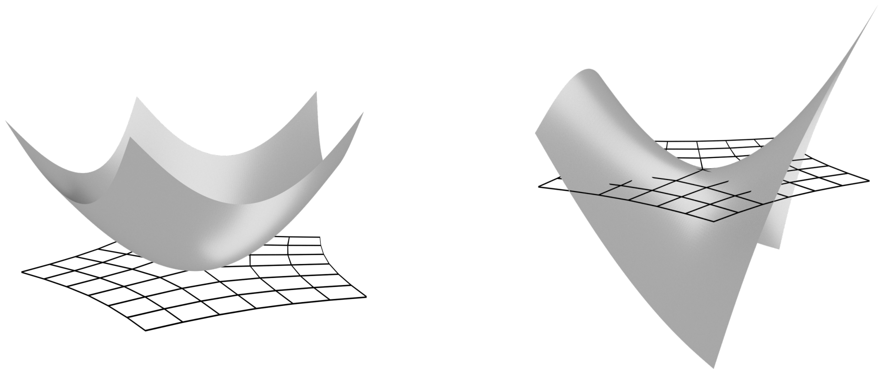

(a) Cup-like geometry $u_3^h(\xi_1,\xi_2)$ (b) Saddle-like geometry $u_4^h(\xi_1,\xi_2)$

Figure 4.5: Quadratic shapes over a characteristic control mesh with valence $v = 5$. Note that $u_5^h(\xi_1,\xi_2)$ has the same shape like $u_4^h(\xi_1,\xi_2)$, but only rotated by $\pi/4$ in the $\xi_1\xi_2$-plane.

in (4.17), repeated here for convenience,

$$\lambda_3 = \lambda_1^2, \quad \lambda_4 = \lambda_1^2, \quad \text{and} \quad \lambda_5 = \lambda_1^2 \quad \text{with } \lambda_1 = \lambda_2$$

can now be related to the curvature of the three quadratic limit surfaces resulted from repeated refinement of $\langle u_j(\boldsymbol{r}_1,\boldsymbol{r}_2), \boldsymbol{l}_j\rangle \boldsymbol{r}_j$ with $j \in \{3,4,5\}$. In order for the limit surfaces $u_j^h(\xi_1,\xi_2)$ to have finite curvature at the extraordinary vertex, when the first two control vertex components scale with $\lambda_1 (= \lambda_2)$ the third has to scale with λ_1^2.

4.3.2 Constrained optimisation

The constrained optimisation problem for determining the subdivision weights α, β and γ that minimise the error in approximating quadratic surfaces is formulated as

$$
\begin{aligned}
&\underset{\alpha,\beta,\gamma}{\text{minimise}} && \frac{\| u_j^h(\xi_1, \xi_2; \alpha, \beta, \gamma) - u_j(\xi_1, \xi_2) \|_e}{\| u_j(\xi_1, \xi_2) \|_e} && \text{(4.29a)}\\
&\text{subject to:} && \lambda_1(\beta, \gamma) = \lambda_2(\beta, \gamma) && \text{(4.29b)}\\
& && \lambda_3(\alpha, \beta, \gamma) = \lambda_1^2(\beta, \gamma) && \text{(4.29c)}\\
& && \lambda_4(\beta, \gamma) = \lambda_1^2(\beta, \gamma) && \text{(4.29d)}\\
& && \lambda_5(\beta, \gamma) = \lambda_4(\beta, \gamma) \,, && \text{(4.29e)}
\end{aligned}
$$

with $j \in \{3, 4, 5\}$ and the constraints representing the necessary C^2-continuity conditions (4.16) and (4.17). As mentioned in Section 4.2.3, owing to the symmetries of the DFT, the constraints (4.29b) and (4.29e) are automatically satisfied. Hence, the constraints reduce to two independent equations for the three unknowns. To reduce the constrained optimisation problem into an unconstrained one, it is convenient to first solve the nonlinear system of equations

$$
\begin{aligned}
\lambda_1(\beta, \gamma) &= \lambda \,, && \text{(4.30a)}\\
\lambda_4(\beta, \gamma) &= \lambda_1^2(\beta, \gamma) \,, && \text{(4.30b)}\\
\lambda_3(\alpha, \beta, \gamma) &= \lambda_1^2(\beta, \gamma) \,. && \text{(4.30c)}
\end{aligned}
$$

That is, to determine the dependence of the weights $\alpha(\lambda)$, $\beta(\lambda)$ and $\gamma(\lambda)$ on the variable λ. To solve (4.30) we use in our implementation the Python library SciPy, to be more specific, the quasi-Newton method with a BFGS update with a suitable cost function. However, $\beta(\lambda)$ and $\gamma(\lambda)$ can become complex for some λ values [48]. For Catmull-Clark, it is smaller λ values which result in complex weights. For instance, there is no real solution for β and γ for $\lambda \leq 0.608$ in case of valence $v = 5$. Instead of excluding λ values leading to complex weights we relax the second constraint (4.30b) by considering the modified constraint equations

$$\lambda_1(\beta,\gamma) = \lambda\,, \tag{4.31a}$$

$$\beta = \gamma, \tag{4.31b}$$

$$\lambda_3(\alpha,\beta,\gamma) = \lambda_1^2(\beta,\gamma)\,. \tag{4.31c}$$

In implementations where the boundedness of curvature must be satisfied, one can constrain the λ value in a valid range or consider more degrees of freedom for optimisation [48, 58]. In our numerical experiments, we found that considering the modified constraint equations leads to smaller energy norm errors in comparison to constraining the range of possible λ values. After solving (4.30) or (4.31) and determining $\alpha(\lambda)$, $\beta(\lambda)$, and $\gamma(\lambda)$ the constrained optimisation problem (4.29a) can now be restated as an unconstrained problem

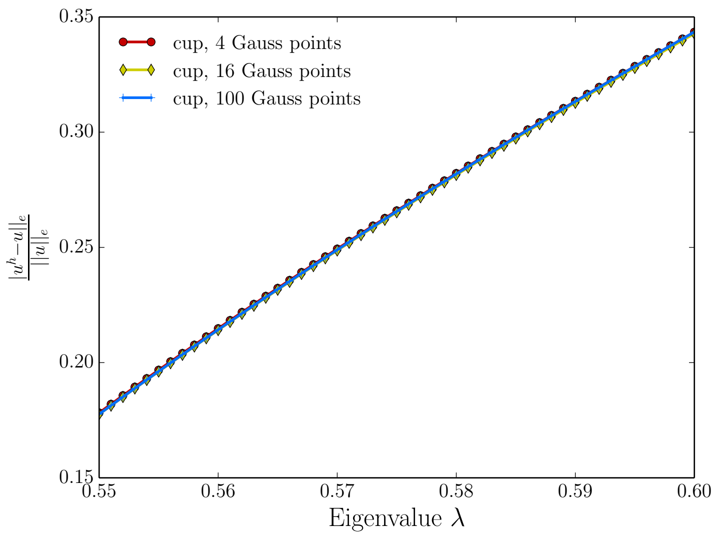
$$\underset{\lambda}{\text{minimise}} \quad \frac{\|u_j^h(\xi_1,\xi_2;\alpha(\lambda),\beta(\lambda),\gamma(\lambda)) - u_j(\xi_1,\xi_2)\|_e}{\|u_j(\xi_1,\xi_2)\|_e}\,, \tag{4.32}$$

which is a one-dimensional optimisation problem that can be solved by direct search.

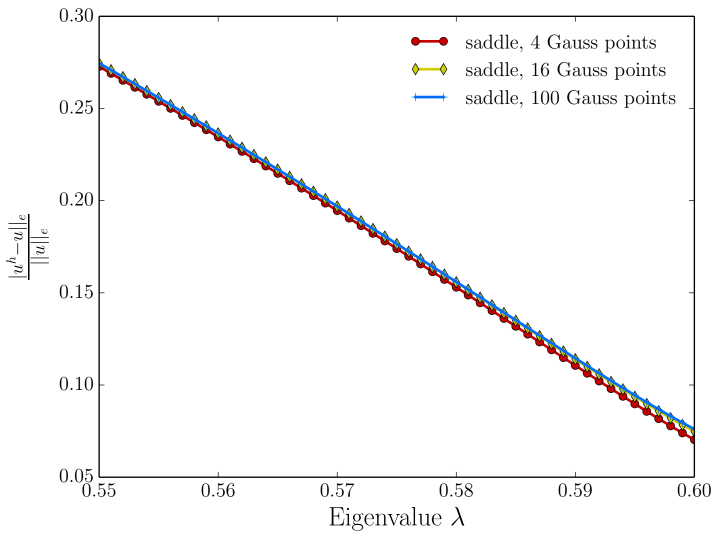
4.3.3 Optimised weights for valence $v = 5$ vertices

As an example for obtaining optimised weights, we consider the valence $v = 5$ vertex case. The proposed optimisation follows the same procedure regardless of valence. It is sufficient to consider only the approximation of the quadratic functions $u_3(\xi_1,\xi_2) = \xi_1^2 + \xi_2^2$ and $u_4(\xi_1,\xi_2) = \xi_1^2 - \xi_2^2$ with cup-like and saddle-like geometries, respectively. The function $u_5(\xi_1,\xi_2) = 2\xi_1\xi_2$ has the same saddle-like geometry as $u_4(\xi_1,\xi_2)$, only rotated by $\pi/4$ in the $\xi_1\xi_2$-plane. During optimisation the thin-plate energy norms in (4.32) are evaluated in the 2-neighbourhood of the extraordinary vertex, which is the same as the support size of the basis functions. This explains why we consider 3-neighbourhood around an extraordinary vertex, see Figure 4.2, because the evaluation in the second-ring elements needs the third-ring control vertices.

Figures 4.6a and 4.6b show the relative energy norm errors in approximating cup- and saddle-like geometries, respectively, when the subdominant eigenvalue λ and number of Gauss integration points are varied. It can be seen that while λ has a significant influence on the error the number of integration points appears to be irrelevant. Figure 4.7 shows the relative energy norm error both in cup- and saddle-like geometries when 4×4 integration points are used. In comparison to Catmull-Clark weights, also indicated in Figure 4.7, for $\lambda \in [0.550, 0.585]$ the optimised subdivision weights lead to a reduction of errors in both



(a) Cup-like geometry



(b) Saddle-like geometry

Figure 4.6: Relative energy norm error in dependence of the sub-dominant eigenvalue λ and number of integration points.

cup- and saddle-like geometries. Moreover, the most optimal value for the cup-like geometry is $\lambda = 0.550$ and for the saddle-like geometry is $\lambda = 0.585$, see Table 4.1 for the values of the optimised weights. According to [44], the obtained subdivision surfaces, same as original Catmull-Clark scheme, are C^1-continuous at extraordinary vertices and C^2 everywhere else, because the eigenvalues satisfy the required relations and the characteristic map is regular and injective.

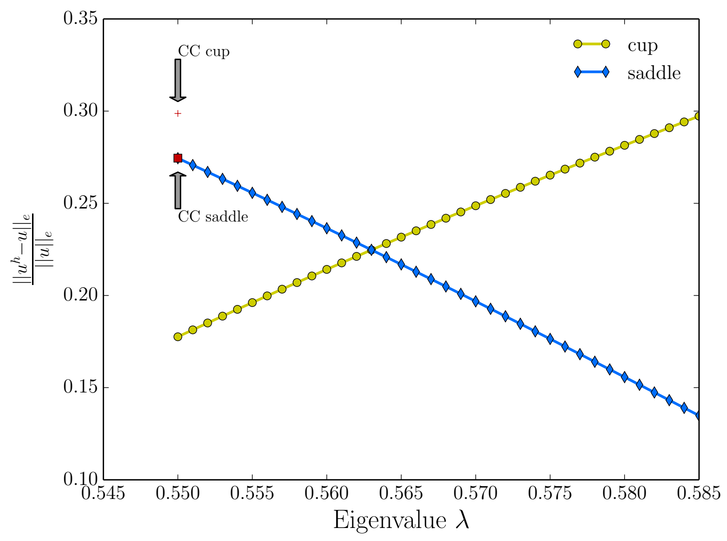


Figure 4.7: Relative energy norm error for cup- and saddle-like geometries in dependence of the sub-dominant eigenvalue λ and for the Catmull-Clark(CC) scheme. The number of integration points used for all data values is 4×4. The most optimal value for the cup-like geometry is $\lambda = 0.550$, while for the saddle-like geometry it is $\lambda = 0.585$. See Table 4.1 for the values of the optimised weights.

Table 4.1: Optimised weights in Catmull-Clark subdivision scheme for valence $v = 5$ vertices.

	α	β	γ	$\lambda_1 = \lambda_2$
Cup	13.4575	0.999938	0.999938	0.550
Saddle	13.9851	0.824885	0.824885	0.585
Original [14]	15	1	1	0.550

4.3.4 Application-dependent choice of refinement weights

When subdivision surfaces are used for finite element analysis, the solution field has quite often a mixture of both cup- and saddle-like components. And the solution field at a specific extraordinary vertex is only known after the finite element analysis. Therefore, in a first step we use the optimal weights for the cup-like geometry to obtain an initial finite element solution. Afterwards, for each extraordinary vertex with valence $v \geq 5$, a local shape decomposition is performed to determine whether the local solution is cup or saddle dominated. If the cup component dominates, the optimal weights for the cup-like geometry are chosen. If instead the saddle component dominates, the optimal weights for the saddle-like geometry are chosen. After the optimal weights for each extraordinary vertex are chosen, a second finite element analysis is performed to obtain the final solution with smaller discretisation errors.

For thin-plate and thin-shell finite element problems the decomposition of a solution into cup- and saddle-like components may be accomplished as described in the following. Suppose that the local finite element solution in a 3-neighbourhood of an extraordinary vertex is denoted as $\boldsymbol{p}^h$ and has the dimensions $(12v+1)\times 3$. In a coordinate system centred at the limit position of the extraordinary vertex, according to (4.15), we can write

$$\boldsymbol{p}^h = \sum_{j=1}^{12v} \boldsymbol{r}_j \langle \boldsymbol{l}_j, \boldsymbol{p}^h \rangle \,. \tag{4.33}$$

The corresponding limit surface has at the extraordinary vertex the normal vector $\boldsymbol{n} \in \mathbb{R}^3$, defined by

$$\boldsymbol{n} = \frac{\langle \boldsymbol{l}_1, \boldsymbol{p}^h \rangle \times \langle \boldsymbol{l}_2, \boldsymbol{p}^h \rangle}{|\langle \boldsymbol{l}_1, \boldsymbol{p}^h \rangle \times \langle \boldsymbol{l}_2, \boldsymbol{p}^h \rangle|} \,, \tag{4.34}$$

where the vectors $\langle \boldsymbol{l}_1, \boldsymbol{p}^h \rangle$ and $\langle \boldsymbol{l}_2, \boldsymbol{p}^h \rangle$ represent the two, usually non-orthogonal, tangent vectors. Multiplying (4.33) with the normal vector gives by eliminating its first two terms

$$\boldsymbol{p}^h \boldsymbol{n}^\mathsf{T} = \sum_{j=3}^{12v} \boldsymbol{r}_j \langle \boldsymbol{l}_j, \boldsymbol{p}^h \rangle \boldsymbol{n}^\mathsf{T} \,. \tag{4.35}$$

The vector $\boldsymbol{p}^h \boldsymbol{n}^\mathsf{T}$ represents the out-of-plane coordinates of the control points in a coordinate system aligned with the tangent plane at the extraordinary vertex. The corresponding limit surface over the characteristic domain has the following representation:

$$\begin{aligned} u^h(\xi_1,\xi_2) &= \boldsymbol{N}^\mathsf{T}\left(\chi^{-1}(\xi_1,\xi_2)\right) \boldsymbol{p}^h \boldsymbol{n}^\mathsf{T} \\ &= \boldsymbol{N}^\mathsf{T}\left(\chi^{-1}(\xi_1,\xi_2)\right) \Big(\boldsymbol{r}_3 \langle \boldsymbol{l}_3, \boldsymbol{p}^h \boldsymbol{n}^\mathsf{T} \rangle + \boldsymbol{r}_4 \langle \boldsymbol{l}_4, \boldsymbol{p}^h \boldsymbol{n}^\mathsf{T} \rangle + \boldsymbol{r}_5 \langle \boldsymbol{l}_5, \boldsymbol{p}^h \boldsymbol{n}^\mathsf{T} \rangle + \cdots \Big) \end{aligned} \tag{4.36}$$

and can be approximated with quadratic functions $u_3(\xi_1,\xi_2) = \xi_1^2 + \xi_2^2$, $u_4(\xi_1,\xi_2) = \xi_1^2 - \xi_2^2$ and $u_5(\xi_1,\xi_2) = 2\xi_1\xi_2$, see (4.22), such that

$$\begin{aligned} u^h(\xi_1,\xi_2) \approx &\left(\frac{\lambda_3}{\lambda_1^2}\right)^\ell \frac{\langle \boldsymbol{l}_3, \boldsymbol{p}^h \boldsymbol{n}^\mathsf{T} \rangle}{\langle \boldsymbol{l}_3, u_3(\boldsymbol{r}_1, \boldsymbol{r}_2) \rangle} (\xi_1^2 + \xi_2^2) \\ &+ \left(\frac{\lambda_4}{\lambda_1^2}\right)^\ell \frac{\langle \boldsymbol{l}_4, \boldsymbol{p}^h \boldsymbol{n}^\mathsf{T} \rangle}{\langle \boldsymbol{l}_4, u_4(\boldsymbol{r}_1, \boldsymbol{r}_2) \rangle} (\xi_1^2 - \xi_2^2) \\ &+ \left(\frac{\lambda_5}{\lambda_1^2}\right)^\ell \frac{\langle \boldsymbol{l}_5, \boldsymbol{p}^h \boldsymbol{n}^\mathsf{T} \rangle}{\langle \boldsymbol{l}_5, u_5(\boldsymbol{r}_1, \boldsymbol{r}_2) \rangle} (2\xi_1\xi_2) + \cdots \\ := &\, k_3(\xi_1^2 + \xi_2^2) + k_4(\xi_1^2 - \xi_2^2) + k_5(2\xi_1\xi_2) + \cdots \,, \end{aligned} \tag{4.37}$$

where ℓ denotes the refinement level required to evaluate at the point (ξ_1, ξ_2) using the Stam [40] algorithm. The refinement level dependent factors always vanish when, as required for curvature continuity, $\lambda_j = \lambda_1^2$ for $j \in 3,4,5$. After computing the energy densities, i.e. the integrand in (4.28), for each of the three quadratic components their ratio can be determined. Moreover, the two last terms with saddle-like geometries are energetically equivalent so that their components can be combined. This gives the following ratio between cup- and saddle-like energies:

$$R = \frac{k_3^2 \|u_3\|_e^2}{k_4^2 \|u_4\|_e^2 + k_5^2 \|u_5\|_e^2} = \frac{k_3^2}{(k_4^2 + k_5^2)} \frac{1+\mu}{1-\mu} . \tag{4.38}$$

In numerical computations the ratio R is used to decide which set of subdivision refinement weights to use.

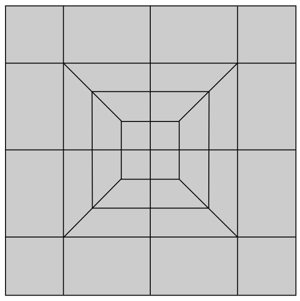
4.4 Examples

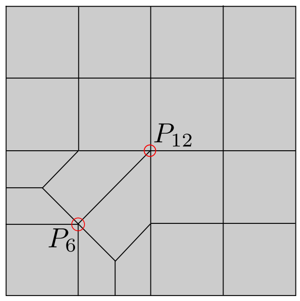
We consider the finite element analysis of thin plates to demonstrate the benefits of the optimised subdivision weights over Catmull-Clark weights. The plates are square shaped, simply supported and subjected to either uniform or sinusoidal distributed transversal loads, see Table 4.2. The thin plate energy functional depends on the second derivatives of the displacement field, c.f. (4.28). Hence, the accurate approximation of the quadratic terms in the solution field is crucial. For details of finite element implementation we refer to [6, 41]. The analytical solutions of all the computed problems are known and can be found in [59, Chapter 5]. Two different unstructured control meshes shown in Figure 4.8 are used in the

Table 4.2: Geometry, material and loading of the computed thin plates.

Length	$L_x = 10$, $L_y = 10$
Thickness	$t = 0.1$
Young's modulus	$E = 200 \times 10^9$
Poisson's ratio	$\mu = 0.3$
Uniform loading	$p = 10^4$
Sinusoidal loading	$p_s = p \sin(2\pi x/L_x) \sin(2\pi y/L_y)$

numerical computations. Both meshes have extraordinary vertices with valences $v = 3$ and $v = 5$. We do not use optimised subdivision weights for valence $v = 3$ vertices with subdominant eigenvalues $\lambda_1 = \lambda_2 < 0.5$. During subdivision refinement their 1-neighbourhoods shrink faster than the 1-neighbourhoods of other vertices with $\lambda_1 = \lambda_2 \geq 0.5$, see also Figure 4.3. Hence, it can be expected that the benefits of optimising the weights of vertices with


(a) Symmetric mesh



(b) Asymmetric mesh

Figure 4.8: Initial unstructured coarse control meshes with level $\ell = 0$.

$v = 3$ will be negligible. For each valence $v = 5$ vertex the optimal subdivision weights are chosen independently based on the dominant component of the quadratic shape at the vertex.

We demonstrate the benefits of the optimised subdivision weights over the Catmull-Clark weights by plotting the convergence of L_2 and energy norm errors. The successively refined meshes are obtained by subdivision using the Catmull-Clark weights. In all examples 4×4 Gauss quadrature points are used to evaluate the finite element integrals, which appears to be sufficiently accurate as shown in Figure 4.6a and Figure 4.6b. See also [60] for a systematic study on numerical integration of subdivision surfaces. For all examples in this section, the boundaries of the plates are simply supported. We tag boundary edges as creases and the four corner vertices as corners, which call for sharp subdivision rules [36]. In this setting, the Dirichlet boundary conditions $\boldsymbol{u} = \mathbf{0}$ on simply supported edges are straightforward to apply.

4.4.1 Uniform loading, symmetric unstructured mesh

As the first example, we compute the deformation of a simply supported square plate subjected to uniform transversal loading. The plate is discretised with the symmetric unstructured mesh shown in Figure 4.8a. Since the displacement field is usually not known prior to a finite element analysis, we use the optimal weights for cup-like shapes to solve the plate bending problem on a level $\ell = 2$ control mesh. Afterwards, for extraordinary vertices with valence $v = 5$ we decompose the local displacement field energetically to determine whether it is cup- or saddle-dominated and choose the optimal subdivision weights accordingly. For the considered uniform loading the shape decomposition shows that saddle dominates at all valence $v = 5$ vertices with cup-saddle ratio $R = 0.766$. Therefore, we choose the optimal weights for saddle-dominated shapes for all $v = 5$ vertices and study the convergence of the finite element solution using meshes from levels $\ell = 1$ to $\ell = 5$.

Figure 4.9 shows the level $\ell = 2$ control mesh and the deformed plate. The convergence of L_2 and energy norm errors are plotted in Figure 4.10. The optimised refinement weights reduce the L_2 norm error by more than 50% and the energy norm error by more than 45% in comparison to Catmull-Clark subdivision weights.

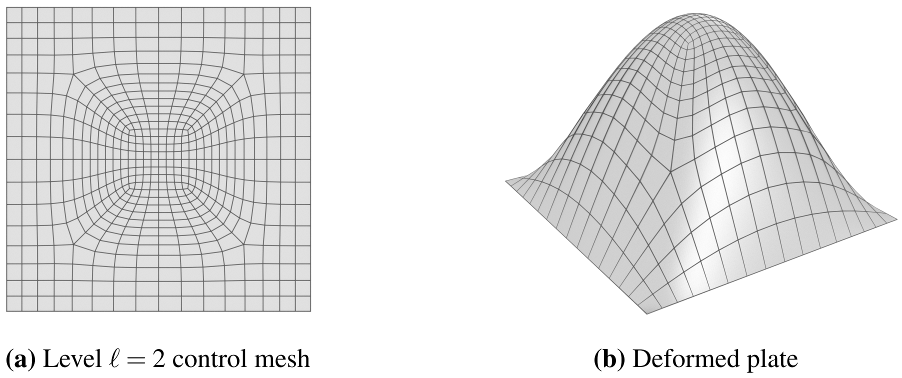

(a) Level $\ell = 2$ control mesh (b) Deformed plate

Figure 4.9: Control mesh with fourfold symmetry and the deformed plate under uniform loading. The shown control mesh is obtained by subdividing the symmetric coarse control mesh in Figure 4.8a twice using Catmull-Clark weights.

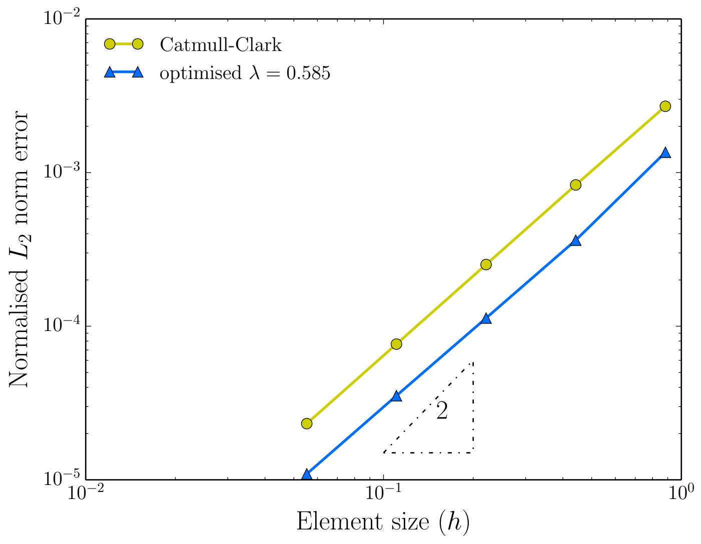
4.4.2 Sinusoidal loading, symmetric unstructured mesh

Next, we compute the deformation of a simply supported square plate discretised with the unstructured mesh shown in Figure 4.8a and subjected to sinusoidal loading. Compared with the first example, the only difference is that sinusoidal loading is applied instead of a uniform loading. After the first finite element analysis with the optimal weights for the cup-like geometry the local shape decomposition shows that the cup component dominates at all valence $v = 5$ vertices with cup-saddle ratio $R = 29.2$. Therefore, we choose the optimal weights for cup-dominated shapes for all $v = 5$ vertices and study the convergence of the finite element solution using meshes from levels $\ell = 1$ to $\ell = 5$.

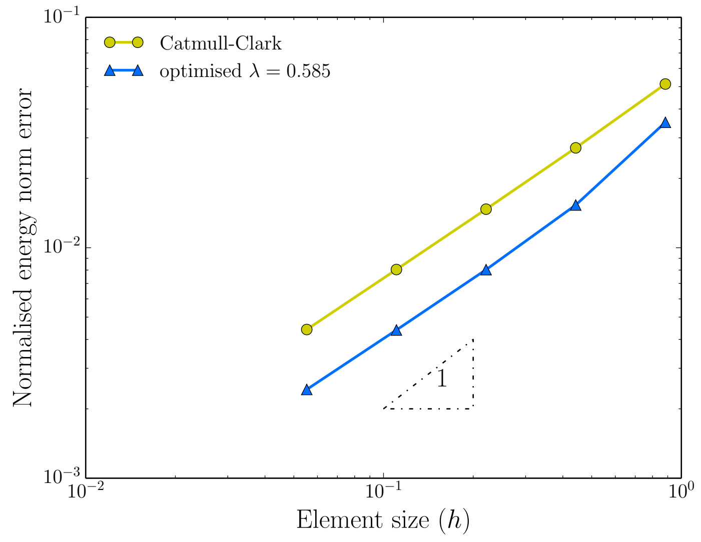
Figure 4.11 shows the level $\ell = 2$ control mesh and the deformed plate. The convergence of L_2 and energy norm errors are plotted in Figure 4.12. The optimised refinement weights reduce the L_2 norm error by more than 50% and the energy norm error by more than 20% in comparison to Catmull-Clark weights.

4.4.3 Sinusoidal loading, asymmetric unstructured mesh

In this last example, we compute the deformation of a simply supported square plate discretised with the asymmetric unstructured mesh shown in Figure 4.8b and subjected to



(a) L_2 norm error



(b) Energy norm error

Figure 4.10: Uniform loading with symmetric unstructured mesh. Saddle dominates ($R = 0.766$) at all valence $v = 5$ vertices. Optimisation reduces the L_2 norm error by more than 50% and energy norm error by more than 45%. See Table 4.1 for the values of the optimised weights corresponding to $\lambda = 0.585$.

sinusoidal loading. As in the first two examples, we obtain the first finite element solution using the optimal weights for cup-like shapes on a level $\ell = 3$ control mesh. Subsequently, for each extraordinary vertex with valence $v = 5$ the local displacement field is decomposed to determine whether it is cup or saddle dominated and the optimal weights are chosen

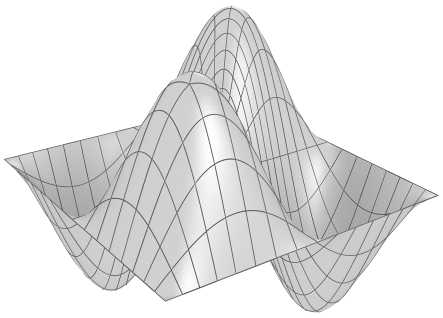


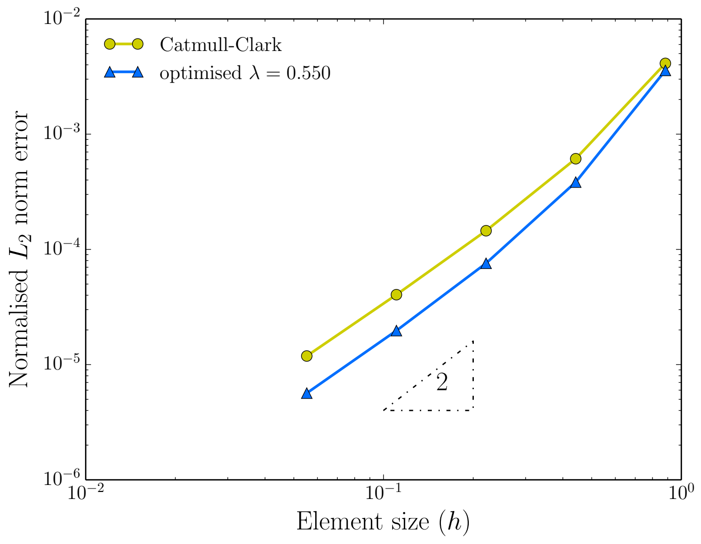
Figure 4.11: The deformed plate under sinusoidal loading on a level $\ell = 2$ symmetric control mesh.

accordingly. The local shape decomposition shows that the local solution at vertex P_6 is cup dominated with cup-saddle ratio $R = 85.8$ and at vertex P_{12} it is saddle dominated with $R = 0.0463$. We choose optimised cup weights for P_6 and saddle weights for P_{12} and study the convergence of the finite element solution using meshes from levels $\ell = 1$ to $\ell = 5$.

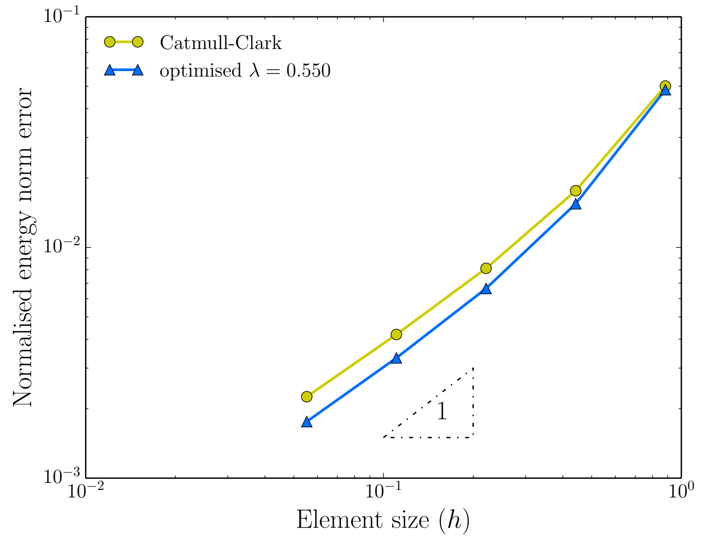
Figure 4.13 shows level $\ell = 3$ control mesh and the deformed plate. The convergence of L_2 and energy norm errors are plotted in Figure 4.14. The optimised refinement weights reduce both L_2 error and energy norm error by more than 50% in comparison to Catmull-Clark weights.

4.5 Conclusions

We have shown that significant reductions in discretisation errors in L_2 and energy norms can be achieved when subdivision weights around extraordinary vertices are optimised for finite element analysis. Although this was demonstrated for Catmull-Clark subdivision surfaces, a similar approach can be developed for other subdivision schemes as well. During finite element analysis the subdivision weights at each extraordinary vertex are chosen depending on whether the local solution has a more cup- or saddle-like shape. Two sets of weights, one for cup and the other for saddle, were derived which depend only on the valence of the extraordinary vertex. We only discussed valence $v = 5$ because it is, in addition to $v = 3$, one of the most occurring valences for quad meshes. The same implementation applies to extraordinary vertices with $v > 5$ with no further modification. For the case valence $v = 3$ we observed that the improvement is not significant. This is as to be expected given that the 1-neighbourhood of a valence $v = 3$ vertex shrinks faster than of other valences. In the optimisation process three of the subdivision weights, α, β and γ in the 1-neighbourhood of an extraordinary vertex were selected as degrees of freedom. By considering the 2-neighbourhood of a vertex it would have been possible to optimise



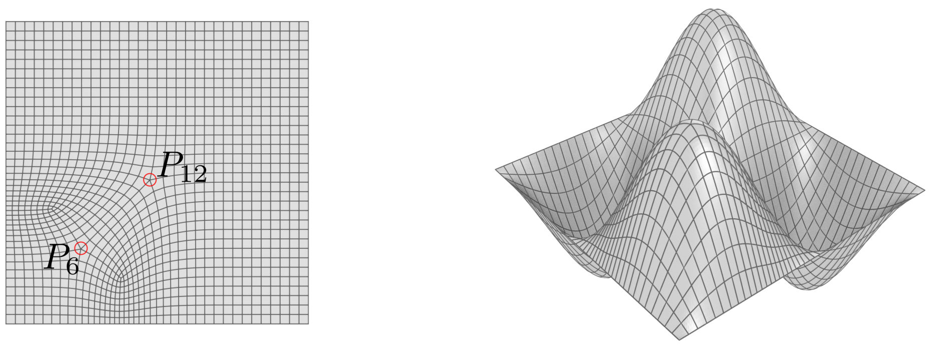
(a) L_2 norm error



(b) Energy norm error

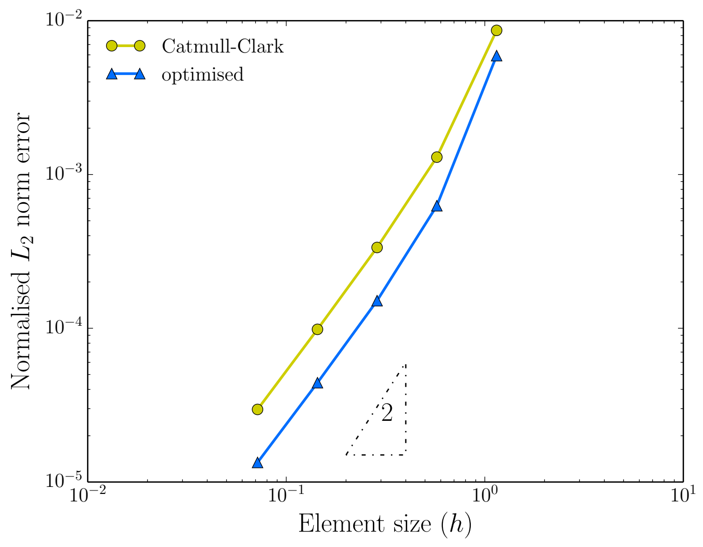
Figure 4.12: Sinusoidal loading with symmetric unstructured mesh. Cup dominates ($R = 29.2$) at all valence $v = 5$ vertices. Optimisation reduces the L_2 error by more than 50% and energy norm error by more than 20%. See Table 4.1 for the values of the optimised weights corresponding to $\lambda = 0.550$.

more than three subdivision weights. This may lead to even larger reductions in the errors although the considered optimisation problems become larger. Finally, subdivision surfaces are equally well suited for finite element analysis and modelling of geometries with arbitrary topology. With the derived optimised weights, geometric models created with subdivision surfaces in the new engineering design systems can be analysed much more efficiently.

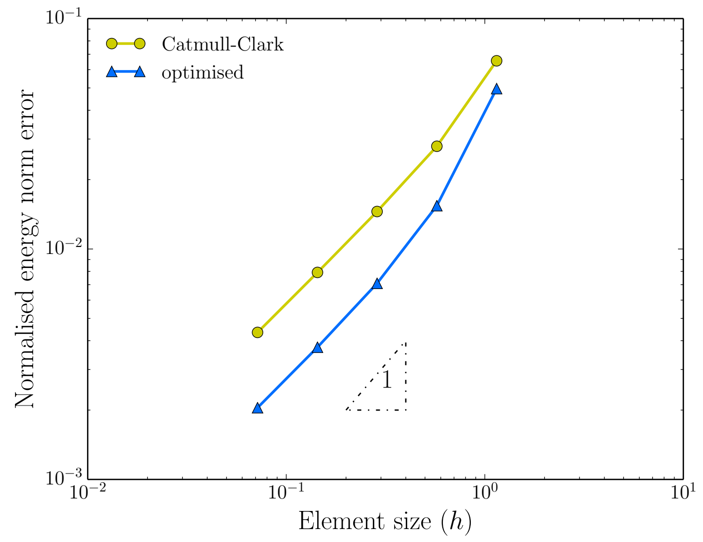


(a) Level-3 control mesh **(b)** Deformed plate

Figure 4.13: Asymmetric control mesh and the deformed plate under sinusoidal loading. The control mesh is obtained by subdividing the asymmetric coarse control mesh shown in Figure 4.8b three times with Catmull-Clark subdivision weights.



(a) L_2 norm error



(b) Energy norm error

Figure 4.14: Sinusoidal loading with asymmetric unstructured mesh. Cup dominates ($R = 85.8$) at vertex P_6 and saddle dominates ($R = 0.0463$) at vertex P_{12}. Therefore, $\lambda = 0.550$ is chosen for vertex P_6 and $\lambda = 0.585$ is chosen for vertex P_{12}. Tuning reduces both the L_2 error and energy norm error by more than 50%. See Table 4.1 for the values of the optimised weights.

Chapter 5

Manifold basis functions with sharp features

In geometric modelling numerous approaches have been developed to represent smooth surfaces with control meshes of arbitrary connectivity. These surface representation techniques can be applied and further developed to derive basis functions for isogeometric analysis on unstructured quadrilateral meshes with extraordinary vertices. Subdivision surfaces, as introduced in the previous chapter, have appealing properties for geometric modelling but lack optimal convergence in finite element analysis in the presence of extraordinary vertices. Alternatively, manifold-based surface construction is another neat technique to deal with extraordinary vertices. Manifold-based basis functions for isogeometric analysis were first introduced in [24] and the finite element simulations indicate an optimal convergence for Poisson problems and near optimal convergence for thin-shell problems even with extraordinary vertices. In this chapter, we present an extension of the previous work in [24] to derive novel manifold-based basis functions with arbitrary prescribed smoothness to represent geometries and solution fields with sharp features, such as creased edges and corners.

Manifold-based surface construction is a well known technique in geometric modelling. The main concept is to construct a smooth surface by blending together overlapping charts (or, patches) as in the differential geometry description of manifolds. Following [24], we derive basis functions on unstructured quadrilateral meshes by combining manifold techniques with conformal parametrisations and the partition-of-unity method. Each chart on the manifold consists of several elements and has a corresponding planar chart with a smooth one-to-one mapping onto the manifold. On the collection of conformally parametrised planar charts the partition-of-unity method is used for approximation. The smooth partition-of-unity, or blending, functions are assembled from tensor-product B-spline segments defined over a unit square. In contrast to [24], in the present work polynomials with prescribed degree and continuity are used as local approximants on each chart. Sharp features are represented with suitably chosen C^0-continuous local polynomials. As will be demonstrated, the new manifold basis functions have to be carefully constructed in order to be suitable for both geometric modelling and analysis. This is achieved by considering several affine and quasi-conformal mappings depending on the local connectivity of the mesh and arrangement of sharp features. The resulting basis functions have arbitrary prescribed smoothness and approximation order.

5.1 Introduction

Manifold-based basis functions for isogeometric analysis were first introduced in [24]. The $C^{k\geq 1}$ continuous manifold basis functions are defined on unstructured quadrilateral meshes, hence can represent parts with arbitrary topology, and give optimal convergence rates in finite element computations. Manifold basis functions resemble spline basis functions in the sense that each basis function has local support and has one corresponding vertex. Manifold constructions have a rich history in geometric modelling [19–23] and our specific approach is motivated by [21]. The basis functions are obtained by blending together polynomial approximants defined on disparate planar chart domains using the partition of unity method. Each of the chart domains is parametrised with a quasi-conformal map which is easy to evaluate and to invert. In this work, we extend the manifold basis functions to represent surfaces with C^0 continuous sharp features, like creases and corners, see Figure 5.1. The crease edges are tagged as such on the control mesh by the user of CAD systems. In geometric modelling with manifolds crease edges have been previously considered in [61–63]. As the treatment of domain boundaries has similarities to C^0 creases, the new basis functions provide also better control over boundaries. The new manifold basis functions are constructed by choosing the polynomial approximants on the charts differently. The new approximants consist out of several polynomial pieces which are C^0 continuously connected across the crease edges. To realise this for an arbitrary number and arrangement of creases it is necessary to modify the chart domains. The new chart domains are chosen such that they are partitioned into equiangular sectors by the creases. In order to parametrise these new chart domains a new quasi-conformal mapping is proposed.

The outline of this chapter is as follows. In Section 5.2 the manifold basis functions introduced in [24] are reviewed. Different from [21] we emphasise the computation of the basis functions with a focus on their implementation in isoparametric finite element software. The treatment of sharp features is discussed in Section 5.3. Depending on the arrangement of crease edges we distinguish between rotationally symmetric and non-symmetric charts. Both require a different quasi-conformal map for parametrisation. In addition, the case of charts leading to non-convex sectors on the characteristic map requires special treatment. Section 5.4 introduces finite element analysis of thin shells with normal control. In Section 5.5 the new manifold basis functions are applied to several Bernoulli beams and linear and nonlinear Kirchhoff-Love shell problems. The numerical convergence with decreasing element size in L^2 norm is demonstrated.

5.2 Review of manifold basis functions

In the following we briefly review the construction of manifold basis functions with a focus on their application in finite elements. The discussion is limited to the manifold basis functions introduced in [24]. For the sake of brevity the underlying fundamental manifold concepts from differential geometry and the partition of unity interpolation are only mentioned in passing.

5.2.1 Univariate basis functions

It is instructive to first review the derivation of the univariate manifold basis functions. We consider the polygon in Figure 5.2 representing a finite element mesh consisting of vertices $\boldsymbol{x}_J$ and elements between consecutive vertices. Our aim is to derive the basis functions for one single element highlighted in the shown mesh. Similar to isoparametric finite elements we define a reference element $\square := [0,1] \ni \eta$ that serves as an integration element for evaluating the finite element integrals. In the manifold approach the basis functions $N_J(\eta)$ are obtained by smoothly blending polynomials defined over several overlapping charts, or patches as

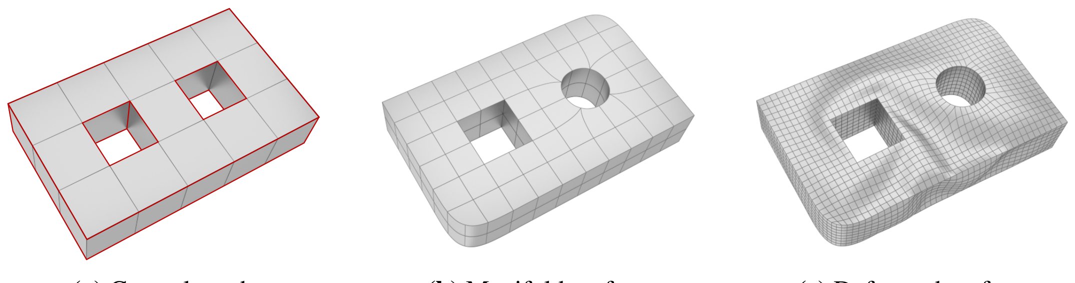

(a) Control mesh **(b)** Manifold surface **(c)** Deformed surface

Figure 5.1: Genus 2 surface with creased edges and sharp corners. On the control mesh (a) the edges to be creased are marked in red. In the constructed manifold surface (b) the creases are faithfully reproduced by selectively reducing the smoothness of the underlying basis functions.

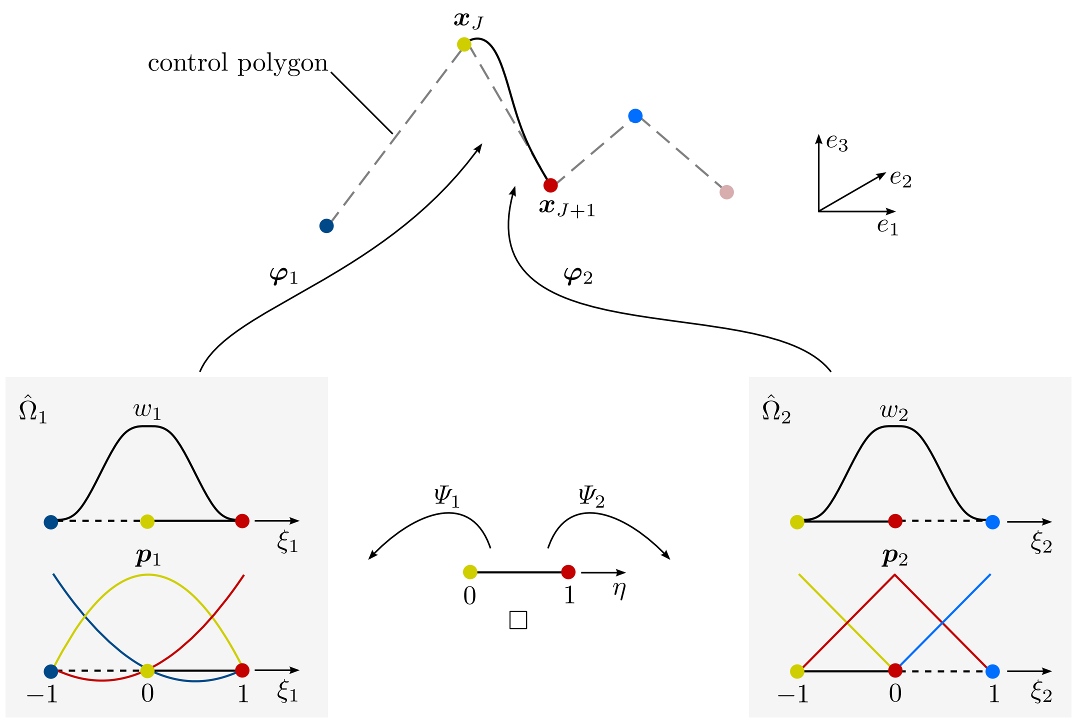


Figure 5.2: Construction of univariate basis functions by blending chart-wise defined Lagrange basis functions $\boldsymbol{p}_j$ using smooth weight functions w_j and the approximation of a control mesh.

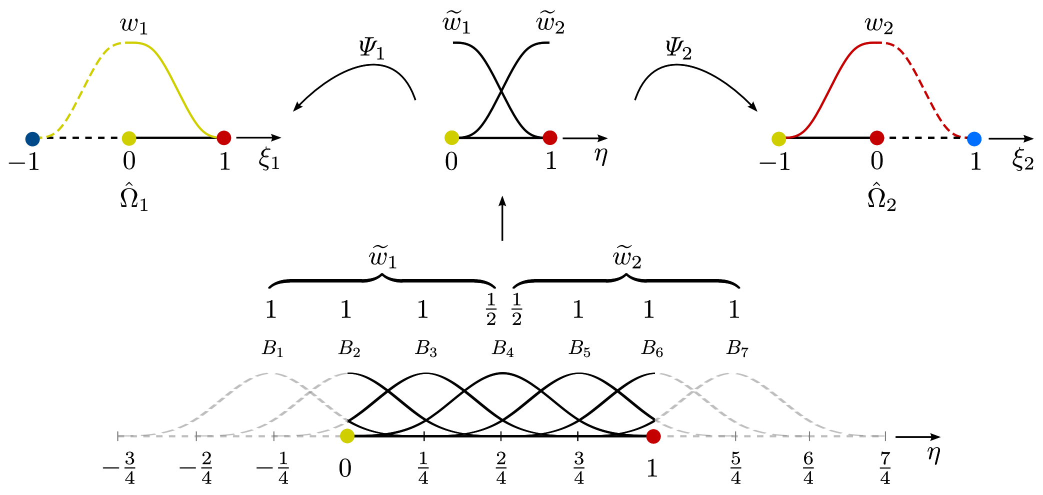
they were called in [24][1]. In the following we consider chart domains $\hat{\Omega}_j := [-1,1] \ni \xi_j$ that correspond to a vertex and the union of the two elements in its 1-neighbourhood. A 1-neighbourhood of a vertex is defined as the union of elements that contain the vertex. The n-neighbourhood is defined recursively as the union of all 1-neighbourhoods of the $(n-1)$-neighbourhood vertices. Hence, each chart contains $2n$ elements and, in turn, the image of each element is present on $2n$ charts. Using 1-neighbourhoods each reference element $\square$ maps to two elements in the domains $\hat{\Omega}_1$ and $\hat{\Omega}_2$, and there are four unique vertices in the union of the two charts. We first define the maps

$$\begin{aligned} \Psi_1 &: \eta \in [0,1] \mapsto \xi_1 \in [0,1] \\ \Psi_2 &: \eta \in [0,1] \mapsto \xi_2 \in [-1,0] \end{aligned} \tag{5.1}$$

from the reference element to the corresponding elements in the two charts, see Figure 5.2. Subsequently, on each chart $\hat{\Omega}_j$ a polynomial approximant

$$f_j(\xi_j) = \boldsymbol{p}_j^{\mathsf{T}}(\xi_j)\boldsymbol{\alpha}_j \tag{5.2}$$

[1]We refrain in this thesis from using the term *patches* because in computer-aided design literature patches denote what are the *elements* in computational mechanics.



w_1
Ψ_1
$\widetilde{w}_1$
$\widetilde{w}_2$
Ψ_2
w_2
-1
0
1
ξ_1
$\hat{\Omega}_1$
0
1
η
-1
0
1
ξ_2
$\hat{\Omega}_2$
$\widetilde{w}_1$
$\widetilde{w}_2$
1 1 1 $\frac{1}{2}$ $\frac{1}{2}$ 1 1 1
B_1 B_2 B_3 B_4 B_5 B_6 B_7
$-\frac{3}{4}$ $-\frac{2}{4}$ $-\frac{1}{4}$ 0 $\frac{1}{4}$ $\frac{2}{4}$ $\frac{3}{4}$ 1 $\frac{5}{4}$ $\frac{6}{4}$ $\frac{7}{4}$ η

Figure 5.3: Construction of smooth weight functions w_j using cubic B-spline basis defined on a uniform parameter space with knot interval length $1/4$. B_j denotes j-th cubic B-spline basis function $B_j(\eta)$ that is defined on a twice uniformly bisected mesh and non-zero in the reference element $\eta \in [0,1]$. See also (5.4) for a closed form expression of the weight functions $w_j(\xi_j)$.

is chosen, where $\boldsymbol{p}_j(\xi_j)$ is a vector containing a polynomial basis and $\boldsymbol{\alpha}_j$ contains its coefficients. It is possible to choose for $\boldsymbol{p}_j(\xi_j)$ any polynomial basis, like Lagrange, Bernstein, Chebyshev or any other. In the example in Figure 5.2 a Lagrange basis is used. The chosen basis on the chart $\hat{\Omega}_1$ is quadratic, whereas on the chart $\hat{\Omega}_2$ it is linear and only C^0 continuous. In addition to the polynomial basis we require on each chart a smooth weight function $w_j(\xi_j)$ that must satisfy the partition of unity property

$$w_1(\xi_1) + w_2(\xi_2) = 1 \quad \forall \eta \in [0,1], \quad \text{with} \quad \xi_j = \Psi_j(\eta). \tag{5.3}$$

Weight functions $w_j(\xi_j)$ can be constructed from B-spline basis. For example, Figure 5.3 illustrates how to obtain C^2 continuous weight functions by combining cubic B-splines, i.e.,

$$w_j(\xi_j) = \widetilde{w}_j(\eta) \quad \text{with} \quad \eta = \Psi_j^{-1}(\xi_j), \tag{5.4a}$$

$$\widetilde{w}_1(\eta) = B_1(\eta) + B_2(\eta) + B_3(\eta) + 0.5B_4(\eta), \tag{5.4b}$$

$$\widetilde{w}_2(\eta) = 0.5B_4(\eta) + B_5(\eta) + B_6(\eta) + B_7(\eta). \tag{5.4c}$$

It is straightforward to show that the weight functions satisfy (5.3) given that a complete B-spline basis sums up to one. It is worth emphasising that the proposed weight functions are polynomial in contrast to the rational ones originally introduced in [24], see Appendix A. Note that the construction defined in (5.4) can be easily extended to the bivariate case using tensor product.

Finally, the approximant over the reference element is obtained by blending the approximants over the two charts

$$f(\eta) = \sum_{j=1}^{2} w_j(\xi_j) \boldsymbol{p}_j^{\mathsf{T}}(\xi_j) \boldsymbol{\alpha}_j \quad \text{with} \quad \xi_j = \Psi_j(\eta) \,. \tag{5.5}$$

Due to the choice of the Lagrange basis for $\boldsymbol{p}_j(\xi_j)$ the coefficients $\boldsymbol{\alpha}_j$ can evidently be interpreted as vertex coefficients or degrees of freedom. The coefficients $\boldsymbol{\alpha}_j$ are obtained from coefficient array $\boldsymbol{f}$ using gather matrices $\boldsymbol{P}_j$ filled with ones and zeros,

$$\boldsymbol{\alpha}_j = \boldsymbol{P}_j \boldsymbol{f} \,. \tag{5.6}$$

This introduced in (5.5) yields the array of manifold-based isogeometric analysis basis functions $\boldsymbol{N}(\eta)$ for each element

$$f(\eta) = \sum_{j=1}^{2} \left(w_j(\xi_j) \boldsymbol{p}_j^{\mathsf{T}}(\xi_j) \boldsymbol{P}_j \right) \boldsymbol{f} = \boldsymbol{N}^{\mathsf{T}}(\eta) \boldsymbol{f} \quad \text{with} \quad \xi_j = \Psi_j(\eta) \,. \tag{5.7}$$

The four basis functions in $\boldsymbol{N}(\eta)$ correspond to the four unique vertices in the union of the two overlapping charts $\hat{\Omega}_1$ and $\hat{\Omega}_2$, see Figure 5.2.

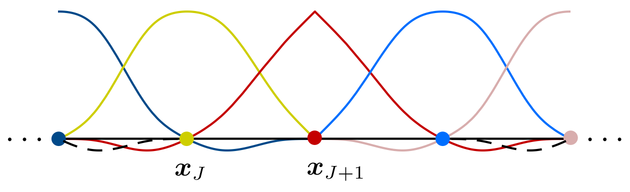
To investigate the smoothness of the basis functions it is necessary to consider the derivatives $d\boldsymbol{N}/d\eta$. Hence, the smoothness of the basis functions depends on the weight functions $w_j(\xi_j)$, the approximants $\boldsymbol{p}_j(\xi_j)$ and the mappings $\Psi_j(\eta)$. To obtain C^k continuous basis functions each of the terms has to be k-times differentiable and, in addition, the function value and derivatives up to k-th order of the weight functions have to be zero at the chart boundaries, i.e.,

$$\frac{d^l w_j(-1)}{d\xi_j^l} = \frac{d^l w_j(1)}{d\xi_j^l} = 0 \quad \forall l \le k \,. \tag{5.8}$$

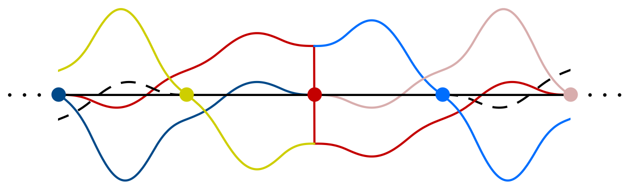
Note that the mappings must also have continuous derivatives at the centre of the chart domain, i.e.

$$\frac{d^l \Psi_1(0)}{d\eta^l} = \frac{d^l \Psi_2(1)}{d\eta^l} \quad \forall l \le k \,. \tag{5.9}$$

The smoothness of the basis can be pointwise reduced by selecting suitable approximants $\boldsymbol{p}_j(\xi_j)$, see Figure 5.2 with linear basis functions on one of the charts. The resulting basis functions and their first derivatives are plotted in Figure 5.4. The smoothness of the



(a) Basis functions



(b) First derivatives

Figure 5.4: Univariate manifold basis functions and their derivatives. Five basis functions (solid) are non-zero over the two elements of the centre chart.

basis is reduced to C^0 at the chosen vertex. This may also have an effect on the number of non-zero basis functions $\boldsymbol{N}(\eta)$. For instance, in the element considered with the linear basis functions there are only three non-zero basis functions. In elements with a quadratic polynomial on each chart there are four smooth non-zero basis functions.

At last, the generated manifold basis functions can be used to represent the curve described by the initial polygon with

$$\boldsymbol{x}(\eta) = \sum_{J=1}^{4} N_J(\eta)\boldsymbol{x}_J \,. \tag{5.10}$$

5.2.2 Bivariate basis functions

The bivariate manifold basis functions provide smooth approximants even on unstructured meshes. We consider the construction of the manifold basis functions for one single element in the quadrilateral finite element mesh shown in Figure 5.5.

The reference element is now defined as a unit square $\square := [0,1] \times [0,1] \ni \boldsymbol{\eta} = (\eta^1, \eta^2)$. Similar to the univariate case each vertex $\boldsymbol{x}_J$ has an associated chart domain $\hat{\Omega}_j \ni \boldsymbol{\xi}_j = (\xi_j^1, \xi_j^2)$, which consists of all the elements in the 1-neighbourhood of the vertex. The number of elements connected to a vertex is called its valence and denoted with v. In quadrilateral meshes, the interior vertices with $v \neq 4$ are the *extraordinary vertices*. The basis functions $N_J(\boldsymbol{\eta})$ are obtained by smoothly blending polynomials defined over each of the four overlapping charts, see Figure 5.6. When the domain of each chart is defined as the 1-neighbourhood of a vertex, the number of charts relevant for the considered element is always four because each quadrilateral has four vertices.

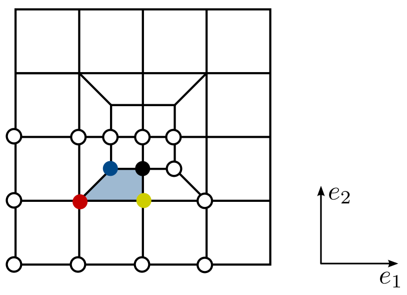


Figure 5.5: An unstructured mesh with extraordinary vertices of valence $v \in 3,5$. The shaded element is overlapped by four charts and the union of the four charts has 16 unique vertices in total.

The bivariate approximant over the reference element $\square$ is obtained by blending the four chart contributions

$$f(\boldsymbol{\eta}) = \sum_{j=1}^{4} w_j(\boldsymbol{\xi}_j) f_j(\boldsymbol{\xi}_j) = \sum_{j=1}^{4} w_j(\boldsymbol{\xi}_j) \boldsymbol{p}_j^{\mathsf{T}}(\boldsymbol{\xi}_j) \boldsymbol{\alpha}_j \quad \text{with} \quad \boldsymbol{\xi}_j = \boldsymbol{\Psi}_j(\boldsymbol{\eta}) , \tag{5.11}$$

where $w_j(\boldsymbol{\xi}_j)$ and $\boldsymbol{p}_j(\boldsymbol{\xi}_j)$ are the partition of unity weight function and polynomial basis of the j-th chart, respectively. In contrast to the univariate case the choice of the map $\boldsymbol{\Psi}_j(\eta)$ requires special care and will be discussed further below. The coefficients of all the unique vertices in the four charts are collected in the array $\boldsymbol{f}$. The number of unique vertices is not fixed and depends on the valences v_j of the four vertices of the element. The approximant (5.11) expressed in dependence of the vertex coefficients $\boldsymbol{f}$ yields

$$f(\boldsymbol{\eta}) = \sum_{j=1}^{4} \left(w_j(\boldsymbol{\xi}_j) \boldsymbol{p}_j^{\mathsf{T}}(\boldsymbol{\xi}_j) \boldsymbol{A}_j \boldsymbol{P}_j \right) \boldsymbol{f} \quad \text{with} \quad \boldsymbol{\xi}_j = \boldsymbol{\Psi}_j(\boldsymbol{\eta}) , \tag{5.12}$$

where we used $\boldsymbol{\alpha}_j = \boldsymbol{A}_j \boldsymbol{P}_j \boldsymbol{f}$ on each chart $\hat{\Omega}_j$. $\boldsymbol{P}_j$ is the gather matrix. The least-squares projection matrix $\boldsymbol{A}_j$ is required because the number of polynomial coefficients can be less than the number of vertices. It is worth mentioning that the projection matrix $\boldsymbol{A}_j$ depends only on the valence v_j of the vertex and can be precomputed and tabulated. Finally, the array of basis functions for isogeometric analysis is defined with

$$\boldsymbol{N}^{\mathsf{T}}(\boldsymbol{\eta}) = \sum_{j=1}^{4} w_j(\boldsymbol{\xi}_j) \boldsymbol{p}_j^{\mathsf{T}}(\boldsymbol{\xi}_j) \boldsymbol{A}_j \boldsymbol{P}_j \quad \text{with} \quad \boldsymbol{\xi}_j = \boldsymbol{\Psi}_j(\boldsymbol{\eta}) . \tag{5.13}$$

As mentioned earlier, the smoothness of the basis functions depends on the smoothness of the weight functions $w_j(\boldsymbol{\xi}_j)$, the approximants $\boldsymbol{p}_j(\boldsymbol{\xi}_j)$ and the mappings $\boldsymbol{\Psi}_j(\boldsymbol{\eta})$. The number of mappings contributing to a specific chart is the same as the number of elements on it. The set of all mappings contributing to a chart is referred to as its parametrisation in

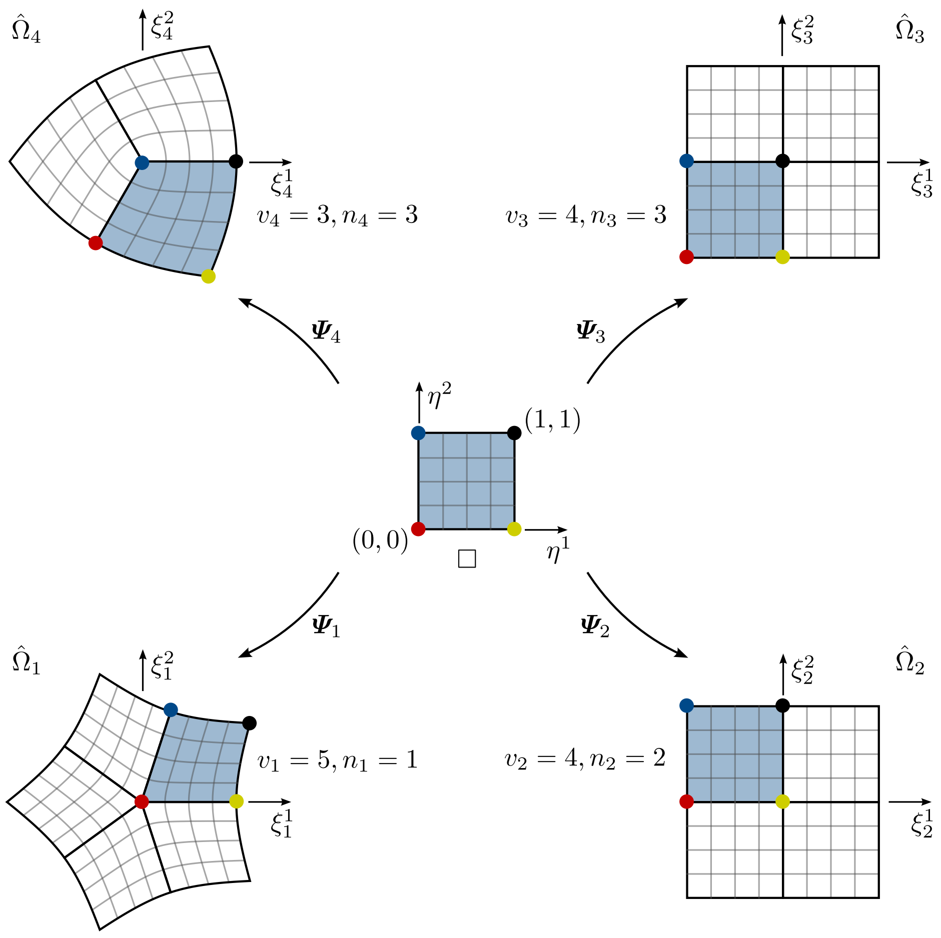


Figure 5.6: The reference element $\square$ (center) and its four overlapping chart domain $\hat{\Omega}_j$ with $j \in \{1,2,3,4\}$. The reference element is mapped with $\boldsymbol{\Psi}_j$ to the blue shaded elements in the four charts $\hat{\Omega}_j$. Notice that the parameter lines in the charts with $\eta^1 = \text{const.}$ or $\eta^2 = \text{const.}$ are always orthogonal to the spoke edges which guarantees that the derivatives of $\hat{\Omega}_j$ are continuous over the entire chart.

computer-aided design and graphics literature. In our approach conformal maps are critical for the smooth parametrisation of charts and for the composition of smooth weight functions. As illustrated in Figure 5.7, the mapping $\boldsymbol{\Psi}_j \colon [0,1] \times [0,1] \ni \boldsymbol{\eta} \mapsto \boldsymbol{\xi}_j$ is composed of an auxilliary linear map $\boldsymbol{\Psi}_{\boldsymbol{l}}$ and a quasi-conformal map $\boldsymbol{\Psi}_{\boldsymbol{c}}$, that is

$$\boldsymbol{\Psi}_j = \boldsymbol{\Psi}_{\boldsymbol{c}} \circ \boldsymbol{\Psi}_{\boldsymbol{l}}\,, \quad \text{with} \quad j \in \{1,2,3,4\}\,. \tag{5.14}$$

To write the conformal map more succinctly, first the coordinates $\boldsymbol{\eta} = (\eta^1, \eta^2)$ of points in the reference element are expressed as a complex number

$$\begin{aligned} z = \eta^1 + i\eta^2 = |z|(\cos\phi + i\sin\phi) = |z|e^{i\phi} \quad \text{with} \\ |z| = \sqrt{(\eta^1)^2 + (\eta^2)^2} \quad \text{and} \quad \phi = \arctan(\eta^2/\eta^1)\,, \end{aligned} \tag{5.15}$$

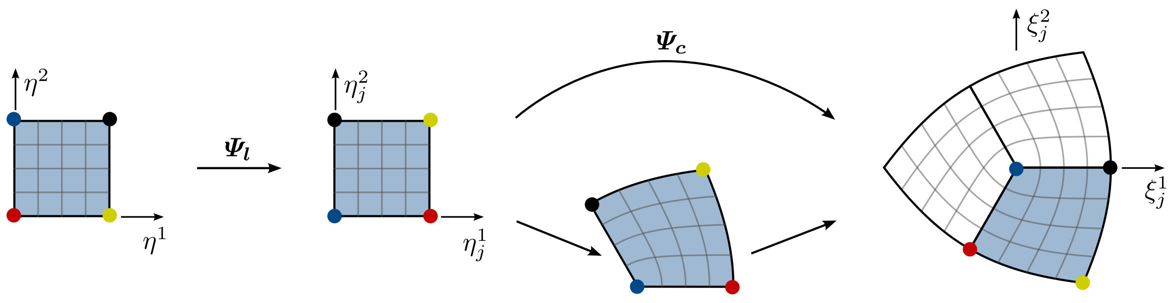


Figure 5.7: The map from the reference element to a chart with a valence $v_j = 3$ centre vertex in Figure 5.6. Substitute the value $j = 4$, $v_j = 3$, $n_j = 3$ into (5.16) and (5.17) to obtain the expression of $\boldsymbol{\Psi_l}$ and $\boldsymbol{\Psi_c}$ in this example.

where $|z|$ is the radius and ϕ the angle in the complex plane. The auxiliary linear map $\boldsymbol{\Psi_l}$ is only responsible for translating and rotating the reference element and is given by

$$\boldsymbol{\Psi_l}(z;j) = (z - z_j)e^{-i\pi(j-1)/2}, \tag{5.16}$$

where $(z - z_j)$ is a translation and $e^{-i\pi(j-1)/2}$ a (rigid body) rotation. z_j is the coordinate of the j-th vertex in the reference element such that $[z_1, z_2, z_3, z_4]^\mathsf{T} = [0 + 0i, 1 + 0i, 1 + 1i, 0 + 1i]^\mathsf{T}$. A more detailed illustration of geometry meanings of complex number operations is given in Appendix B. The quasi-conformal map $\boldsymbol{\Psi_c}$ maps the reference element into a wedge-shaped domain and is chosen as

$$\boldsymbol{\Psi_c}(z; v_j, n_j) = \frac{|z|^\beta}{|z|^{4/v_j}} z^{4/v_j} e^{i2\pi(n_j-1)/v_j} = |z|^\beta e^{i\phi 4/v_j} e^{i2\pi(n_j-1)/v_j}, \tag{5.17}$$

where β is a free parameter, v_j denotes the valence of the centre vertex and n_j is the face number of the considered element in the j-th chart. According to the first expression in (5.17), the map $\boldsymbol{\Psi_c}$ is composed of a standard conformal map z^{4/v_j} followed by a scaling of the radius with $|z|^{\beta - 4/v_j}$ and a rotation $e^{i2\pi(n_j-1)/v_j}$. When $\beta = 4/v_j$ the scaling term drops and the map $\boldsymbol{\Psi_c}$ is a conformal map, as used in [24]. The second expression in (5.17) shows that the radius $|z|$ will remain constant when $\beta = 1$. Hence, in this work we choose $\beta = 1$ to obtain a more uniform mapping $\boldsymbol{\Psi}_j$ with a more uniform Jacobian. See Appendix B for an illustration of the influence of the parameter β on the conformal/quasi-conformal images.

With the mapping $\boldsymbol{\Psi}_j$ at hand we can evaluate the manifold basis functions defined in (5.13). Given any integration point (η^1, η^2), we first find its image (ξ_j^1, ξ_j^2) in each of the four overlapping charts. Then it is straightforward to evaluate the weight functions $w_j(\boldsymbol{\xi}_j)$ and local polynomials $\boldsymbol{p}_j(\boldsymbol{\xi}_j)$. The weight functions $w_j(\boldsymbol{\xi}_j)$ are defined as the tensor-product of univariate weight functions given in (5.4).

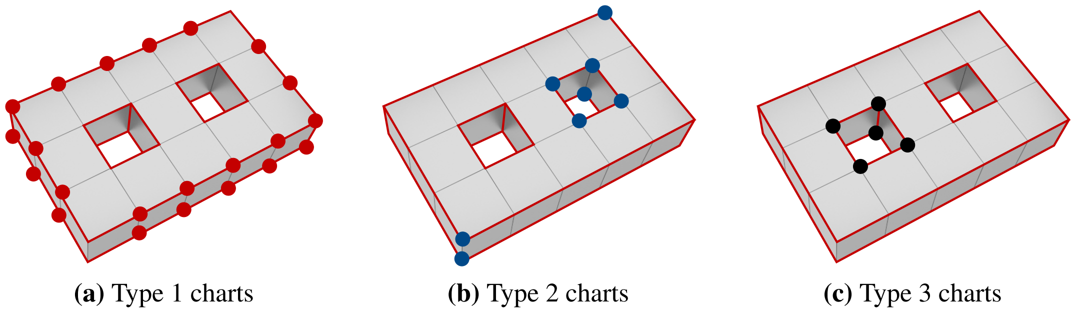

(**a**) Type 1 charts (**b**) Type 2 charts (**c**) Type 3 charts

Figure 5.8: Genus 2 surface with creased edges and sharp corners. Three different types of possible crease charts with the centre vertex marked in colour: Type 1 (red), Type 2 (blue) and Type 3 (black). See Figure 5.1 for the manifold surface represented by this control mesh and notice the differences between the two holes (one is square and the other is circular).

5.3 Creased bivariate manifold basis functions

In this section, we classify crease charts into three different types, depending on the local arrangement of crease edges. Figure 5.8 illustrates all three chart types, which call for different parametrisations discussed in the following.

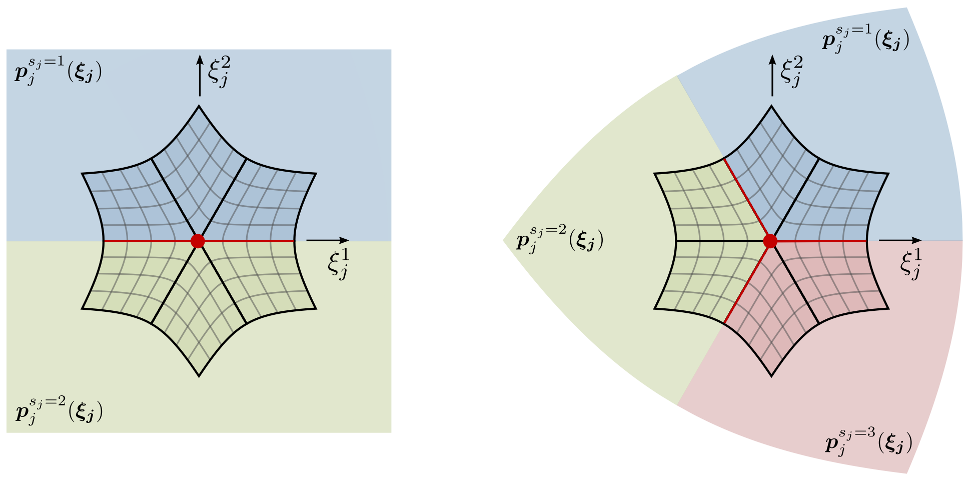
5.3.1 Type 1: Rotationally symmetric charts

The crease chart Type 1 shown in Figure 5.9 has a rotationally symmetric connectivity with respect to the arrangement of the creased edges. There can be $k \geq 2$ spoke edges which are tagged as crease edges[2]. Specifically, the number of elements in each of the sectors is the same. The basic idea is to define the local approximant $f_j(\xi_j^1, \xi_j^2)$ in each sector independently while ensuring the C^0 continuity across each crease edge shared by two sectors. Figure 5.9a shows one example of a Type 1 chart with $k_j = 2$ creases. In this case, the local approximant is chosen as

$$f_j(\xi_j^1, \xi_j^2) = \begin{cases} f_j^{s_j=1}(\xi_j^1, \xi_j^2) & \text{if } \xi_j^2 \geq 0 \\ f_j^{s_j=2}(\xi_j^1, \xi_j^2) & \text{if } \xi_j^2 < 0 \end{cases} \quad \text{with} \quad f_j^{s_j=1}(\xi_j^1, 0) = f_j^{s_j=2}(\xi_j^1, 0)\,. \tag{5.18}$$

It is straightforward to match the approximation $f_j^{s_j=1}$ in upper part and $f_j^{s_j=2}$ in lower part at the horizontal axis. However, in the case of $k_j \geq 3$ creases which are not necessarily axis-aligned, further effort is needed to guarantee C^0 continuity at all creases. One option is to choose a polynomial basis $\boldsymbol{b}(\boldsymbol{\theta})$ defined in a square domain such that the boundary curve

[2]In this section, k is used for the number of creased spoke edges, which is different from what means regularity in G^k and C^k continuity.



(a) $v_j = 6$ and $k_j = 2$ **(b)** $v_j = 6$ and $k_j = 3$

Figure 5.9: Examples of Type 1 crease chart domains. The k_j tagged creased edges divide the chart into k_j sectors shaded in different colours. Notice the rotational symmetry of the sectors.

only depends on control points on the boundary. Then map the basis from a square to each sector to obtain the sector-wise basis $\boldsymbol{p}_j^{s_j}(\boldsymbol{\xi}_j)$. In this case, the local approximants $f_j^{s_j}(\boldsymbol{\xi}_j)$ in two adjacent sectors meet C^0 continuously because they share the same basis and the same control points on the common sector boundary. Therefore, for Type 1 crease charts with $k_j \geq 3$ we use the quasi-conformal map introduced in Section 5.2.2, i.e.,

$$\boldsymbol{p}_j^{s_j}(\boldsymbol{\xi}_j) = \boldsymbol{b}(\boldsymbol{\theta}) \quad \text{with} \quad \boldsymbol{\theta} = \boldsymbol{\Psi}_{\boldsymbol{c}}^{-1}(\boldsymbol{\xi}_j; k_j, s_j)\,, \tag{5.19}$$

where k_j is the number of creases in the considered chart and $s_j \in \{1, 2, \cdots, k_j\}$ denotes the sector number. The expression of quasi-conformal map $\boldsymbol{\Psi}_{\boldsymbol{c}}$ is given in (5.17). As for the polynomial basis $\boldsymbol{b}$ defined in a square reference domain, various choices are available, e.g., Lagrange or Bernstein polynomials. Obviously, the polynomial degree should be chosen such that the number of degrees of freedom is less than or equal to the number of vertices in each sector. More supporting nodes can be added in each chart if a higher degree basis is needed. In our implementation, middle-face and middle-edge supporting nodes are introduced, resulting in $6v + 1$ control points in each chart of valence v.

5.3.2 Type 2: Non-symmetric charts

The crease chart Type 2 has no rotationally symmetric connectivity with respect to the arrangement of the creased edges, see Figures 5.10 and 5.11 . That is, the number of elements in each of the sectors is not the same. In this case the map $\boldsymbol{\Psi}_j \colon [0,1] \times [0,1] \ni \boldsymbol{\eta} \mapsto \boldsymbol{\xi}_j$ from

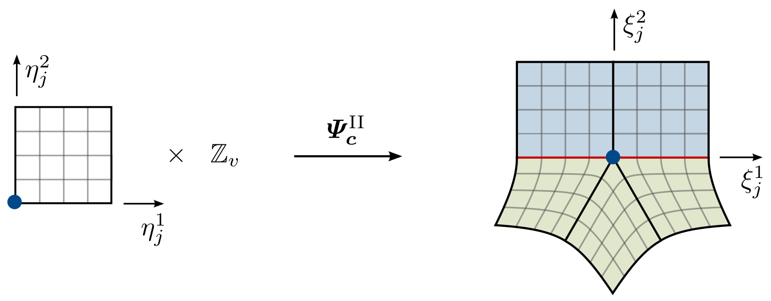


Figure 5.10: Example of Type 2 crease chart domain. The centre vertex has valence $v_j = 5$ and $k_j = 2$ crease edges.

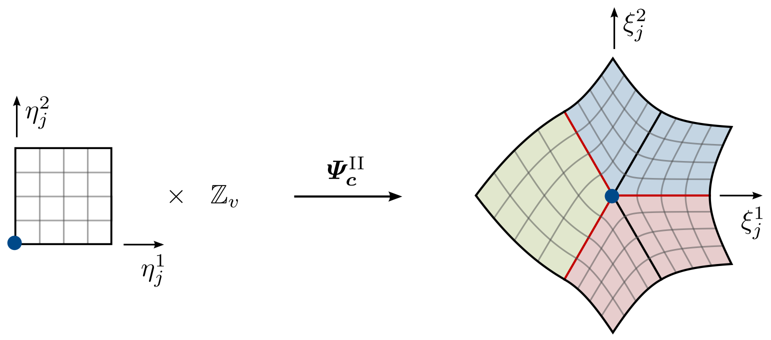


Figure 5.11: Example of Type 2 crease chart domain. The centre vertex has valence $v_j = 5$ and $k_j = 3$ crease edges.

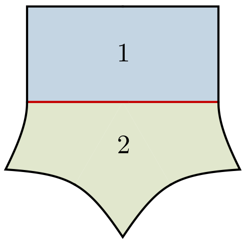
the reference element to Type 2 crease chart is different from what is used for smooth chart and Type 1 crease chart. Similar to (5.14), the map $\boldsymbol{\Psi}_j^{\mathrm{II}}$ for Type 2 crease chart is defined as

$$\boldsymbol{\Psi}_j^{\mathrm{II}} = \boldsymbol{\Psi}_{\boldsymbol{c}}^{\mathrm{II}} \circ \boldsymbol{\Psi}_{\boldsymbol{l}}\,, \quad \text{with} \quad j \in \{1,2,3,4\}\,. \tag{5.20}$$

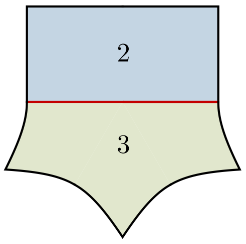
Essentially, the map $\boldsymbol{\Psi}_{\boldsymbol{c}}$ in (5.14) is replaced with a modified map $\boldsymbol{\Psi}_{\boldsymbol{c}}^{\mathrm{II}}$, defined as

$$\boldsymbol{\Psi}_{\boldsymbol{c}}^{\mathrm{II}}(\boldsymbol{\eta}_j; k_j, s_j, l_s, m_s) = \boldsymbol{\Psi}_{\boldsymbol{c}}(\boldsymbol{\eta}_j; k_j l_s, (s_j - 1) l_s + m_s)\,, \tag{5.21}$$

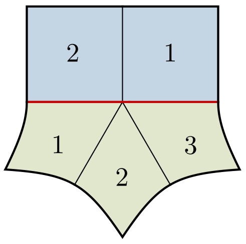
where k_j is the number of creases in the chart, $s_j \in \{1, 2, \cdots, k_j\}$ is the sector number, l_s is the number of elements in the sector and $m_s \in \{1, 2, \cdots, l_s\}$ is the local element number in a sector. $\boldsymbol{\Psi}_{\boldsymbol{c}}$ is the quasi-conformal map defined in (5.17). Figure 5.12 clarifies what these variables mean. After the Type 2 crease chart is mapped to an equiangular domain, local approximants are defined in each sector and matched C^0 continuously along the crease edges, using the same approach as discussed in Section 5.3.1.



(a) Sector number s_j



(b) Number of elements in a sector l_s



(c) Local element number m_s

Figure 5.12: Illustration of the numbering used to define the map $\boldsymbol{\Psi}_{\boldsymbol{c}}^{\mathrm{II}}$ in (5.21), taking the example of the Type 2 crease chart shown in Figure 5.10. The number of creases $k_j = 2$ takes the same value for every element.

5.3.3 Type 3: Charts with concave corners

Convex and concave corners, while being topologically equivalent, are not differentially equivalent and they require separate mapping rules as also pointed out in previous work on manifold [61] and subdivision [64]. There is no C^1 non-singular mapping from a convex domain to a non-convex domain. The arrangement of crease edges in Type 2 crease chart domains can sometimes lead to non-convex sectors on surfaces. When the map (5.20) is used in non-convex sectors it leads to foldover in the surface as shown in Figure 5.15. To cope with non-convex sectors, the map $\boldsymbol{\Psi}_j^{\mathrm{III}}$ for Type 3 crease chart is defined as

$$\boldsymbol{\Psi}_j^{\mathrm{III}} = \boldsymbol{\Psi}_{\boldsymbol{c}}^{\mathrm{III}} \circ \boldsymbol{\Psi}_{\boldsymbol{l}}, \quad \text{with} \quad j \in \{1,2,3,4\}. \tag{5.22}$$

Instead of using the quasi-conformal map $\boldsymbol{\Psi}_{\boldsymbol{c}}^{\mathrm{II}}$ defined in (5.21), a modified map $\boldsymbol{\Psi}_{\boldsymbol{c}}^{\mathrm{III}}$ is introduced as

$$\boldsymbol{\Psi}_{\boldsymbol{c}}^{\mathrm{III}}(\boldsymbol{\eta}_j; k_j, s_j, l_s, m_s) = \begin{cases} \boldsymbol{\Psi}_{\boldsymbol{c}}(\boldsymbol{\eta}_j; 4(k_j-1)l_s, (s_j-1)l_s + m_s) & \text{if } s_j < k_j \\ \boldsymbol{\Psi}_{\boldsymbol{c}}(\boldsymbol{\eta}_j; 4l_s/3, m_s)e^{i\pi/2} & \text{if } s_j = k_j \end{cases}, \tag{5.23}$$

where the number of creases k_j, the sector number s_j, the number of elements in a sector l_s and the local element number in a sector m_s are defined in the same way as mentioned in Section 5.3.2, see Figure 5.14. Essentially, the concave sector with $s_j = k_j$ is mapped to a L-shaped domain and all other sectors are mapped to the first quadrant, see Figure 5.13. In the concave sector, i.e., the L-shaped domain, three Bézier pieces are $C^{k\geq 1}$ continuously connected within the sector while they are C^0 continuously matched with polynomial pieces in other sectors along the crease edges. Here, to derive the continuity matching requirements, we have used the algorithms of Bézier curves introduced in Section 2.1, e.g., derivatives and degree elevation.

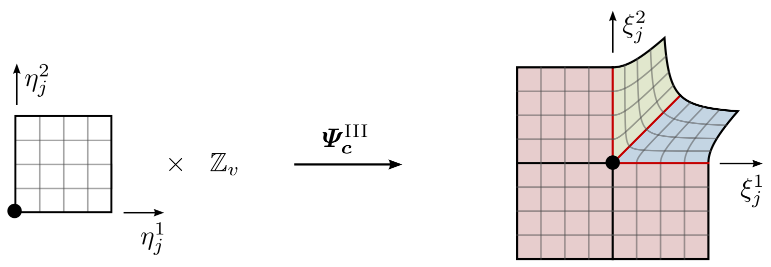


Figure 5.13: Example of Type 3 crease chart domain. The centre vertex has valence $v_j = 5$ and $k_j = 3$ creased edges. One of the sectors is non-convex.

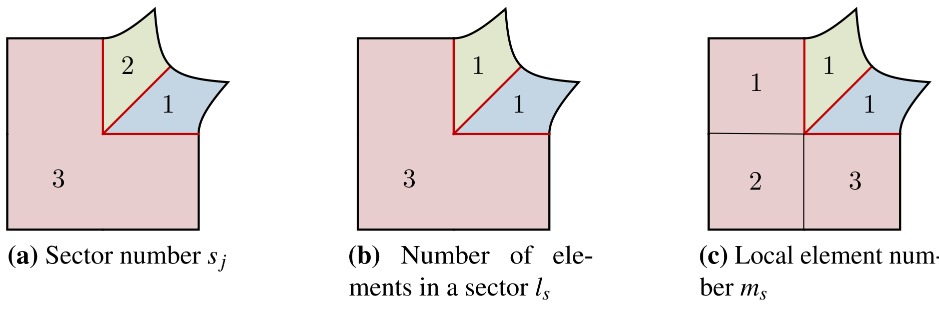

2
1
3
(a) Sector number s_j
1
1
3
(b) Number of elements in a sector l_s
1
1
1
2
3
(c) Local element number m_s

Figure 5.14: Illustration of the numbering used to define the map $\boldsymbol{\Psi}_{\boldsymbol{c}}^{\mathrm{III}}$ in (5.23), taking the example of the Type 3 crease chart shown in Figure 5.13. The number of creases $k_j = 3$ takes the same value for every element.

5.3.4 Examples with creases

We consider the construction of the manifold basis functions for one single element in the unstructured quadrilateral finite element mesh shown in Figure 5.16, which includes prescribed crease edges. There are four overlapping charts, see Figure 5.17. Notice that the mapping for the Type 2 crease chart with local vertex index $j = 1$ is different from the mapping for the smooth valence-5 chart in Figure 5.6, while the mappings for other charts are the same. Note that the presence of creased edges only changes the chart parametrisation for the construction of local sector-wise polynomials $\boldsymbol{p}_j(\boldsymbol{\xi}_j)$. The construction of weight functions $\boldsymbol{w}_j(\boldsymbol{\xi}_j)$ is the same for all charts regardless of the crease edges since the element-wise smooth weight functions are defined in the reference element and mapped to each element in the chart domains, see (5.4).

Figure 5.18 shows a geometry represented by an unstructured quadrilateral mesh, which includes prescribed creased edges. The eight charts of valence $v = 6$ include a smooth chart,

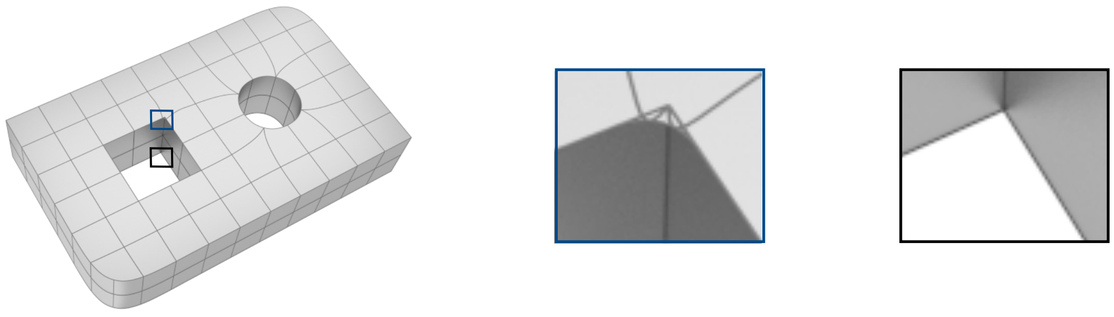


Figure 5.15: Behaviour of the manifold surface at concave corners. The close-up image in blue box (middle) shows the behaviour of Type 2 mapping, which develops a fold at the concave corner. In contrast, the close-up image in the black box (right) has no fold and demonstrates that Type 3 mapping is required for concave corners. In close-up images the control meshes are added for the purpose of visualisation. In Figure 5.1, the crease charts centred at the eight vertices around the square hole all use Type 3 mapping.

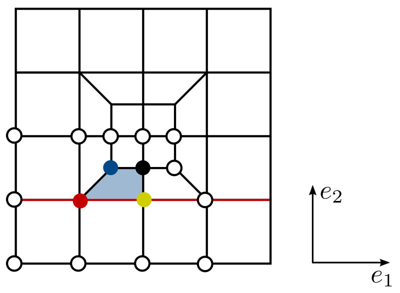


Figure 5.16: An unstructured mesh with extraordinary vertices of valence $v \in 3,5$. The shaded element is overlapped by four charts and the union of the four charts have 16 unique vertices in total. Edges tagged as crease are coloured in red.

four Type 1 crease charts and three Type 2 crease charts. In the constructed manifold surface, all creases are faithfully reproduced.

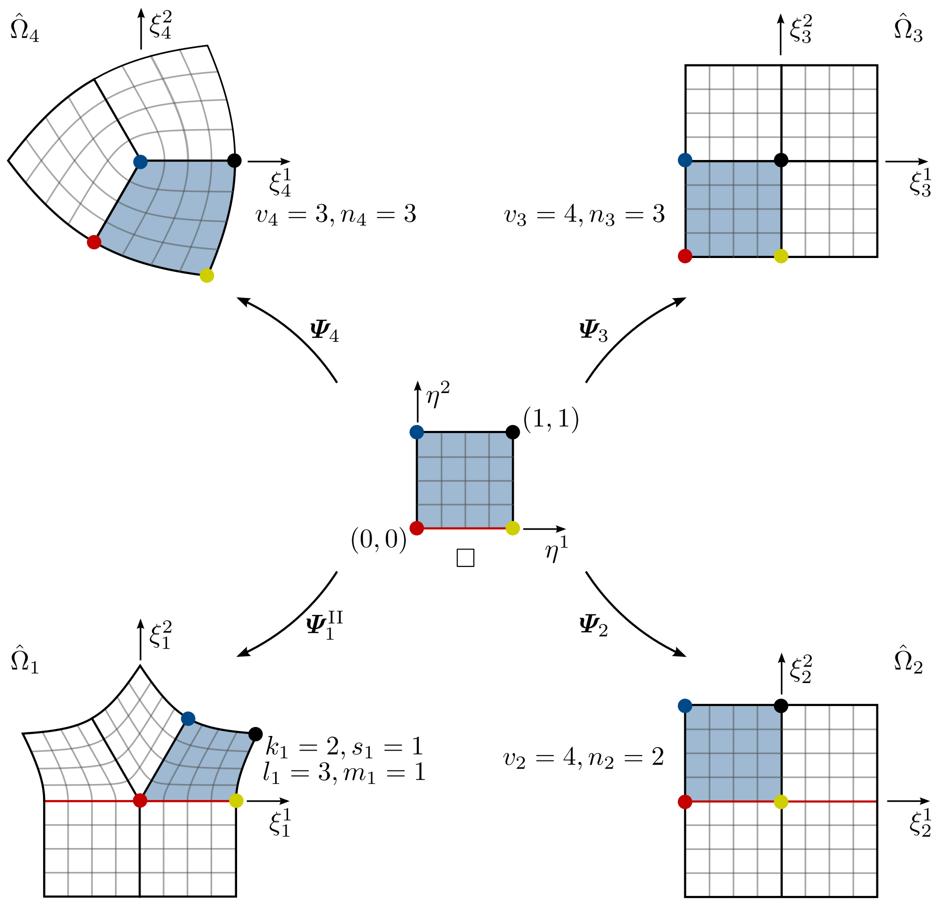


Figure 5.17: The reference element $\square$ (center) and its four overlapping chart domain $\hat{\Omega}_j$ with $j \in \{1,2,3,4\}$. The reference element is mapped with $\boldsymbol{\Psi}_j$ to the blue shaded elements in the four charts $\hat{\Omega}_j$. Notice that the parameter lines in the charts with $\eta^1 =$ const. and $\eta^2 =$ const. are always orthogonal to the spoke edges which guarantees that the derivatives of $\hat{\Omega}_j$ are continuous over the entire chart.

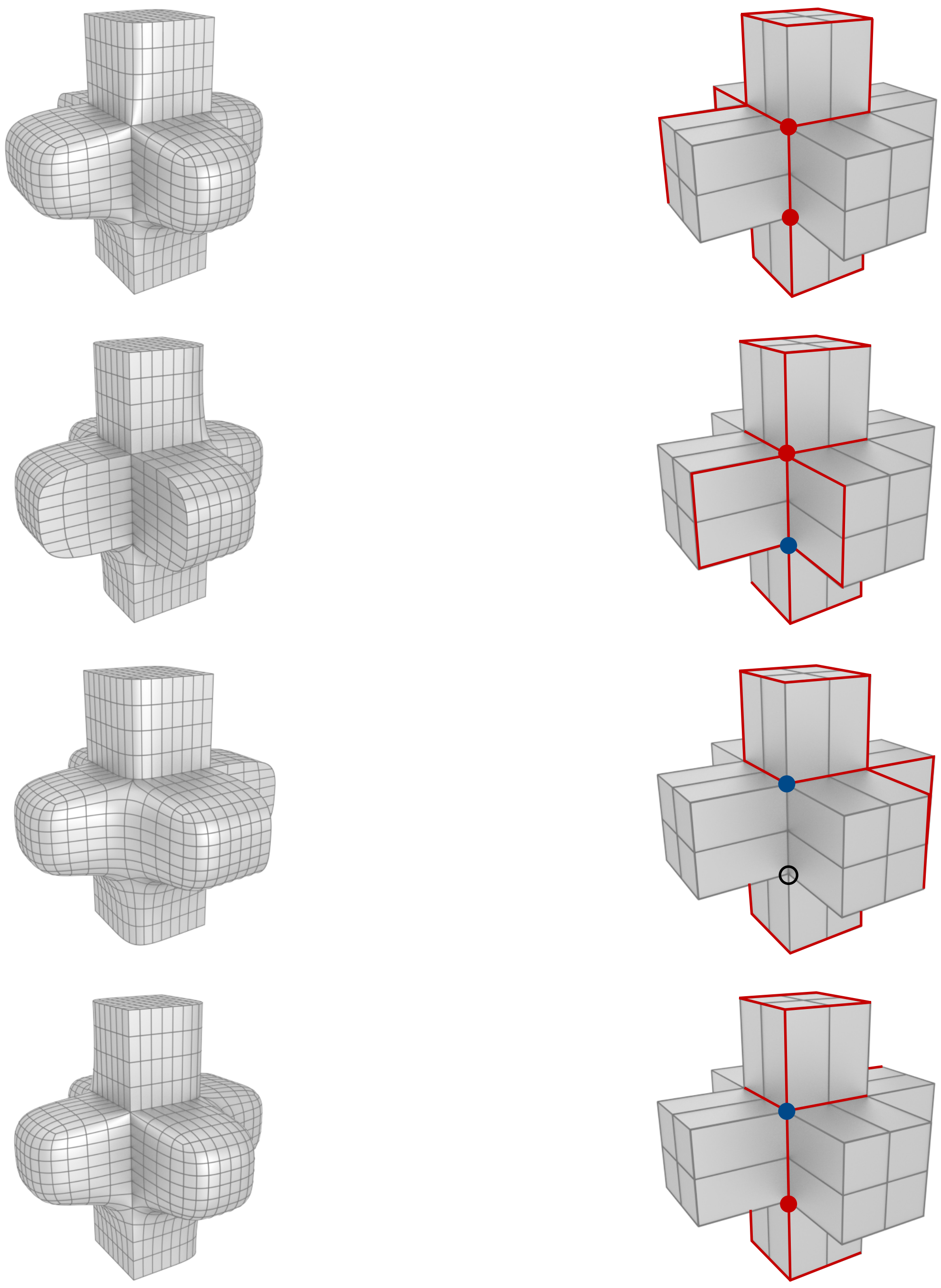


Figure 5.18: Four different views of a manifold surface (left) modelled by using control points on the limit surface of Catmull-Clark subdivision given the control mesh (right). In the control mesh, crease edges are marked as red. This mesh includes eight valence-6 vertices and the corresponding charts are categorised as smooth chart (empty dot), Type 1 (red dot) or Type 2 (blue dot) crease charts.

5.4 Thin-shell analysis

In this section, we briefly review the Kirchhoff-Love thin-shell equations and their finite element discretisation. In addition, we introduce how we apply normal constraints in thin-shell analysis, which will be used in the following numerical examples.

5.4.1 Kinematics

With the convected coordinates (η^1, η^2, η^3), the position vector $\boldsymbol{\varphi}(\eta^1, \eta^2, \eta^3)$ of a material point in the deformed configuration of the shell is assumed to be

$$\boldsymbol{\varphi}(\eta^1, \eta^2, \eta^3) = \boldsymbol{x}(\eta^1, \eta^2) + \eta^3 \boldsymbol{a}_3(\eta^1, \eta^2) \quad \text{with} \quad \eta^3 \in \left[-\frac{t}{2}, \frac{t}{2}\right], \tag{5.24}$$

where $\boldsymbol{x}(\eta^1, \eta^2)$ is the position vector on the shell mid-surface, $\boldsymbol{a}_3(\eta^1, \eta^2)$ the unit normal to the mid-surface and t the shell thickness. The definition in the reference configuration follows similarly

$$\boldsymbol{\Phi}(\eta^1, \eta^2, \eta^3) = \boldsymbol{X}(\eta^1, \eta^2) + \eta^3 \boldsymbol{A}_3(\eta^1, \eta^2). \tag{5.25}$$

The covariant base vectors $\boldsymbol{a}_\alpha$ on the mid-surface are computed as

$$\boldsymbol{a}_1 = \frac{\partial \boldsymbol{x}}{\partial \eta^1}, \quad \boldsymbol{a}_2 = \frac{\partial \boldsymbol{x}}{\partial \eta^2}, \quad \boldsymbol{a}_3 = \frac{\boldsymbol{a}_1 \times \boldsymbol{a}_2}{|\boldsymbol{a}_1 \times \boldsymbol{a}_2|}, \quad a_{\alpha\beta} = \boldsymbol{a}_\alpha \cdot \boldsymbol{a}_\beta, \quad \alpha, \beta \in \{1, 2, 3\}. \tag{5.26}$$

The contravariant base vectors $\boldsymbol{a}^\alpha$ on the mid-surface are determined by

$$\boldsymbol{a}^\alpha \cdot \boldsymbol{a}_\beta = \delta^\alpha_\beta, \quad a^{\alpha\beta} = \boldsymbol{a}^\alpha \cdot \boldsymbol{a}^\beta. \tag{5.27}$$

The covariant base vectors $\boldsymbol{A}_\alpha$ and contravariant base vectors $\boldsymbol{A}^\alpha$ in the reference configuration are defined similarly.

With these definitions, the deformation gradient can be expressed as

$$\boldsymbol{F} = \frac{\partial \boldsymbol{\varphi}}{\partial \boldsymbol{\Phi}} = \frac{\partial \boldsymbol{\varphi}}{\partial \eta^\alpha} \frac{\partial \eta^\alpha}{\partial \boldsymbol{\Phi}} = \boldsymbol{a}_\alpha \otimes \boldsymbol{A}^\alpha, \tag{5.28}$$

and the Green-Lagrange strain tensor as

$$\boldsymbol{E} = \frac{1}{2}(\boldsymbol{F}^\mathsf{T} \boldsymbol{F} - \boldsymbol{I}). \tag{5.29}$$

A straightforward calculation leads to

$$\boldsymbol{E} = \boldsymbol{\alpha} + \eta^3 \boldsymbol{\beta} + (\eta^3)^2 \cdots , \tag{5.30}$$

with the components

$$\boldsymbol{\alpha} = \frac{1}{2}(\boldsymbol{a}_\alpha \cdot \boldsymbol{a}_\beta - \boldsymbol{A}_\alpha \cdot \boldsymbol{A}_\beta) \boldsymbol{A}^\alpha \otimes \boldsymbol{A}^\beta , \tag{5.31a}$$

$$\boldsymbol{\beta} = \frac{1}{2}(\boldsymbol{a}_\alpha \cdot \boldsymbol{a}_{3,\beta} + \boldsymbol{a}_\beta \cdot \boldsymbol{a}_{3,\alpha} - \boldsymbol{A}_\alpha \cdot \boldsymbol{A}_{3,\beta} - \boldsymbol{A}_\beta \cdot \boldsymbol{A}_{3,\alpha}) \boldsymbol{A}^\alpha \otimes \boldsymbol{A}^\beta . \tag{5.31b}$$

5.4.2 Thin-shell energy functional and its discretisation

The potential energy of the shell consists of internal and external energy, i.e,

$$\Pi[\boldsymbol{x}] = \Pi^{\text{int}}[\boldsymbol{x}] + \Pi^{\text{ext}}[\boldsymbol{x}] . \tag{5.32}$$

The internal potential energy can be computed as

$$\Pi^{\text{int}}[\boldsymbol{x}] = \int_\Omega W(\boldsymbol{\alpha}, \boldsymbol{\beta}) \, \mathrm{d}\Omega \tag{5.33}$$

$$= \int_\Omega \left(\frac{1}{2} \frac{Et}{1-\nu^2} H^{\alpha\beta\gamma\delta} \alpha_{\alpha\beta} \alpha_{\gamma\delta} + \frac{1}{2} \frac{Et^3}{12(1-\nu^2)} H^{\alpha\beta\gamma\delta} \beta_{\alpha\beta} \beta_{\gamma\delta} \right) \mathrm{d}\Omega , \tag{5.34}$$

where E is Young's modulus, ν Poisson's ratio, t thickness and H a constant fourth-order tensor with components

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

The potential energy of applied loads takes the form

$$\Pi^{\text{ext}}[\boldsymbol{x}] = -\int_\Omega \boldsymbol{q} \cdot (\boldsymbol{x} - \boldsymbol{X}) \, \mathrm{d}\Omega - \int_\Gamma \boldsymbol{N} \cdot (\boldsymbol{x} - \boldsymbol{X}) \, \mathrm{d}\Gamma , \tag{5.35}$$

where $\boldsymbol{q}$ is the distributed loads per unit area of Ω and $\boldsymbol{N}$ the axial forces per unit length of Γ.

To derive discrete equations, $\boldsymbol{x}$ and $\boldsymbol{X}$ in the potential energy functional (5.32) are replaced with the following finite element approximations

$$\boldsymbol{x}^h(\eta^1, \eta^2) = \sum_J N_J(\eta^1, \eta^2) \boldsymbol{x}_J , \quad \boldsymbol{X}^h(\eta^1, \eta^2) = \sum_J N_J(\eta^1, \eta^2) \boldsymbol{X}_J , \tag{5.36}$$

and the integrals are evaluated with Gauss quadrature. According to the principle of minimum potential energy, the discrete equilibrium equations can be derived as

$$\frac{\partial \Pi(\boldsymbol{x}^h)}{\partial \boldsymbol{x}_J} = \frac{\partial \Pi^{\text{int}}(\boldsymbol{x}^h)}{\partial \boldsymbol{x}_J} + \frac{\partial \Pi^{\text{ext}}(\boldsymbol{x}^h)}{\partial \boldsymbol{x}_J} = \boldsymbol{0}, \tag{5.37}$$

which are nonlinear equations because the internal potential energy $\Pi^{\text{int}}(\boldsymbol{x}_J)$ is a nonlinear function of the nodal position vector $\boldsymbol{x}_J$. We solve the nonlinear discrete equations using the Newton-Raphson method, i.e.,

$$\frac{\partial^2 \Pi\left(\boldsymbol{x}^{(n-1)}\right)}{\partial \boldsymbol{x}_I \partial \boldsymbol{x}_J}\left(\boldsymbol{x}^{(n)} - \boldsymbol{x}^{(n-1)}\right) = -\frac{\partial \Pi\left(\boldsymbol{x}^{(n-1)}\right)}{\partial \boldsymbol{x}_I}, \tag{5.38}$$

where $\boldsymbol{x}^{(n-1)}$ refers to the solution obtained at the previous iteration step and $\boldsymbol{x}^{(n)}$ is going to be determined at the current iteration. It becomes clear that $\partial^2\Pi/\partial \boldsymbol{x}_I \partial \boldsymbol{x}_J$ contributes to the left-hand side matrix and $\partial \Pi/\partial \boldsymbol{x}_I$ to the right-hand side vector. For brevity, here we do not include the full expression of (5.38), which can be found in [6, 65].

5.4.3 Normal constraints

When rigid joints and clamped boundaries are present, it is necessary to constrain the surface normals, which is realised by using penalty method in our implementation. To this end penalty terms are added to the potential energy functional (5.32) as follows:

$$\Pi^C[\boldsymbol{x}] = \Pi[\boldsymbol{x}] + \underbrace{\frac{\gamma_1}{2}\int_{\Gamma_\text{r}} (\boldsymbol{a}_3^l \cdot \boldsymbol{a}_3^r - \boldsymbol{A}_3^l \cdot \boldsymbol{A}_3^r)^2 d\Gamma}_{\text{rigid edges}} + \underbrace{\frac{\gamma_2}{2}\int_{\Gamma_\text{c}} (\boldsymbol{a}_3 - \boldsymbol{A}_3)\cdot(\boldsymbol{a}_3 - \boldsymbol{A}_3) d\Gamma}_{\text{clamped edges}}, \tag{5.39}$$

where $\boldsymbol{a}_3^l$ and $\boldsymbol{a}_3^r$ denote the mid-surface normal on the left and right side of the rigid joint in the deformed configuration, and $\boldsymbol{A}_3^l$ and $\boldsymbol{A}_3^r$ in the reference configuration. γ_1 and γ_2 are penalty parameters. Obviously, the penalty terms vanish in the absence of rigid joints and clamped boundaries, in which case (5.39) is identical to (5.32).

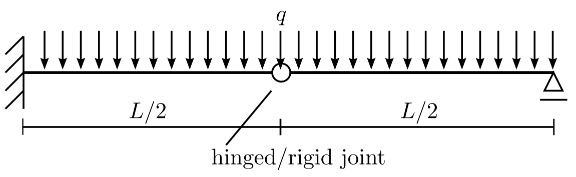
5.5 Examples

We consider several representative examples to demonstrate the versatility of the proposed manifold basis functions in finite element analysis. In all examples, the Kirchhoff-Love model is used for thin-shell analysis and Gauss quadrature is used for numerical integration, as introduced in Section 5.4. The linear Kirchhoff-Love model is used for the bending beam and plate problems, while a nonlinear model is solved with Newton-Raphson method for the pinched tube example. For convergence studies in the first two examples, we use 9×9 Gauss quadrature points per element. However, for the nonlinear simulation of the pinched tube, we use 3×3 quadrature points due to the large number of iterations needed.

Regarding the manifold constructions, all domain boundaries are tagged as crease edges, excluding the need of ghost cells as in [24]. In all computations, weight functions are chosen to be cubic B-splines and local basis is (piecewise) tensor-quadratic polynomial. To be able to use tensor-quadratic basis in all charts, middle-face and middle-edge vertices are introduced, resulting in $6v+1$ control points in each chart of valence v. In all examples, the normal constraints are enforced by using penalty method, as explained in Section 5.4.3, with the penalty parameters chosen to be $\gamma = \mathcal{O}(E)$ where E is Young's modulus.

5.5.1 Propped cantilever beam with a hinge

As an introductory example, we compute the deformation of a beam subjected to uniform transversal loading, see Figure 5.19. In our computations, there are three different settings at the centre: (1) continuous beam: all interior charts are smooth; (2) hinged beam: the middle vertex is tagged as crease; (3) joined beam: the middle vertex is tagged as crease, where the continuity of normal is enforced. This example is used to demonstrate the ability of the proposed manifold basis functions to model arbitrary smoothness with exact boundary and normal control.



Geometry:	$L = 1.0$ $t = 0.01$
Material property:	$E = 3.0 \cdot 10^6$
Uniform loading:	$q = 1.0$

Figure 5.19: Geometry and loading of the propped cantilever beam with a hinge.

The analytical deflection for the continuous beam is

$$u(x) = \frac{q}{4Et^3}(2x^4 - 5Lx^3 + 3L^2x^2)\,, \tag{5.40}$$

and for the hinged beam is

$$u(x) = \begin{cases} \frac{q}{2Et^3}(x^4 - 3Lx^3 + 3L^2x^2) & \text{if } x < L/2 \\ \frac{q}{2Et^3}(x^4 - 3Lx^3 + 3L^2x^2 - 2L^3x + L^4) & \text{if } x \geq L/2 \end{cases}. \tag{5.41}$$

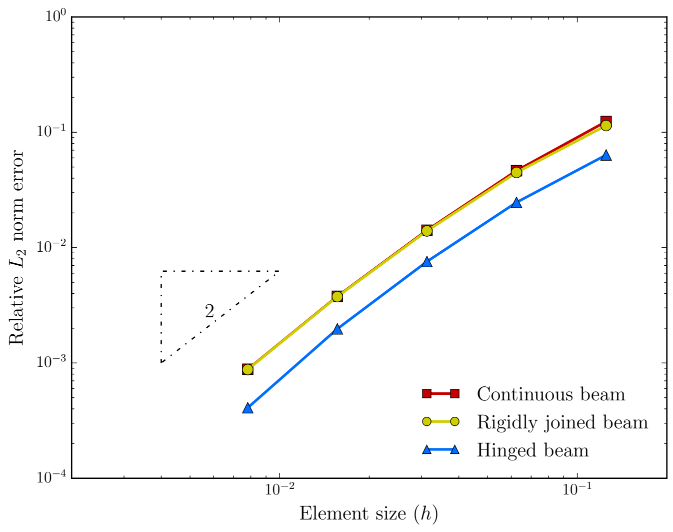


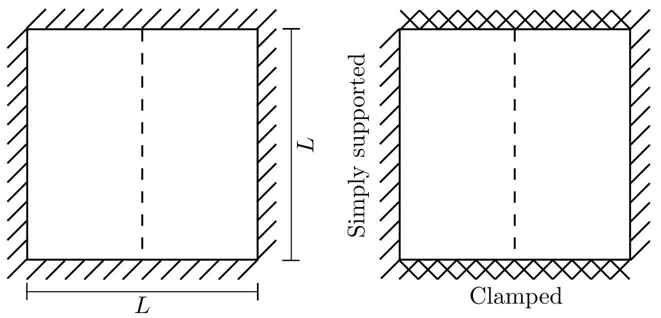
Figure 5.20: Convergence of L_2 error of the bending beam. Cubic weight functions and quadratic local polynomials are used.

The beam is discretised with a uniform mesh. Both the left and right ends are tagged as crease, excluding the need of ghost cells. The left end is clamped requiring the constraint of normal, while the right end is simply supported. Figure 5.20 plots the relative L_2 norm error, which converges at the optimal rate 2 in all situations. As expected, the continuous beam and joined beam lead to similar results.

5.5.2 Bending plate

Next, we compute the deformation of a square plate subjected to uniform loading. Two types of boundary conditions are considered: (1) all edges are simply supported; (2) two opposite edges are simply supported and the other two clamped. Figure 5.21 shows the problem definition and Figure 5.22 shows two meshes used in this computation. The analytical deflection for both boundary conditions can be found in [59, Chapter 5 and 6]. The deformed shapes under both boundary conditions are shown in Figure 5.23. It is clear that the plate deforms very differently close to simply supported and clamped edges.

The plate is discretised with meshes shown in Figure 5.22, which are refined with Catmull-Clark subdivision for the convergence study. Figure 5.24 plots the relative L_2 norm error for both boundary conditions with or without clamped edges. It is noteworthy that the manifold basis functions achieve the optimal convergence order even with the presence of extraordinary vertices in the meshes.



Geometry:	$L = 1.0$
	$t = 0.01$
Material property:	$E = 70.0 \cdot 10^9$
	$\nu = 0.3$
Uniform loading:	$q = 10.0 \cdot 10^3$

Figure 5.21: Definition of the plate bending problem. Two types of boundary conditions are considered, either all edges simply supported (left) or two opposite edges simply supported and the other two clamped.

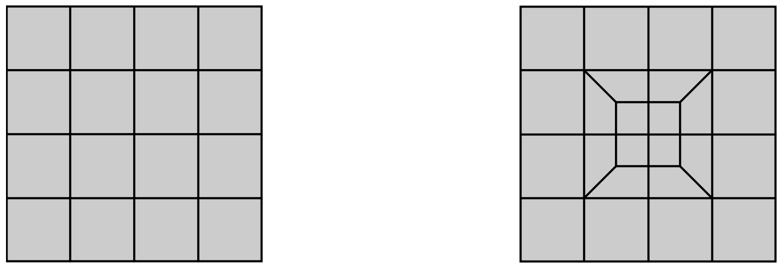


Figure 5.22: The structured (left) and unstructured (right) coarse meshes used for computations.

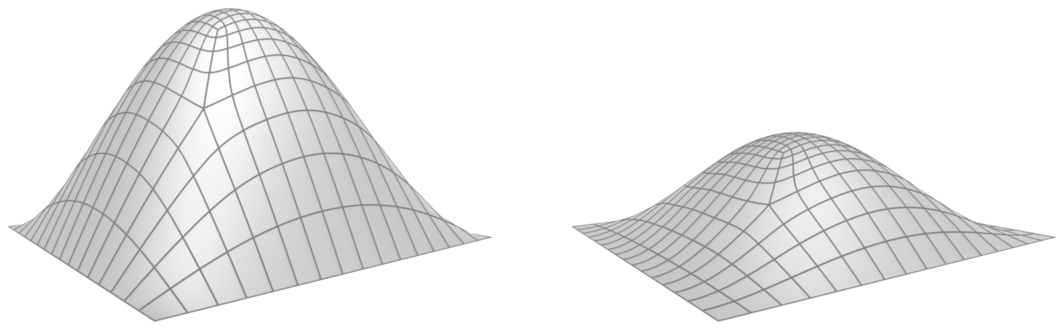
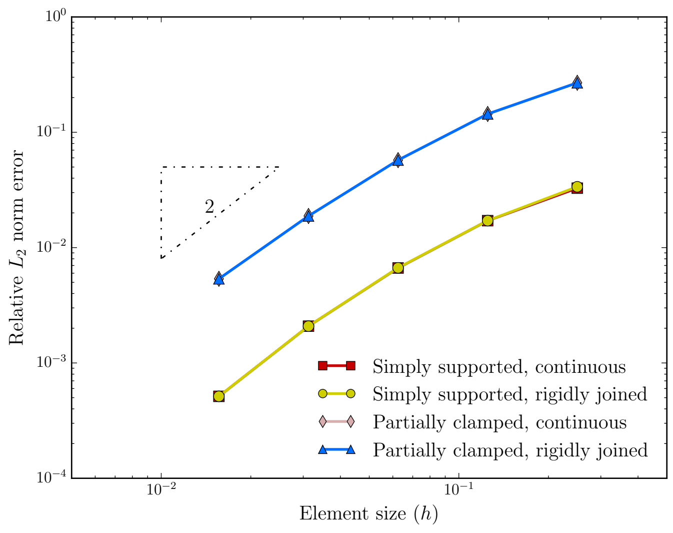


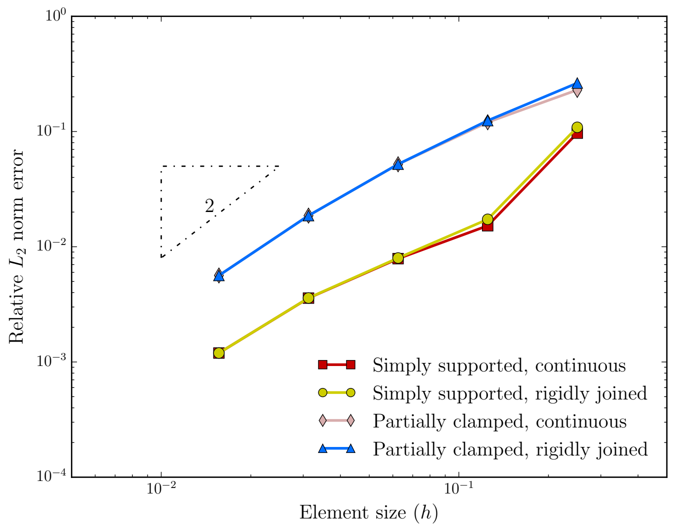
Figure 5.23: Deformed shapes of the plate under two different boundary conditions: Left: all edges are simply supported; Right: two opposite edges are simply supported and the other two clamped. The deflections are scaled with the same factor in the two cases.

5.5.3 Pinched square tube

In this last example, we compute the deformation of a pinched square tube subjected to two diametrically opposite concentrated forces, shown in Figure 5.25. The tube is discretised with a uniform structured mesh, where the element size is $h = L/16$, or with an unstructured mesh as shown in Figure 5.26. To simulate the rigid joint, relevant edges are tagged as crease and the angle between two corresponding normals is enforced to be constant during deformation.



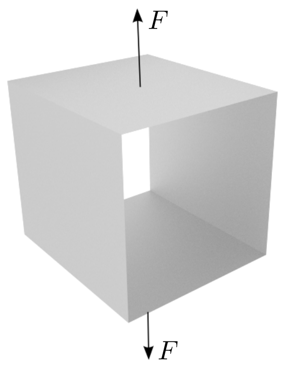
(a) Structured mesh



(b) Unstructured mesh

Figure 5.24: Convergence of relative L_2 norm error for two different boundary conditions discretised by either structured mesh or unstructured mesh. The coarse meshes shown in Figure 5.22 are refined up to five times with Catmull-Clark subdivision for the convergence study.

Figure 5.26 presents the deformed tubes at three different load values. It is clear that the right angles are well preserved in all deformed shapes. The nonlinear load-displacement curve is plotted in Figure 5.27. This example demonstrates that the proposed manifold basis functions can be used in thin-shell analysis with rigid joints.



Geometry:	$L = 1.0$
	$t = 0.01$
Material property:	$E = 3.0 \cdot 10^6$
	$\nu = 0.0$
Concentrated force:	$F = 1$–100

Figure 5.25: Geometry and loading of the pinched square tube.

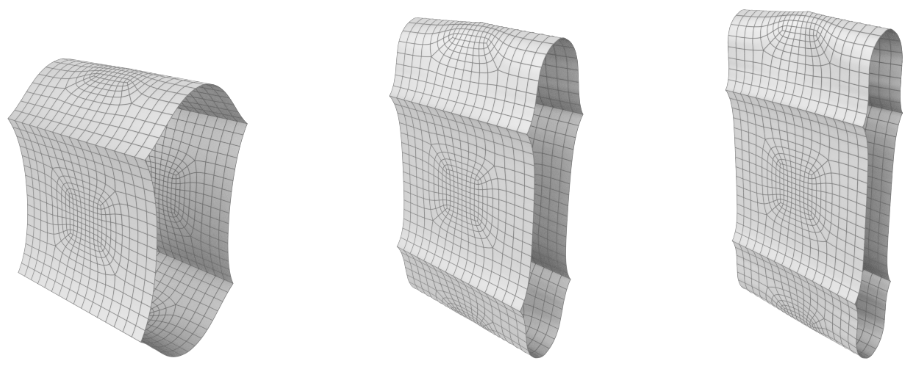


Figure 5.26: Deformed shapes of the pinched tube at different load values. $F = 10, 50, 100$ (left to right).

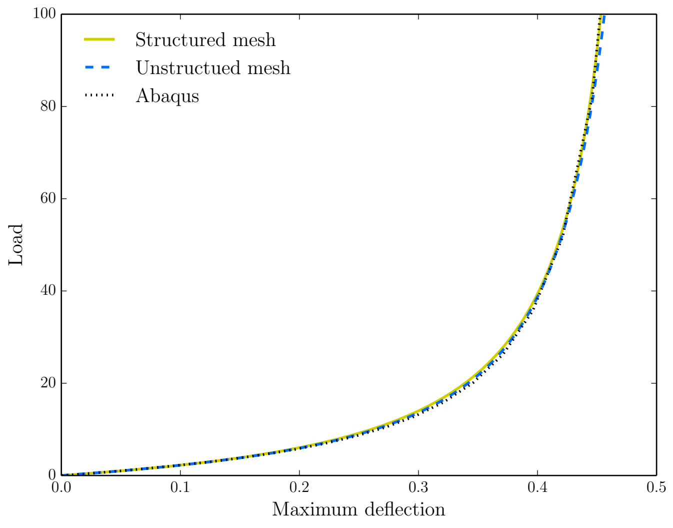


Figure 5.27: Load-displacement curve of the pinched square tube at the load attachment point.

Chapter 6

Manifold-spline basis functions

A spline is a piecewise polynomial, whose pieces meet with the highest possible continuity for the given polynomial degree. Tensor-product of spline curves generates spline surfaces defined on structured quadrilateral meshes. In Chapter 2, we have learned that splines have different representations, such as Bézier form and B-spline form. The B-spline representation is so far the most convenient one in many implementations, including geometry modelling and finite element approximation. In this chapter, we introduce the manifold construction for splines focusing on the univariate setting. This is motivated by the observation that manifold-based basis functions can achieve optimal convergence in finite element analysis on unstructured meshes [24], see also the numerical examples in Chapter 5. We aim to construct manifold-spline basis functions that reproduce spline surfaces, i.e., subdivision surfaces in regular region, and meanwhile maintain their optimal convergence around extraordinary vertices. To fix ideas, we consider first only univariate cubic manifold-spline basis functions. It is however straightforward to generalise the proposed techniques to the bivariate case and higher (odd) degree splines.

6.1 Manifold construction representing B-spline curves

We first consider a particular manifold construction that automatically leads to B-splines. Consider the cubic uniform B-splines $B_j^3(\xi)$ shown in Figure 6.1. Since we only discuss about cubic B-splines, for brevity we will omit the superscript 3 in $B_j^3(\xi)$ which denotes polynomial degree. Thus $B_j(\xi)$ denotes a uniform cubic B-spline basis function centred at the knot $\xi = j$. To keep the discussion concise, at the moment we use a global parametrisation rather than the local parametrsiation for each reference element $\square := [0,1] \ni \eta$. In any

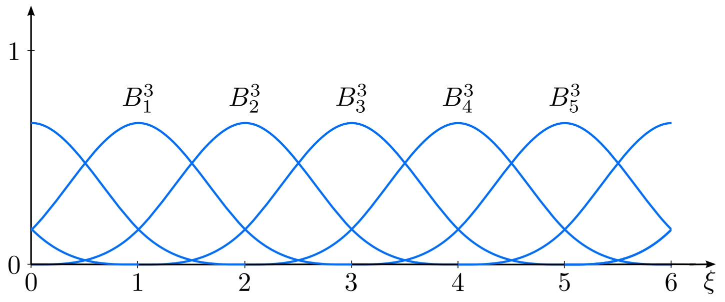


Figure 6.1: Cubic B-spline basis functions.

interval $\xi \in [n, n+1]$, a B-spline function is represented as

$$f^B(\xi) = \sum_{j=n-1}^{n+2} B_j(\xi)\beta_j\,, \quad \xi \in [n, n+1]\,. \tag{6.1}$$

As introduced in Section 5.2.1, we consider manifold charts $\hat{\Omega}_j := [j-1, j+1] \ni \xi_j$ consisting of two elements. As a result, any interval $[n, n+1]$ is overlapped by two manifold charts $\hat{\Omega}_n$ and $\hat{\Omega}_{n+1}$. The corresponding univariate manifold representation reads

$$f^M(\xi) = w_n(\xi) f_n(\xi) + w_{n+1}(\xi) f_{n+1}(\xi)\,, \quad \xi \in [n, n+1] \tag{6.2}$$

where $w_j(\xi)$ and $f_j(\xi)$ are the weight functions and local polynomial approximants in the manifold chart $\hat{\Omega}_j$. The weight functions form a partition of unity, i.e.,

$$w_n(\xi) + w_{n+1}(\xi) = 1\,, \quad \xi \in [n, n+1]\,. \tag{6.3}$$

One straightforward approach to reproduce the B-spline function $f^B(\xi)$ in the considered interval is to choose in manifold construction (6.2) the local polynomial approximants $f_j(\xi)$ in both charts to be identical to the spline function $f^B(\xi)$, i.e.,

$$f_n(\xi) = f_{n+1}(\xi) = f^B(\xi) \quad \xi \in [n, n+1]\,. \tag{6.4}$$

Given the particular choice (6.4) and the partition of unity (6.3), we can rewrite the manifold representation (6.2) as

$$f^M(\xi) = w_n(\xi) f^B(\xi) + (1 - w_n(\xi)) f^B(\xi) = f^B(\xi)\,. \tag{6.5}$$

This seems to be an obvious observation that due to the partition of unity property of weight functions $w_j(\xi)$ the reproduction of B-spline function $f^B(\xi)$ follows immediately when the local polynomial approximation in each manifold chart is chosen as $f_j(\xi) = f^B(\xi)$. Recall that, as mentioned in Section 5.2.1, the local approximants $f_j(\xi)$ in manifold construction are often represented as

$$f_j(\xi) = \boldsymbol{p}_j^{\mathsf{T}}(\xi)\boldsymbol{\alpha}_j\,, \tag{6.6}$$

where $\boldsymbol{p}_j(\xi)$ is a vector containing a polynomial basis and $\boldsymbol{\alpha}_j$ a vector containing the coefficients. In conventional smooth manifold constructions, in each chart domain $\boldsymbol{p}_j(\xi)$ is a C^∞ continuous polynomial basis rather than a piecewise polynomial basis. However, if we would like to satisfy the particular choice (6.4), the local basis $\boldsymbol{p}_j(\xi)$ must be chosen as a piecewise polynomial. This is because the B-spline function $f^B(\xi)$ is essentially a piecewise cubic polynomial which is C^2 continuous at element boundaries.

One simple approach is to choose $\boldsymbol{p}_j(\xi)$ as B-spline basis functions and it follows that $\boldsymbol{\alpha}_j$ should be the B-spline control coefficients. There is no need for least square fitting in regular charts. For example, we consider the domain interval $\xi \in [2,4]$ overlapped by three charts centred at $\xi = 2,3,4$, respectively. The local approximations in the three relevant charts read

$$\begin{aligned}
f_2(\xi) &= B_0(\xi)\beta_0 + B_1(\xi)\beta_1 + B_2(\xi)\beta_2 + B_3(\xi)\beta_3 + B_4(\xi)\beta_4\,, \quad \xi \in \hat{\Omega}_2 := [1,3]\,.\\
f_3(\xi) &= B_1(\xi)\beta_1 + B_2(\xi)\beta_2 + B_3(\xi)\beta_3 + B_4(\xi)\beta_4 + B_5(\xi)\beta_5\,, \quad \xi \in \hat{\Omega}_3 := [2,4]\,.\\
f_4(\xi) &= B_2(\xi)\beta_2 + B_3(\xi)\beta_3 + B_4(\xi)\beta_4 + B_5(\xi)\beta_5 + B_6(\xi)\beta_6\,, \quad \xi \in \hat{\Omega}_4 := [3,5]\,.
\end{aligned} \tag{6.7}$$

In the knot interval $\xi \in [2,3]$, which is overlapped by two manifold charts $\hat{\Omega}_2$ and $\hat{\Omega}_3$, we have

$$\begin{aligned}
f^M(\xi) &= (1 - w_3(\xi))f_2(\xi) + w_3(\xi)f_3(\xi)\\
&= (1 - w_3(\xi))B_0(\xi)\beta_0 + B_1(\xi)\beta_1 + B_2(\xi)\beta_2 + B_3(\xi)\beta_3 + B_4(\xi)\beta_4 + w_3(\xi)B_5(\xi)\beta_5\\
&= B_1(\xi)\beta_1 + B_2(\xi)\beta_2 + B_3(\xi)\beta_3 + B_4(\xi)\beta_4\\
&= f^B(\xi)
\end{aligned} \tag{6.8}$$

where we have used (6.2), (6.3), (6.7) together with $B_0(\xi) = B_5(\xi) = 0$ for $\xi \in [2,3]$. Similarly, we can show $f^M(\xi) = f^B(\xi)$ for $\xi \in [3,4]$.

As this approach uses the same control points as B-spline curves, it is straightforward to interface it with B-splines and subdivision surfaces. Only close to an EV the manifold construction becomes active. However, in this construction each chart has some supporting vertices located outside the chart domain, which differs from the conventional manifold concept. Next, we discuss how to reproduce B-splines through the approach of least-squares fitting.

6.2 Fitting manifold splines to B-splines

As discussed in previous section, the key to reproduce B-spline functions is to reproduce B-spline functions by the local approximant in each chart, see (6.4) and (6.5). This requires

$$f_j(\xi) = \boldsymbol{p}_j^{\mathsf{T}}(\xi)\boldsymbol{\alpha}_j = f^B(\xi)\,, \tag{6.9}$$

where the local approximation basis $\boldsymbol{p}_j(\xi)$ is chosen as a piecewise polynomial basis and the coefficients $\boldsymbol{\alpha}_j$ are determined through least-squares fitting. Note that the B-spline function $f^B(\xi)$ is a piecewise cubic polynomial. To determine a cubic polynomial, four sample points are needed. This means (6.9) can be satisfied if

1. In each chart, local basis functions are piecewise cubic polynomials, e.g., cubic B-spline, Bézier or Lagrange. (We do not need to worry about the continuity of local basis functions at element boundaries. This is because the continuity is inherited in the sample points we are trying to fit to.)
2. In each chart, sufficient sample points calculated from the B-spline function $f^B(\xi)$ are used for least-squares fitting. For example, seven sample points (three in each element plus one in the centre) are sufficient for C^0 local basis, such as Bézier or Lagrange. If we use cubic B-splines which are C^2 continuous, five sample points are sufficient because there are only five degrees of freedom in each chart.

In a regular chart $\hat{\Omega}_j$, the least squares fitting is performed as follows:

$$\boldsymbol{A}\boldsymbol{\alpha}_j = \boldsymbol{x}_j \quad \text{with} \quad A_{IJ} = p_J(\xi_I^*)\,, \tag{6.10}$$

where the sample points $\boldsymbol{x}_j$ in each chart are the limit positions (or, points on the B-spline curve) computed from

$$\boldsymbol{x}_j = \boldsymbol{M}\boldsymbol{\beta}_j \quad \text{with} \quad M_{IJ} = B_J(\xi_I^*)\,. \tag{6.11}$$

Notice that if we use B-splines as local basis functions in each chart, then $\boldsymbol{A} = \boldsymbol{M}$, which leads to

$$\boldsymbol{\alpha}_j = (\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\mathsf{T}}\boldsymbol{x}_j = (\boldsymbol{A}^{\mathsf{T}}\boldsymbol{A})^{-1}\boldsymbol{A}^{\mathsf{T}}\boldsymbol{M}\boldsymbol{\beta}_j = \boldsymbol{\beta}_j \tag{6.12}$$

and the local approximation becomes

$$f_j(\xi) = \boldsymbol{p}_j(\xi)\boldsymbol{\alpha}_j = \boldsymbol{B}_j(\xi)\boldsymbol{\beta}_j = f^B(\xi)\,. \tag{6.13}$$

which shows the least-squares fitting results in (6.4). Together with (6.5), it is straightforward to show that this manifold construction reproduces B-spline in regular region.

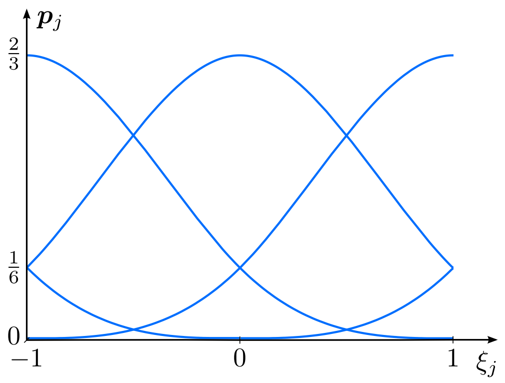


Figure 6.2: Local polynomial basis in a manifold regular chart domain.

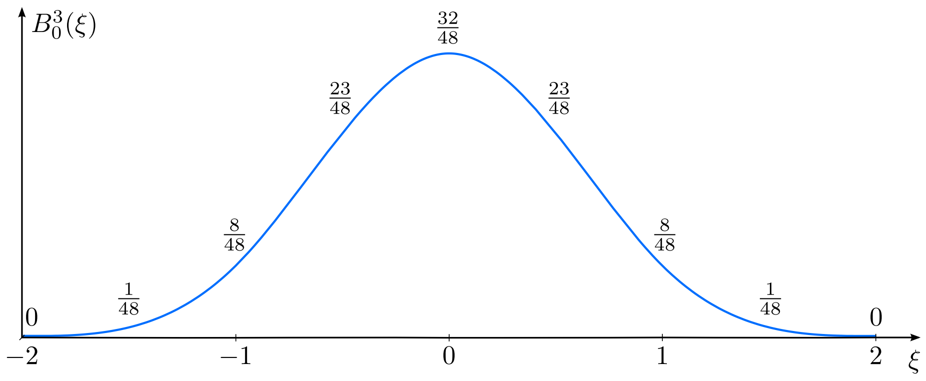


Figure 6.3: A uniform cubic B-spline basis function.

6.3 Cubic manifold-spline basis functions

In each chart domain $\hat{\Omega}_j := [j-1, j+1] \ni \xi_j$, the local polynomial basis $\boldsymbol{p}_j(\xi_j)$ is chosen to be uniform cubic B-splines as plotted in Figure 6.2 and the weight function $w_j(\xi_j)$ is constructed from cubic B-splines in the same way as introduced in Section 5.2.1. We plot one cubic uniform B-spline basis function and mark its value at key points in Figure 6.3. All other cubic uniform B-splines are obtained by shifting $B_0^3(\xi)$.

In the following, we use the least-square fitting approach as discussed in Section 6.2 to derive the cubic manifold spline basis functions. As the local basis $\boldsymbol{p}_j(\xi_j)$ has five degrees of freedoms, five sample points are used for least squares fitting, i.e., $\boldsymbol{\xi}^* = [-1, -0.5, 0, 0.5, 1]$.

The least square fitting matrix in (6.10) is readily computed as

$$\boldsymbol{A} = \frac{1}{48}\begin{bmatrix} 8 & 32 & 8 & 0 & 0 \\ 1 & 23 & 23 & 1 & 0 \\ 0 & 8 & 32 & 8 & 0 \\ 0 & 1 & 23 & 23 & 1 \\ 0 & 0 & 8 & 32 & 8 \end{bmatrix}. \tag{6.14}$$

Each row is the complete local basis $\boldsymbol{p}_j(\xi_j)$ evaluated at one sample point ξ_I^* and thus sums to one. Each column refers to one local basis function evaluated at all the samples points. Given $\boldsymbol{A}$ is a square matrix, least-squares fitting (6.10) simply results in

$$\boldsymbol{\alpha}_j = \boldsymbol{A}^{-1}\boldsymbol{x}_j = \frac{1}{6}\begin{bmatrix} 47 & -88 & 68 & -24 & 3 \\ -3 & 24 & -22 & 8 & -1 \\ 1 & -8 & 20 & -8 & 1 \\ -1 & 8 & -22 & 24 & -3 \\ 3 & -24 & 68 & -88 & 47 \end{bmatrix}\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ x_4 \end{bmatrix}. \tag{6.15}$$

Therefore, in each chart the local approximant $f_j(\xi_j)$ is represented as

$$f_j(\xi_j) = \boldsymbol{p}_j^{\mathsf{T}}(\xi_j)\boldsymbol{\alpha}_j = \underbrace{\boldsymbol{p}_j^{\mathsf{T}}(\xi_j)\boldsymbol{A}^{-1}}_{\boldsymbol{N}_j^{\mathsf{T}}(\xi_j)}\boldsymbol{x}_j, \tag{6.16}$$

which gives the expression for the basis functions $\boldsymbol{N}_j(\xi_j)$ in each chart. Figure 6.4 plots the five basis functions in each chart, which are essentially linear combinations of B-spline basis functions and thus still C^2 continuous cubic splines. The global approximation $f(\xi)$ and basis function $\boldsymbol{N}(\xi)$ are obtained by blending together the contributions from several overlapping charts with the smooth weight functions $w_j(\xi_j)$ as follows:

$$f(\xi) = \sum_j w_j(\xi_j)f_j(\xi_j) = \sum_j w_j(\xi_j)\boldsymbol{N}_j^{\mathsf{T}}(\xi_j)\boldsymbol{x}_j = \underbrace{\left(\sum_j w_j(\xi_j)\boldsymbol{N}_j^{\mathsf{T}}(\xi_j)\boldsymbol{P}_j\right)}_{\boldsymbol{N}^{\mathsf{T}}(\xi)}\boldsymbol{x}, \tag{6.17}$$

where $\boldsymbol{P}_j$ are picking up matrices filled with ones and zeros such that

$$\boldsymbol{x}_j = \boldsymbol{P}_j\boldsymbol{x}. \tag{6.18}$$

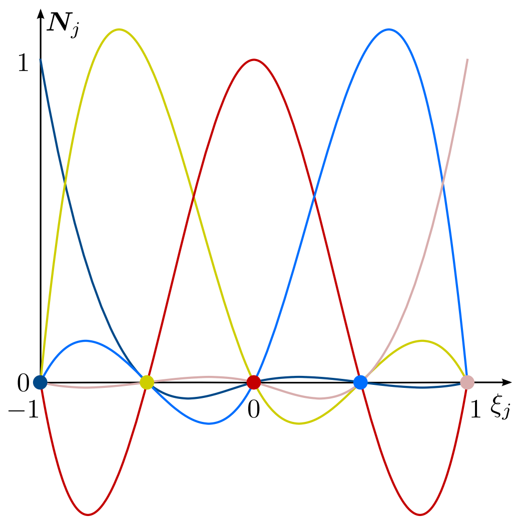


Figure 6.4: Basis functions in a manifold regular chart domain.

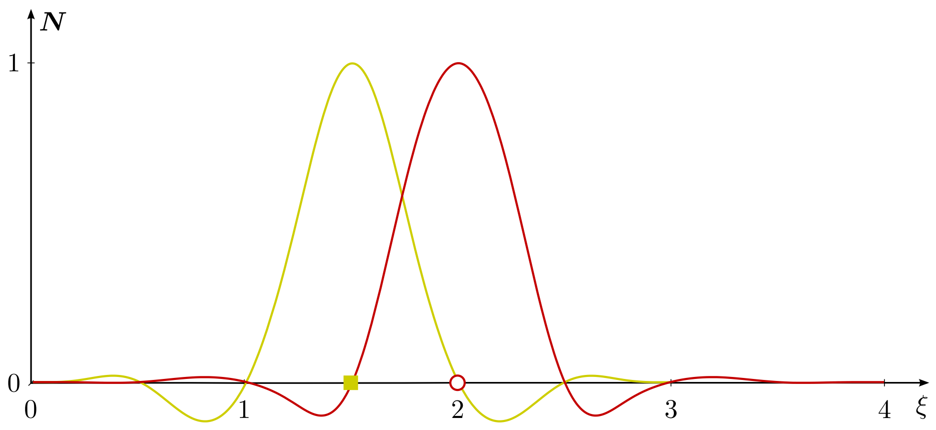


Figure 6.5: Basis functions corresponding to the vertex point (marked as red circle) and the mid-edge point (marked as yellow square), whose support size is four and three elements, respectively. Other basis functions are shifted instances of these two assuming all charts are regular.

The picking up matrices $\boldsymbol{P}_j$ essentially represent an index mapping between local control points of each chart domain and global control points. Assuming that all charts are regular, the basis functions in all charts are the same as shown in Figure 6.4. After blending the contributions from the overlapping charts, we obtain the manifold-spline basis functions plotted in Figure 6.5. The basis functions associated with vertices receive contributions from three overlapping charts and their support size is four elements. Those associated with mid-edge points receive contributions from two charts only and their support size is three elements.

With the manifold-spline basis functions in place, we can reproduce B-spline functions. For example, consider a B-spline function represented by the B-spline basis functions plotted

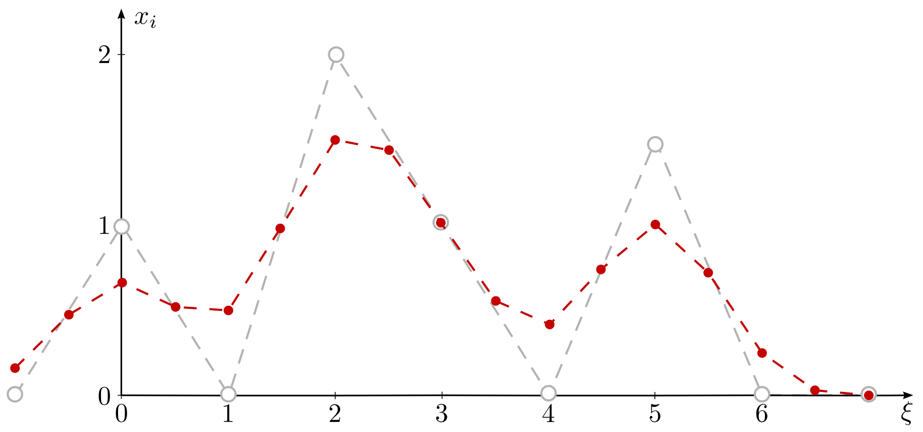


Figure 6.6: Control polygons for B-spline (grey) and manifold spline (red) representations.

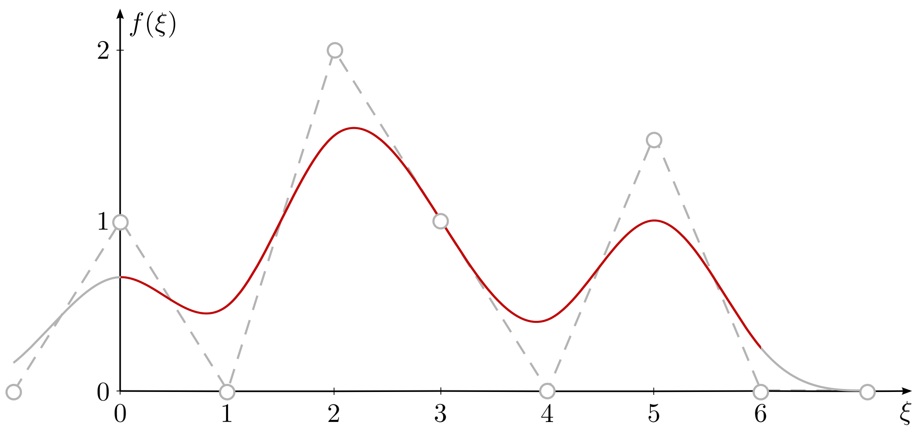
in Figure 6.1 and the control coefficients $\boldsymbol{\beta}^\mathsf{T} = [0,1,0,2,1,0,1.5,0,0]$. We can compute the corresponding control coefficients $\boldsymbol{x}$ for the manifold spline function according to (6.11) and obtain

$$\boldsymbol{x}^\mathsf{T} = \frac{1}{96}[16 \;\; 46 \;\; 64 \;\; 50 \;\; 48 \;\; 96 \;\; 144 \;\; 138 \;\; 96 \;\; 53 \;\; 40 \;\; 71 \;\; 96 \;\; 69 \;\; 24 \;\; 3 \;\; 0]\,. \tag{6.19}$$

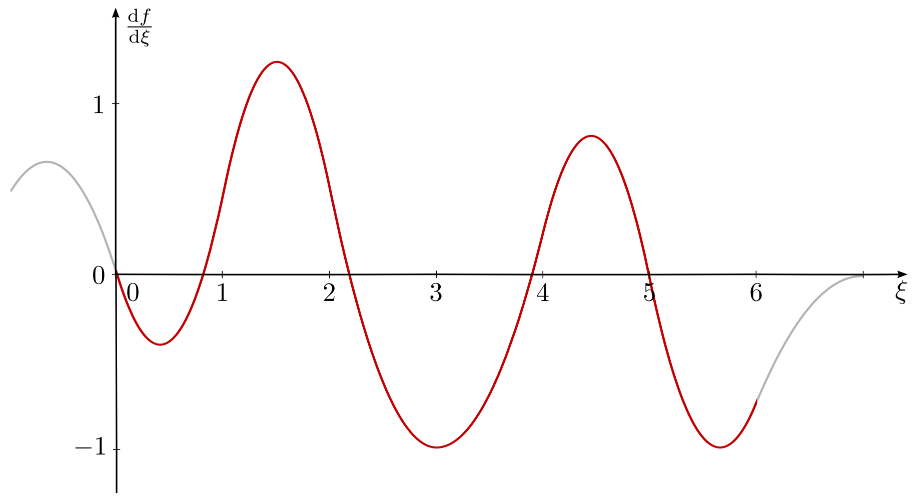
Figure 6.6 plots the two sets of control coefficients. The B-spline representation (6.1) and manifold-spline representation (6.17) both lead to exactly the same cubic spline function, see Figure 6.7.

Note that B-spline functions are reproduced, but the resulting interpolating manifold-spline basis functions plotted in Figure 6.5 are clearly different from B-spline basis in Figure 6.1. This confirms that we have derived a manifold representation for splines.

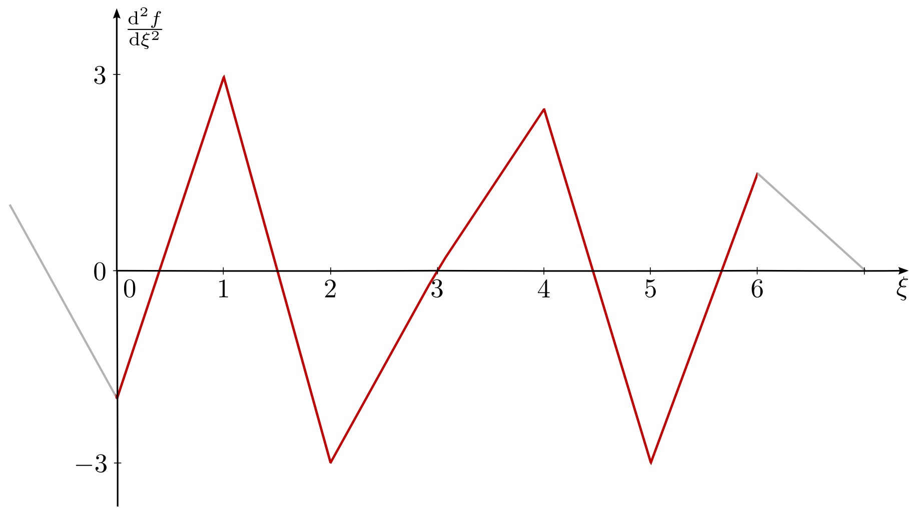
With the technique of tensor product, the demonstrated univariate derivation can be readily extended the bivariate case, where we can choose in regular charts C^2-continuous piecewise polynomials as the local basis but one piece of polynomial basis which is C^∞ in irregular charts centred at extraordinary vertices. In the bivariate case, subdivision surfaces are evaluated at the sample points used for least-squares fitting. This subdivision-compatible manifold construction differs from the manifold construction implementation reviewed in Chapter 5 only in the choice of the local basis in regular manifold chart domains. Even though the idea of choosing piecewise polynomials as the local basis can be found in [61] as well, it is a totally different construction aiming to reduce derivative magnitudes and achieve better surface quality rather than to reproduce subdivision surfaces away from extraordinary vertices.



(a) Cubic spline function



(b) First derivative



(c) Second derivative

Figure 6.7: A cubic B-spline function (grey curve) is reproduced by the manifold spline construction (red curve). The first and second derivatives further confirm the reproduction.

Chapter 7

Conclusions and future research

7.1 Conclusions

This thesis contributes to the development of basis functions for isogeometric design and analysis of surfaces on structured and unstructured quadrilateral meshes. To this end, we have investigated and further developed two different geometry representations, i.e., subdivision surfaces and manifold-based surface constructions.

For subdivision surfaces, we have proposed a new systematic approach to optimise the subdivision weights around extraordinary vertices to improve their approximation properties in finite element analysis. We first construct charateristic meshes from the eigenvectors of the subdivision matrix and use these charateristic meshes to approximate quadratic functions. The approximation error is measured in thin-shell energy norm and minimised to derive the optimal subdivision weights for finite element analysis. Two sets of optimal refinement weights can be pre-determined for two quadratic functions, cup-like and saddle-like geometries, respectively. For a given finite element problem, an initial analysis is run with the optimal weights for cup-like functions. Then the finite element solution around each extraordinary vertex is categorised as cup-dominant or saddle-dominant through a novel local shape decomposition, which decides the set of optimal weights for each extraordinary vertex independently. The proposed optimisation approach is demonstrated for Catmull-Clark subdivision scheme with a particular focus on extraordinary vertices of valence $v = 5$. The optimal weights derived can be easily incorporated into any existing implementation of the Catmull-Clark subdivision scheme. The low-cost optimisation procedure can reduce the finite element discretisation error by 50% or more as shown in our finite element analysis of thin-plates, even though the convergence rate is still the same. In fact, optimal convergence should be achievable by C^2 subdivision schemes. There is, however, a degree estimate for subdivision surfaces based on quadrilateral polynomial patches [66] saying that the minimal bi-degree for a parametrically C^2 scheme is 6. Hence, Catmull-Clark subdivision scheme, as a generalisation of bi-cubic B-splines, cannot converge optimally without applying further tricks. Advanced subdivision schemes like guided subdivision surfaces [67] that are good for geometry modelling might also be good for simulation. They might be able to achieve optimal convergence rates, but the simplicity of standard subdivision schemes is lost.

For basis functions generated by manifold constructions, we have extended previous work on smooth basis functions to manifold basis functions with arbitrary prescribed smoothness. A new quasi-conformal map is proposed to parametrise the planar manifold chart domains as well as construct piecewise local polynomial basis. The new manifold basis functions are able to model and analyse surfaces with sharp features, such as creases and corners, which greatly widens the range of applications of manifold-based basis functions in isogeometric analysis. As the treatment of boundaries is similar to C^0 creased edges, a precise control of boundaries and surface normals becomes possible with the new basis functions. In particular, we have demonstrated that hinged or rigidly joined thin-shell structures, which have wide applications in structural mechanics, can be modelled and analysed using the introduced manifold basis functions. Optimal convergence is achieved for both structured and unstructured meshes.

At the end, we have presented a novel manifold representation that is able to reproduce B-spline curves/surfaces. The driving motivation is the observation that the manifold and subdivision representations have distinct benefits as basis functions for isogeometric analysis. The manifold scheme shows an optimal convergence in finite element analysis, whereas the subdivision scheme has appealing properties for geometric modelling, such as the convex hull property and the highest continuity for a given polynomial degree. Therefore, we explore the possibility of analysing subdivision surfaces(already available in computer-aided design software) with the recently developed manifold basis functions, which requires a subdivision-compatible manifold scheme. To demonstrate the possibility, we derive in detail a univariate manifold basis, with which uniform cubic B-spline curves are exactly reproduced. With the technique of tensor product, the demonstrated univariate derivation can be readily extended to the bivariate case, where we need to choose in regular charts C^2-continuous piecewise cubic polynomials as the local basis but one piece of cubic polynomial basis which is C^∞ in irregular charts centred at extraordinary vertices. In this bivariate setting, we can expect that a Catmull-Clark subdivision surface can be easily converted to its manifold representation which exactly reproduces it in regular region, that is, tensor-product uniform cubic B-splines, and only slight differences are present in the neighbourhood of extraordinary vertices. This provides a basis for an integrated subdivision-manifold design-analysis framework.

7.2 Future research

To develop basis functions for isogeometric analysis on unstructured meshes, the subdivision-compatible manifold representation looks promising as it inherits the benefits of both subdivision and manifold surface modelling techniques. Along this line, we suggest the following directions for future research:

- One can consider the reproduction of subdivision surfaces in regular region with sharp features.
- A rigorous mathematical proof is desired for the properties, such as the approximation order and the linear independence, of the basis functions generated by manifold constructions.
- Further exploration is needed to reduce the magnitude of derivatives of the generated manifold basis functions such that the finite element discretisation leads to a better conditioned matrix.
- For different manifold constructions, the basis functions may be piecewise or rational polynomials in one element, which requires a careful choice for numerical integration.
- By considering a non-uniform parameter space, one might be able to develop a NURBS-compatible manifold representation.

Appendix A

Weight functions

The weight functions proposed in this thesis are polynomial in contrast to the rational weight functions used in [24]. The difference between the two constructions is best understood by comparing Figure 5.3 and Figure A.1. In Figure A.1 the construction process of the rational weight functions is illustrated. Notice that the cubic B-spline basis used within the reference element is not complete so that it does not add up to one. Therefore, normalisation is necessary to satisfy the partition of unity property, resulting in rational weight functions. With the numbering introduced in Figure A.1 the rational blending functions are given by

$$\widetilde{w}_j(\eta) = \frac{B_{2j}(\eta)}{\sum_{k=1}^{2} B_{2k}(\eta)} \quad \text{with} \quad w_j(\xi_j) = \widetilde{w}_j(\Psi_j^{-1}(\xi_j)) \quad \text{and} \quad j \in \{1,2\}\,.$$

It is straightforward to extend this construction to the bivariate case. As a final remark, the rational blending functions have one knot and the polynomial weight functions have three knots within the reference element.

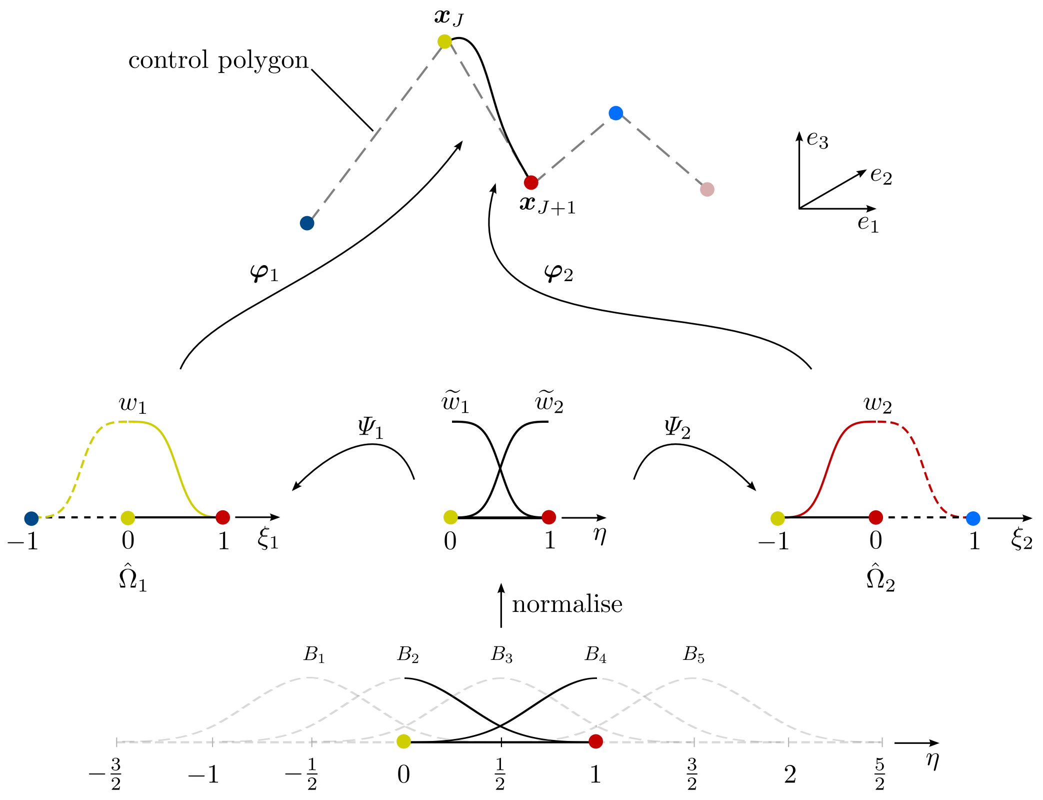


Figure A.1: Construction of smooth rational weight functions from cubic B-splines defined over a parameter space with a knot-distance $1/2$.

Appendix B

Conformal maps

In the manifold construction presented in Chapter 5, conformal maps are extensively used to parametrise manifold chart domains as well as create piecewise polynomial basis functions for the local approximation in creased charts. Conformal maps can be better understood in the complex plane. As shown in Figure B.1a, any point in the complex plane is expressed as a complex number z and it corresponds to a point in a two-dimensional Cartesian coordinate system (η_1, η_2) as follows:

$$(\eta_1, \eta_2) = (|z|\cos\phi, |z|\sin\phi) \quad \text{and} \quad |z| = \sqrt{(\eta^1)^2 + (\eta^2)^2}, \quad \phi = \arctan(\eta_2/\eta_1) \tag{B.1}$$

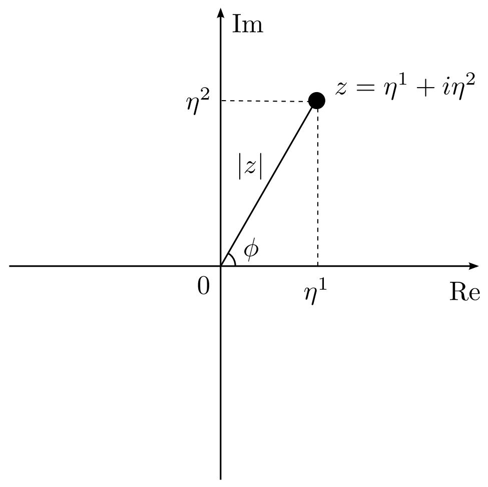
$$z = \eta^1 + i\eta^2 = |z|(\cos\phi + i\sin\phi) = |z|e^{i\phi} \tag{B.2}$$

where $|z|$ denotes the radius and ϕ the phase. Any operation leading to only the change of the radius $|z|$ is a radius scaling and only the change of the phase ϕ is a rotation. For example, Figure B.1b illustrates that $ze^{i\theta} = |z|e^{i(\phi+\theta)}$ is obtained by rotating z anticlockwise by the angle of θ, while $1/2z = |z|/2e^{i\phi}$ is obtained by scaling the radius to half length. Rotation and radius scaling can be composed and they commute.

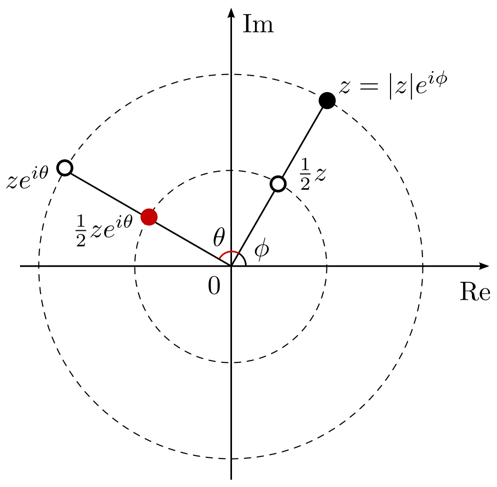
With the operation of rotation and scaling in mind, we can understand the geometry meaning of the conformal map defined as

$$z^{4/v} = |z|^{4/v}e^{i4\phi/v}, \tag{B.3}$$

where v is the valence of the centre vertex of the chart. When $v = 4$ the conformal map is an identity map. When $v \neq 4$ the angle is stretched from ϕ to $4\phi/v$ and the radius from $|z|$ to $|z|^{4/v}$. The angle is uniformly scaled but the radius is not. In the manifold construction



(a) Complex plane



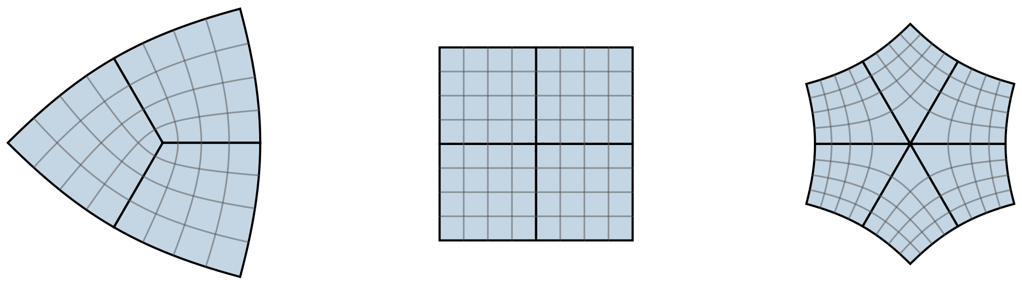
(b) Rotation and scaling in complex plane

Figure B.1: Illustration of a complex number in the complex plane and the operations of rotation and scaling.

proposed in this thesis, we use a quasi-conformal map defined as

$$|z|^{\beta-4/v} z^{4/v} = |z|^{\beta} e^{i4\phi/v}. \tag{B.4}$$

Figure B.2 illustrates the influence of the parameter β on the conformal images. The value $\beta = 4/v$ defining a conformal map is used in [24], while $\beta = 1$ defining a quasi-conformal map is used in this thesis to provide smooth parametrisations for crease charts as defined in (5.21) and (5.23).


(a) Conformal map with $\beta = 4/v$

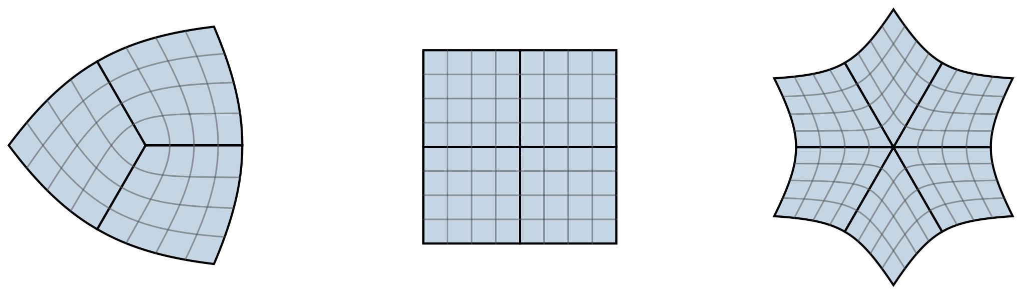

(b) Quasi-conformal map with $\beta = 1$

Figure B.2: Comparison of the iso-parameter lines for the confomal and the quasi-conformal map depending on the value of the parameter β for valences $v \in \{3,4,6\}$. See the definition in (B.4).

References

[1] A. Hrennikoff. Solution of problems of elasticity by the framework method. *Journal of Applied Mechanics*, 8(4):169–175, 1941.

[2] R. Courant. Variational methods for the solution of problems of equilibrium and vibrations. *Bulletin of the American Mathematical Society*, 49(1):1–23, 1943.

[3] P. E. Bézier. Example of an existing system in the motor industry: the Unisurf system. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 321(1545):207–218, 1971.

[4] 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(39–41):4135–4195, 2005.

[5] B. Marussig and T. J. R. Hughes. A review of trimming in isogeometric analysis: Challenges, data exchange and simulation aspects. *Archives of Computational Methods in Engineering*, 25(4):1059–1127, 2018.

[6] F. Cirak, M. Ortiz, and P. Schröder. Subdivision surfaces: a new paradigm for thin-shell finite-element analysis. *International Journal for Numerical Methods in Engineering*, 47(12):2039–2072, 2000.

[7] A. Collin, G. Sangalli, and T. Takacs. Analysis-suitable G^1 multi-patch parametrizations for C^1 isogeometric spaces. *Computer Aided Geometric Design*, 47:93–113, 2016.

[8] D. Toshniwal, H. Speleers, R. R. Hiemstra, and T. J. R. Hughes. Multi-degree smooth polar splines: A framework for geometric modeling and isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 316:1005–1061, 2017.

[9] M. Kapl, F. Buchegger, M. Bercovier, and B. Jüttler. Isogeometric analysis with geometrically continuous functions on planar multi-patch geometries. *Computer Methods in Applied Mechanics and Engineering*, 316:209–234, 2017.

[10] C. L. Chan, C. Anitescu, and T. Rabczuk. Isogeometric analysis with strong multipatch C^1-coupling. *Computer Aided Geometric Design*, 62:294–310, 2018.

[11] Q. Zhang, M. Sabin, and F. Cirak. Subdivision surfaces with isogeometric analysis adapted refinement weights. *Computer-Aided Design*, 102:104–114, 2018.

[12] M. Breitenberger, A. Apostolatos, B. Philipp, R. Wüchner, and K.-U. Bletzinger. Analysis in computer aided design: Nonlinear isogeometric B-Rep analysis of shell structures. *Computer Methods in Applied Mechanics and Engineering*, 284:401–457, 2015.

[13] Y. Guo, J. Heller, T. J. R. Hughes, M. Ruess, and D. Schillinger. Variationally consistent isogeometric analysis of trimmed thin shells at finite deformations, based on the STEP exchange format. *Computer Methods in Applied Mechanics and Engineering*, 336:39–79, 2018.

[14] E. Catmull and J. Clark. Recursively generated B-spline surfaces on arbitrary topological meshes. *Computer-Aided Design*, 10(6):350–355, 1978.

[15] D. Doo and M. Sabin. Behaviour of recursive division surfaces near extraordinary points. *Computer-Aided Design*, 10(6):356–360, 1978.

[16] U. Reif and P. Schröder. Curvature integrability of subdivision surfaces. *Advances in Computational Mathematics*, 14(2):157–174, 2001.

[17] F. Cirak and M. Ortiz. Fully C^1-conforming subdivision elements for finite deformation thin-shell analysis. *International Journal for Numerical Methods in Engineering*, 51(7):813–833, 2001.

[18] F. Cirak, M. J. Scott, E. K. Antonsson, M. Ortiz, and P. Schröder. Integrated modeling, finite-element analysis, and engineering design for thin-shell structures using subdivision. *Computer-Aided Design*, 34(2):137–148, 2002.

[19] C. M. Grimm and J. F. Hughes. Modeling surfaces of arbitrary topology using manifolds. *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques (SIGGRAPH '95)*, pages 359–368, 1995.

[20] J. C. Navau and N. P. Garcia. Modeling surfaces from meshes of arbitrary topology. *Computer Aided Geometric Design*, 17(7):643–671, 2000.

[21] L. Ying and D. Zorin. A simple manifold-based construction of surfaces of arbitrary smoothness. *ACM Transactions on Graphics (TOG)*, 23(3):271–275, 2004.

[22] G. Della Vecchia, B. Jüttler, and M.-S. Kim. A construction of rational manifold surfaces of arbitrary topology and smoothness from triangular meshes. *Computer Aided Geometric Design*, 25(9):801–815, 2008.

[23] C. V. Beccari, D. E. Gonsor, and M. Neamtu. RAGS: Rational geometric splines for surfaces of arbitrary topology. *Computer Aided Geometric Design*, 31(2):97–110, 2014.

[24] M. Majeed and F. Cirak. Isogeometric analysis using manifold-based smooth basis functions. *Computer Methods in Applied Mechanics and Engineering*, 316:547–567, 2017.

[25] G. Farin. *Curves and Surfaces for Computer-Aided Geometric Design (3rd Edition)*. Academic Press, 1993.

[26] C. de Boor. On calculating with B-splines. *Journal of Approximation Theory*, 6(1):50–62, 1972.

[27] W. Boehm. Inserting new knots into B-spline curves. *Computer-Aided Design*, 12(4):199–201, 1980.

[28] R. N. Goldman. Blossoming and knot insertion algorithms for B-spline curves. *Computer Aided Geometric Design*, 7(1–4):69–81, 1990.

[29] G. M. Chaikin. An algorithm for high-speed curve generation. *Computer Graphics and Image Processing*, 3(4):346–349, 1974.

[30] R. F. Riesenfeld. On Chaikin's algorithm. *Computer Graphics and Image Processing*, 4(3):304–310, 1975.

[31] J. M. Lane and R. F. Riesenfeld. A theoretical development for the computer generation and display of piecewise polynomial surfaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2(1):35–46, 1980.

[32] T. J Cashman, U. H. Augsdörfer, N. A. Dodgson, and M. A. Sabin. NURBS with extraordinary points: high-degree, non-uniform, rational subdivision schemes. *ACM Transactions on Graphics (TOG)*, 28(3):46:1–46:9, 2009.

[33] T. J. Cashman, N. A. Dodgson, and M. A. Sabin. Selective knot insertion for symmetric, non-uniform refine and smooth B-spline subdivision. *Computer Aided Geometric Design*, 26(4):472–479, 2009.

[34] J. Kosinka, M. Sabin, and N. Dodgson. Creases and boundary conditions for subdivision curves. *Graphical Models*, 76(5):240–251, 2014.

[35] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stuetzle. Piecewise smooth surface reconstruction. *Proceedings of the 21st annual conference on Computer graphics and interactive techniques (SIGGRAPH '94)*, pages 295–302, 1994.

[36] T. DeRose, M. Kass, and T. Truong. Subdivision surfaces in character animation. *Proceedings of the 25th annual conference on Computer graphics and interactive techniques (SIGGRAPH '98)*, pages 85–94, 1998.

[37] J. W. Barrett and C. M. Elliott. Finite element approximation of the Dirichlet problem using the boundary penalty method. *Numerische Mathematik*, 49(4):343–366, 1986.

[38] A. Embar, J. Dolbow, and I. Harari. Imposing Dirichlet boundary conditions with Nitsche's method and spline-based finite elements. *International Journal for Numerical Methods in Engineering*, 83(7):877–898, 2010.

[39] D. Schillinger, I. Harari, M.-C. Hsu, D. Kamensky, S. K.F. Stoter, Y. Yu, and Y. Zhao. The non-symmetric Nitsche method for the parameter-free imposition of weak boundary and coupling conditions in immersed finite elements. *Computer Methods in Applied Mechanics and Engineering*, 309:625–652, 2016.

[40] 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)*, pages 395–404, 1998.

[41] F. Cirak and Q. Long. Subdivision shells with exact boundary control and non-manifold geometry. *International Journal for Numerical Methods in Engineering*, 88(9):897–923, 2011.

[42] K. Bandara and F. Cirak. Isogeometric shape optimisation of shell structures using multiresolution subdivision surfaces. *Computer-Aided Design*, 95:62–71, 2018.

[43] U. Reif. A unified approach to subdivision algorithms near extraordinary vertices. *Computer Aided Geometric Design*, 12(2):153–174, 1995.

[44] J. Peters and U. Reif. Shape characterization of subdivision surfaces—basic principles. *Computer Aided Geometric Design*, 21(6):585–599, 2004.

[45] K. Karčiauskas, J. Peters, and U. Reif. Shape characterization of subdivision surfaces—case studies. *Computer Aided Geometric Design*, 21(6):601–614, 2004.

[46] M. Halstead, M. Kass, and T. DeRose. Efficient, fair interpolation using Catmull-Clark surfaces. *Proceedings of the 20th annual conference on Computer graphics and interactive techniques (SIGGRAPH '93)*, pages 35–44, 1993.

[47] L. Kobbelt. A variational approach to subdivision. *Computer Aided Geometric Design*, 13(8):743–761, 1996.

[48] U. H. Augsdörfer, N. A. Dodgson, and M. A. Sabin. Tuning subdivision by minimising Gaussian curvature variation near extraordinary vertices. *Computer Graphics Forum*, 25(3):263–272, 2006.

[49] L. Barthe and L. Kobbelt. Subdivision scheme tuning around extraordinary vertices. *Computer Aided Geometric Design*, 21:561–583, 2004.

[50] I. Ginkel and G. Umlauf. Local energy-optimizing subdivision algorithms. *Computer Aided Geometric Design*, 25(3):137–147, 2008.

[51] P. G. Ciarlet. *An introduction to differential geometry with applications to elasticity*. Springer, 2005.

[52] J. Peters and U. Reif. *Subdivision Surfaces*. Springer Series in Geometry and Computing. Springer, 2008.

[53] M. Sabin. *Analysis and Design of Univariate Subdivision Schemes*. Springer Series in Geometry and Computing. Springer, 2010.

[54] T. Nguyen, K. Karčiauskas, and J. Peters. C^1 finite elements on non-tensor-product 2d and 3d manifolds. *Applied Mathematics and Computation*, 272(1):148–158, 2016.

[55] 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*, 327:411–458, 2017.

[56] T. Nguyen and J. Peters. Refinable C^1 spline elements for irregular quad layout. *Computer Aided Geometric Design*, 43:123–130, 2016.

[57] U. Reif. TURBS—topologically unrestricted rational B-splines. *Constructive Approximation*, 14(1):57–77, 1998.

[58] M. Donatelli, P. Novara, L. Romani, S. Serra-Capizzano, and D. Sesana. Surface subdivision algorithms and structured linear algebra: a computational approach to determine bounds of extraordinary rule weights. Technical Report 2016-012, Department of Information Technology, Uppsala University, 2016.

[59] S. Timoshenko and S. Woinowsky-Krieger. *Theory of Plates and Shells*. McGraw-Hill, 1959.

[60] B. Jüttler, A. Mantzaflaris, R. Perl, and M. Rumpf. On numerical integration in isogeometric subdivision methods for PDEs on surfaces. *Computer Methods in Applied Mechanics and Engineering*, 302:131–146, 2016.

[61] E. Tosun and D. Zorin. Manifold-based surfaces with boundaries. *Computer Aided Geometric Design*, 28(1):1–22, 2011.

[62] G. Della Vecchia and B. Jüttler. Piecewise rational manifold surfaces with sharp features. *Proceedings of IMA International Conference on Mathematics of Surfaces*, pages 90–105, 2009.

[63] R. Wang, L. Liu, Z. Yang, K. Wang, W. Shan, J. Deng, and F. Chen. Construction of manifolds via compatible sparse representations. *ACM Transactions on Graphics (TOG)*, 35(2):14:1–14:10, 2016.

[64] H. Biermann, A. Levin, and D. Zorin. Piecewise smooth subdivision surfaces with normal control. *Proceedings of the 27th annual conference on Computer graphics and interactive techniques (SIGGRAPH '00)*, pages 113–120, 2000.

[65] Q. Long, P. B. Bornemann, and F. Cirak. Shear-flexible subdivision shells. *International Journal for Numerical Methods in Engineering*, 90(13):1549–1577, 2012.

[66] U. Reif. A degree estimate for subdivision surfaces of higher regularity. *Proceedings of the American Mathematical Society*, 124(7):2167–2174, 1996.

[67] 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):84–95, 2018.