Ruprecht-Karls-Universität Heidelberg
Fakultät für Mathematik und Informatik

Smoothness analysis of subdivision algorithms

Diplomarbeit im Fach Mathematik
vorgelegt von

Bastian Rieck

Betreut durch Dr. Susanne Krömker
Heidelberg, November 2010

Last built on 26th July 2011

Erklärung

Hiermit versichere ich, dass ich meine Arbeit selbstständig unter Anleitung verfasst habe, dass ich keine als die angegebenen Quellen und Hilfsmittel benutzt habe, und dass ich alle Stellen, die dem Wortlaut oder dem Sinne nach anderen Werken entlehnt sind, durch die Angabe der Quellen als Entlehnung kenntlich gemacht habe.

(Unterschrift)	(Ort, Datum)

Abstract

In computer graphics, subdivision algorithms are common tools for smoothing down irregularly shaped meshes. Of special interest, due to their simple formulations, are algorithms that generalize B-spline subdivision. Their conceptual simplicity is in stark contrast to the complexity of analysing their results. A complete formal examination of smoothness properties for subdivision schemes was only recently performed by J. Peters and U. Reif.

This thesis presents a precise and detailed introduction to the analysis of subdivision algorithms. For this purpose, first of all, the necessary background in B-spline theory is established. Building on this, two of the most common subdivision algorithms, the Doo-Sabin and the Catmull-Clark scheme, are motivated. Their treatment is followed by an in-depth description of methods for analysing smoothness properties of subdivision schemes, as developed by Peters and Reif. Afterwards, these methods are applied to the two aforementioned algorithms, thereby establishing smoothness for *both* algorithms in their original form. Last, in order to demonstrate the effects of choosing unsuitable weights, a number of *degenerate* weights, which produce irregular shapes in almost all cases, are derived for both schemes—these have hitherto not been published.

Zusammenfassung

Unterteilungsalgorithmen sind ein gebräuchliches Werkzeug der Computergrafik zur Glättung annähernd beliebig geformter Flächen. Aufgrund ihrer einfachen mathematischen Beschreibung sind jene Algorithmen, die Unterteilungsverfahren von B-splines verallgemeinern, von besonderem Interesse. Ihre konzeptionelle Einfachheit steht in krassem Gegensatz zur Komplexität der Analyse ihrer Resultate. Eine vollständige formale Untersuchung der Glattheitseigenschaften von Unterteilungsschemata wurde erst vor wenigen Jahren durch J. Peters und U. Reif durchgeführt.

Die vorliegende Diplomarbeit stellt eine präzise und detailreiche Einführung in die Analyse von Unterteilungsalgorithmen dar. Zu diesem Zweck wird zunächst der benötigte Hintergrund der Theorie der B-Splines etabliert. Darauf aufbauend werden zwei der bekanntesten Unterteilungsalgorithmen, das Doo-Sabin- und das Catmull-Clark-Schema, motiviert. Ihrer Behandlung schließt sich eine eingehende Beschreibung der durch Peters und Reif entwickelten Methoden zur Analyse der Glattheitseigenschaften von Unterteilungsalgorithmen an. Danach werden diese Methoden auf die beiden zuvorgenannten Algorithmen angewandt, wodurch die *Glattheit* für beide Algorithmen in ihrer ursprünglichen Beschreibung festgestellt wird. Um die Auswirkungen unpassender Gewichte zu veranschaulichen, werden schließlich *degenerierte* Gewichte, die in beinahe jedem Fall ungleichmäßige Formen erzeugen, für beide Schemata hergeleitet. Diese Gewichte sind bis dato noch nicht publiziert worden.

Acknowledgements

First and foremost, I would like to express my gratitude to all people who contributed directly or indirectly to this thesis. I am grateful to my supervisor Dr. Susanne Krömker who introduced me to the interesting and complex field of subdivision algorithms. I owe thanks to my colleagues at the Interdisciplinary Center for Scientific Computing (IWR) for providing a productive and social atmosphere. In particular, I thank Jens Fangerau and Julia Portl for fruitful discussions and critical comments. I am indebted to my cohort of proof readers, consisting of Jens Fangerau, Hans Rieck, Loreen Ruff, and Anja Schäfer—their corrections were indispensable and helped improve this thesis. Any errors are my fault.

Last but not least, I thank my family and friends for their encouragement.

Contents

List of Figures

1 Introduction

The subject matter of this thesis are *subdivision algorithms*[1] for 2-dimensional manifold meshes[2]. When being applied to a mesh, subdivision algorithms act as *local operators* with the purpose of performing a *smoothing process*: Sharp edges and creases will become more rounded and fair. Examples of this process are depicted in Figure 1.1 and Figure 1.2.

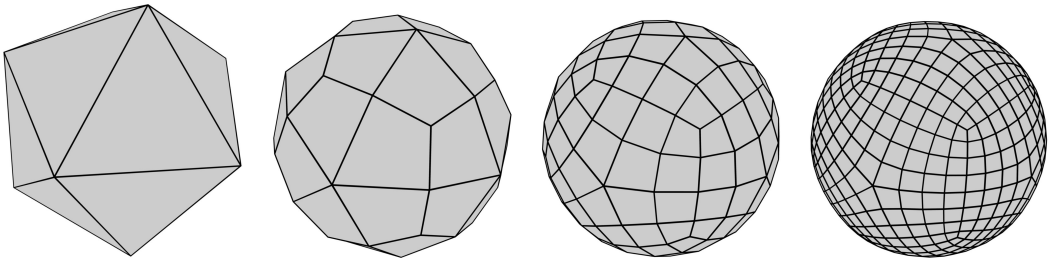


Figure 1.1: *Three steps of the Catmull-Clark subdivision algorithm*

Taking a look at the first few steps of an algorithm as depicted by Figure 1.1, we may conjecture that the algorithm will converge against some limit surface. Proving this *formally*, however, is not straightforward. Consequently, in this thesis, we will develop a framework for determining convergence properties of subdivision algorithms. Furthermore, we will see that it is possible to predict some properties of the limit surface. More precisely, we will at least be able to ascertain that the limit surface of an algorithm does not contain any self-intersections.

This chapter provides a gentle introduction into the subject matter. First, the motivation for the analysis of subdivision algorithms is explained. After this, a brief synopsis of the history of subdivision algorithm research is given. This is followed by an explanation of certain notations required for subsequent chapters. The chapter then concludes with acknowledgements and an overview of the remaining chapters. Throughout this chapter, footnotes are used to refer the reader to relevant definitions that appear only much later in the thesis.

Some closing remarks about the spelling: This thesis uses "Oxford spelling", as employed, for example, by the Oxford English Dictionary. The main characteristic of this spelling is the use of the suffix "-ize" (instead of "-ise") for words of Greek origin, whereas the suffix "-yse" (instead of "-yze") is retained.

---

[1] A geometrical definition of these algorithms is given in Chapter 3. A more formal approach is given by Definition 4.10 on page 54.

[2] We may view a *mesh* as a set of vertices in $\mathbb{R}^3$ along with a description of connectivity for the points, which yields a graph. Manifold meshes satisfy certain requirements in their local appearance, making them easier to be handled algorithmically. See Definition 3.4 on page 32 for more details.

Figure 1.2: *Original mesh (left) and mesh subdivided by three steps of the Catmull-Clark algorithm (right). The smoothing effects can be observed in particular at the transition between socket and statue. We will thoroughly analyse the Catmull-Clark scheme in Chapter 5.*

1.1 Motivation

The appeal of subdivision algorithms is obvious from Figure 1.2, for example: Even a small number of steps results in a visually smoother object. In addition to this, subdivision algorithms feature interesting properties, making them indispensable tools in geometric modelling:

Almost arbitrary topologies Subdivision algorithms may be applied to 2-dimensional manifold meshes, thereby having less restrictions than the standard spline-based methods[3].

Local operators The smoothing process is a *local* operation. It can be chosen to affect only small parts of the mesh. This enables the use of *adaptive* methods that are based on the shape of meshes.

Recursive definition Since subdivision schemes are recursive operations, they may be used as natural *level of detail* methods.

Ease of implementation Provided that appropriate data structures for storing and querying the mesh exist, subdivision algorithms can be implemented *efficiently*[4].

A precondition for the analysis of subdivision algorithms is their *convergence*[5] to a *limit surface*. In several papers, the limit surface is also referred to as *subdivision surface*. However, in order to stress that convergence is an essential property of a subdivision algorithm, this thesis prefers the term *limit surface*.

The goal of this thesis is to provide a concise analysis of properties of the limit surface. More precisely, we will be interested in the *smoothness* of limit surfaces: The desired result is to have convergent

[3] A discussion about the restrictions of spline-based methods is given in Chapter 3.

[4] A sample implementation of these algorithms is described in Chapter B of the appendix.

[5] See Definition 4.4 on page 50. This definition will become useful in a more rigorous mathematical framework, which we will construct in Chapter 4.

algorithms whose limit surface does not have any sharp edges or self-intersections[6]. Yet, this endeavour is not easy—see the following section on the history of the most important analysis attempts. Consequently, it took approximately 20 years after the first publication of subdivision algorithms for a detailed analysis to be published. This was accomplished by Peters and Reif [Rei95, Rei98, PR98, PR08] over the course of several publications. Their analysis, however, is *quite* involved and requires knowledge of several branches of mathematics, such as algebraic and differential topology, abstract algebra, complex analysis, and numerical analysis. Furthermore, due to length restrictions of the papers, some details are not mentioned or cannot be expanded on. This makes comprehension of the publications a very daunting task. Hence, this thesis was written with the following objectives in mind:

- Sufficiently establish the theoretical background in B-spline theory (upon which several popular subdivision schemes are based).
- Focus on a *detailed* description of methods required for the analysis of subdivision algorithms.
- "Fill the gaps" in proofs; expand them or rewrite them based on current knowledge.
- Provide a *thorough* derivation of results that are *known* in literature but still lack details.

1.2 History of the analysis of subdivision algorithms

In this section, we will take a look at research in subdivision algorithms through the last four decades. Many references to further publications are provided.

1970s Research in subdivision algorithms began with a paper of Chaikin [Cha74], which describes a fast algorithm for calculating smooth curves by iterated subdivision rules. By a result of Riesenfeld [Rie75], these curves were shown to be *B-splines*. In 1978, Catmull and Clark [CC78] presented a subdivision scheme that generalizes bicubic B-spline subdivision. At the same time, Doo [Doo78] described an algorithm generalizing biquadratic B-spline subdivision. The scheme was extended by Doo and Sabin [DS78], who also performed (in the same paper) the first preliminary analysis of the Catmull-Clark subdivision scheme. However, although the comments of Doo and Sabin led to some improvements of the published version of the Catmull-Clark scheme, in total, the analysis was rather cursory: In the end, their analysis could only *motivate*, but not sufficiently explain, the cause for some artefacts in the algorithm. This attempt, however, stressed the usage of the discrete Fourier transform[7] and may be rightly viewed as the basis of subsequent analyses.

The behaviour of subdivision algorithms is largely determined by the way they handle *extraordinary parts* of a mesh[8]: In their publication [CC78], Catmull and Clark noted, for example, that different ways

[6] Whether self-intersections appear depends on the input data. The algorithm is only required to ensure that no self-intersections occur when the input data are well-behaved. Later chapters will clarify these terms.

[7] Put briefly, the discrete Fourier transform (DFT) decomposes a large matrix into smaller blocks by a similarity transformation, thereby simplifying the calculation of eigenvalues and eigenvectors. The DFT will be formally introduced in Section 4.3 on page 65.

[8] In order to model a wide range of shapes, meshes cannot be regular tilings of the Euclidean plane at every point. A subdivision algorithm for *almost arbitrary topologies* needs special rules for handling these *irregular* parts. We will clarify this in Chapter 3.

of creating *vertex points*[9] may lead to surfaces that are "too pointy". They admitted that their choice of weights[10] was "somewhat arbitrary" and they surmised, although there was no proof, that "the surface is at least continuous in tangent everywhere"—a guess that would turn out to be correct.

1980s In 1988, Ball and Storry [BS88] provided the first analysis of *tangent plane continuity* for the Catmull-Clark scheme. Their work laid the foundation for *all* analysis methods that are currently in use. Moreover, Ball and Storry introduced the *natural configuration* that describes control points around vertices of certain valencies[11]. This configuration is later used by Reif's *characteristic map*[12]. In addition, Ball and Storry gave an almost correct overview of the spectrum of the subdivision matrix and created a graphical representation of limitations for the weights of the algorithm. Ultimately, though, their proof was shown by Reif [Rei95] to contain subtle errors. Furthermore, tangent plane continuity is a weak smoothness criterion that still allows self-intersections.

1990s In 1995, Reif [Rei95], building on Ball and Storry's methods, was able to derive smoothness criteria that do not allow degenerate cases. He coined the term *characteristic map*, which describes an invariant of subdivision schemes. At first, he analysed a special case of the Doo-Sabin scheme. In 1998, in a joint paper with Peters [PR98], the first correct analysis of the Doo-Sabin and the Catmull-Clark scheme was performed. Both schemes were shown to satisfy C^1-smoothness[13]. This analysis was expanded on in Reif's habilitation thesis [Rei98]. Further research of Peters and Reif produced *midedge subdivision* [PR97], which is a conceptually simple algorithm with a complicated eigenstructure—namely, the *subdominant eigenvalue*[14] does not have a multiplicity of 2.

At the same time of the analysis of Peters and Reif for algorithms generalizing B-spline subdivision, Reif's work was used to analyse different subdivision schemes. In 1996, Schweitzer [Sch96] examined Loop's subdivision scheme[15]. A complete analysis of the same scheme was performed independently by Zorin [Zor97].

2000s In 2000, Umlauf [Uml00] extended Reif's characteristic map to triangular subdivision schemes. This was followed by a general *framework* of Zorin [Zor00], which greatly simplifies the analysis of subdivision algorithms. Zorin used the framework to establish smoothness for the Butterfly[16] scheme as well as for an interpolatory scheme of Kobbelt. At this point, smoothness for all common subdivision

[9] A type of refined control point that occurs when applying the algorithm. See Section 3.4.2 on page 40.

[10] The weights are the *coefficients* for the affine combinations of old control points.

[11] The valency of a vertex is the number of faces that meet at the vertex. We will introduce this concept, along with a detailed examination of the topology of meshes, in Chapter 3.

[12] An invariant of a subdivision scheme that we will use for smoothness analysis. See Section 4.12 on page 56.

[13] A detailed account of smoothness properties is given in Chapter 4. In short, C^1-smoothness means that the limit surface is a C^1-manifold in the sense of differential topology.

[14] The algorithms in this thesis are assumed to be described by a matrix where the largest eigenvalue is 1 and the modulus of all other eigenvalues is smaller. In particular, the second and third eigenvalue are required to be equal and larger than the rest. Hence, this eigenvalue is called subdominant. See Definition 4.14 on page 63 for more details.

[15] The scheme generalizes box spline subdivision, which this thesis does not expand on. Section 3.1 contains a brief introduction to this scheme and other schemes. Furthermore, the algorithm was implemented for comparison purposes. Some examples are contained in Chapter B of the appendix.

[16] The masks of this subdivision scheme for triangular meshes resemble the shape of a butterfly. Since the scheme is *not* based on splines, smoothness properties were indeterminable prior to Zorin's framework.

schemes had been shown. Thus, in an effort to present the current state of the art in subdivision schemes, Peters and Reif [PR08] published a book that contains extensions of their theories so that a wider range of different algorithms may be encompassed.

Present day Nowadays, research is performed in order to obtain subdivision algorithms that provide C^2-smoothness, which is also denoted by *curvature continuity*. The starting point is a paper of Peters and Reif [PR04], who investigated how to characterize the *shape* of limit surfaces by analysing the *curvature*. Together with Karčiauskas [KPR04], the Catmull-Clark and Loop subdivision algorithms were examined. A further approach to better shapes employs *guided subdivision*, where the distance of the subdivided surface to a control polygon is minimized. Yet, C^2-smoothness is still an open problem. Peters and Reif [PR08] aptly stated that the limit surfaces of subdivision algorithms are "fair from afar, but far from being fair".

1.3 Notation

Care has been taken not to deviate too much from the standard mathematical notation. The *interior* of a closed curve c is denoted by $I(c)$, whereas the interior of a set U is denoted by $\mathring{U}$. For the *closure* of a set, the operator cl is used, e.g. $\mathrm{cl}(U)$.

Row vectors are denoted by small Latin letters and normal brackets, e.g. $x = (0, 1, 2)$. *Matrices* are denoted by large Latin letters. A colon is used to indicate ranges for their rows and columns, e.g. $A^{1:2,3:4}$, which refers to rows 1–2 and columns 3–4 of the matrix. *Transposed vectors* and matrices are denoted by a superscript T, e.g. x^T for a vector and A^T for a matrix. *Eigenvalues* are denoted by the letter λ, their corresponding *eigenvectors* will be denoted by ψ. Since the characteristic map, which depends on the eigenvectors, is denoted by Ψ, this notation is hoped to serve as a mnemonic help.

Matrix-vector multiplication is used in a *generalized* sense: An $n \times n$ matrix A may contain entries from $\mathbb{R}^m$. Multiplication with a vector from $\mathbb{R}^n$ is then to be understood *componentwise*. This nonstandard notation *greatly* simplifies the equations of B-spline surfaces, for example.

When dealing with complex numbers, the *imaginary unit* is written as a simple i. The *complex conjugate* of a number will be either displayed by an overline, e.g. $\overline{x} = a - ib$, or, for reasons of space and layout, by a superscript asterisk, e.g. A^*.

Further notations, in particular for the discrete Fourier transform, will be mentioned in the respective chapters.

1.4 Chapter overview

Chapter 2 starts with a basic introduction to B-spline theory. It covers the most important properties of B-spline basis functions and explains how to use these basis functions to obtain B-spline curves and B-spline surfaces. Contrary to most literature, the chapter motivates *matrix representations* of B-spline surface patches. We shall require these matrices for some of the derivations in Chapter 3. The chapter closes with some remarks about Bézier splines. These splines may be viewed as special forms of B-splines and will play a central role in Chapter 5.

Chapter 3 concisely treats *subdivision algorithms*. The chapter begins with a brief historical development of several subdivision algorithms. This display then leads to subdivision methods for B-spline surfaces. This is followed by a short discussion about the misused term "arbitrary topology", which is unfortunately much too prevalent in literature. Afterwards, the Doo-Sabin and the Catmull-Clark subdivision schemes will be derived from B-spline subdivision methods. Both schemes are described in great detail. For the Catmull-Clark scheme a variant requiring only *three* parameters, which may be used for "tuning" the algorithm, is introduced. Following this, the reader's attention is called to some interesting properties of the schemes.

Chapter 4 presents suitable methods for the *analysis* of subdivision algorithms. A test problem that is sufficient for determining the *smoothness* of limit surfaces of subdivision algorithms is presented in precise mathematical terms. A special function, the *characteristic map* Ψ, which is an invariant of the respective subdivision algorithm, is introduced. The properties of Ψ are analysed at length, and it turns out that the algorithm yields smooth results if Ψ is regular[17] and injective. The rest of Chapter 4 has the purpose of finding criteria determining regularity and injectivity. The main result of the last part is a criterion concerning the signs of partial derivatives of a certain function. This criterion will be used for the subsequent analysis of subdivision algorithms.

In Chapter 5, all methods of the preceding chapters are put to use. Here, the smoothness of the Doo-Sabin and the Catmull-Clark scheme is analysed. For their original weights[18], the limit surfaces of both schemes turn out to satisfy C^1-smoothness. Following this result, *degenerate* weights for both schemes are introduced. Using material from Chapter 4, the degenerate weights are proven to yield degenerate surfaces. Comparisons between the schemes with original weights and with degenerate weights demonstrate the effects of erroneously chosen weights. As a consequence, the range of *permissible* weights, i.e. weights that do *not* yield degenerate surfaces, is explored. For the Doo-Sabin scheme, a result of Peters and Reif [PR98] is cited. For the Catmull-Clark scheme, conditions for the weights are derived. The chapter ends with a graphical representation of permissible weights for the Catmull-Clark algorithm.

1.5 Results

Chapter 5 presents a clear and almost[19] exhaustive examination of the Doo-Sabin and the Catmull-Clark scheme. Computational results have been derived *independently* of the referenced papers. Comparison of the author's results with established results revealed several errors in the publications.[20] In particular, this thesis contains the correct derivatives of the characteristic maps for both subdivision schemes, as well as an accurate matrix representation of the Catmull-Clark scheme. Moreover, the thesis contains some results whose proofs were only sketched or omitted due to length restrictions in the original publications.

[17] This condition means that the Jacobian determinant of the map is nonzero. A precise formulation of the map is given by Definition 4.12 on page 56.

[18] See Chapter 3, pages 39 and 44, or the respective sections in Chapter 5.

[19] A special case for the Doo-Sabin scheme has not been considered. The result is cited from Peters and Reif [PR98] instead.

[20] Since most publications presented very advanced research results, which had not yet been verified by other sources, these errors are to be expected.

An example for this is the claim that the original weights[21] for both algorithms lead to smooth surfaces. This claim requires some eigenvalue estimates[22] for the Catmull-Clark scheme. In addition, this thesis presents *degenerate* weights for both algorithms in order to illustrate the effects of violations of the theory introduced in Chapter 4. Hitherto, these weights had not been examined. They are hoped to yield more insights into the workings of subdivision algorithms.

The thesis is rounded with an *open-source implementation* of the Doo-Sabin, the Catmull-Clark, and the Loop subdivision algorithm, thereby offering the possibility to compare the algorithms and their modi operandi using the same program.

1.6 Further acknowledgements

Although the results of Chapter 5 have been derived *independently* by the author, the thesis is inspired by many works of many authors. They deserve to be named not only in the bibliography. Thus, the following paragraphs attempt to highlight their contributions.

Proofs The sources of the proofs have been acknowledged in the respective chapters. *All* proofs have been extended in order to be more accessible. Often, the notation has been altered to merge proofs from different sources. The section about permissible weights for the Catmull-Clark scheme has been motivated by a technical report of Zorin [Zor98].

Figures All figures were created by the author. They have been inspired by the works of Reif [Rei95, Rei98], Peters and Reif [PR97, PR98, PR08], and Zorin [Zor98].

Mesh data The meshes of platonic solids are provided by courtesy of John Burkardt of Florida State University. The gargoyle model is provided courtesy of Bruno Lévy and Raphaëlle Chaine by the AIM@SHAPE Shape Repository. The dragon and bunny meshes are taken from the Stanford University Computer Graphics Laboratory.

[21] See Section 5.2.3 and Section 5.3.4 for their exact formulæ.

[22] The calculations are included in Section A.2 of the appendix.

2 B-splines and B-spline surfaces

A common problem in computer-aided design consists of drawing smooth curves that either *interpolate* or *approximate* a given shape. Spline-based methods are one of the most successful approaches in this area. They use piecewise polynomial functions in conjunction with a *control polygon* through which the resulting shape of the curve can be manipulated easily. In contrast to methods based on high-order polynomials, piecewise polynomial functions have the advantage of numerical stability and do not tend to oscillate. Furthermore, we will see that spline-based methods allow *efficient local shape manipulation*.

In this chapter, we will focus on B-spline curves and B-spline surfaces because several of the well-established subdivision schemes, such as the algorithms developed by Catmull and Clark or Doo and Sabin, employ B-spline surfaces as their underlying settings. Consequently, a basic knowledge of B-spline theory aids in understanding details of the analysis of a specific subdivision scheme. Readers with a working knowledge of B-splines may skim this chapter or even skip it completely.

In the following sections, we will touch only briefly on the most important theoretical aspects. Detailed accounts may be obtained from the books of Farin [Far96] and Yamaguchi [Yam88]. These books present spline methods and the required theory in a more universal context. Readers interested in more practical aspects of B-splines are referred to the books of Salomon [Salo5] or Piegl and Tiller [PT96] for more information about this very rich branch of mathematics.

2.1 B-spline basis functions

Spline methods are used in order to approximate smooth curves. To facilitate manipulation of these approximations, piecewise polynomial basis functions are defined. These basis functions may then be used as coefficients for sums of vectors of $\mathbb{R}^2$ and $\mathbb{R}^3$, thereby defining smooth curves and surfaces. The reason for using polynomials is that they are easily manageable, and furthermore, their analytical properties are well known.

This section will follow the approach of Cox, de Boor, and Mansfield [dB72]. They use a recurrence formula in order to define the basis functions. There are other methods to define B-spline basis functions, such as repeated convolution, on which the section will *not* expand.

Definition 2.1 (B-spline basis functions)**.** Let $T = \{t_0, t_1, \ldots, t_m\}$ be an increasing sequence of real numbers. We will refer to the t_i as *knots* and to T as the *knot vector*. An interval $[t_i, t_{i+1})$, which is allowed to be empty, will be called *knot span*. The ith B-spline basis function of degree p is then defined

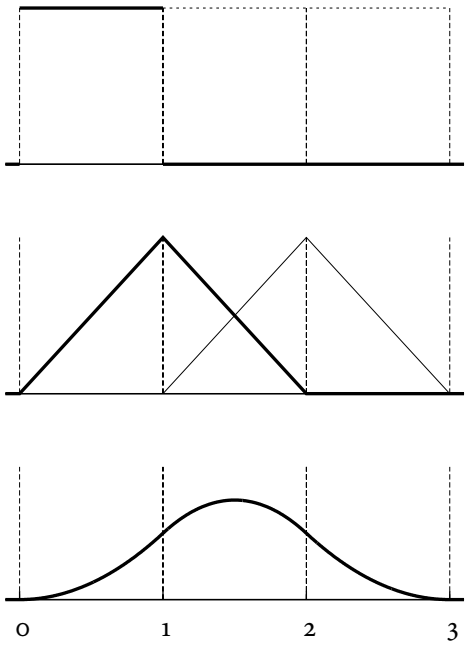


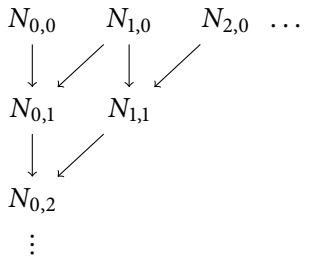
Figure 2.1: *An example for the B-spline recurrence formula. The knot vector is* $T = (0, 1, 2, 3)$ *and, from top to bottom, the B-spline basis functions of order* 0, 1, *and* 2 *are shown. The dashed lines indicate the knot spans.*

recursively as:

$$N_{i,0}(t) = \begin{cases} 1 & \text{if } t_i \le t < t_{i+1} \\ 0 & \text{else} \end{cases}$$
$$N_{i,p}(t) = \frac{t - t_i}{t_{i+p} - t_i} N_{i,p-1}(t) + \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} N_{i+1,p-1}(t) \tag{2.1}$$

Since the knots are *not* required to be distinct, a quotient $0/0$ may appear in Equation 2.1. We define this quotient as 0, which is justified because it does not make sense to define a function over an empty interval.

For $p = 0$, every basis function is the characteristic function of its corresponding knot span. For $p > 0$, basis functions are combinations of basis functions with lower degree. Figure 2.1 shows an example for the recurrence formula. We can also visualize the recursive computation of B-spline basis functions by using a table:



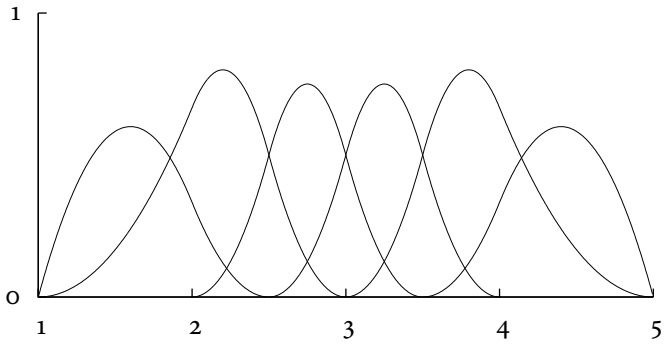


Figure 2.2: *The B-spline basis functions $N_{1,2}$, $N_{2,2}$, $N_{3,2}$, $N_{4,2}$, $N_{5,2}$, and $N_{6,2}$ for the knot vector $T = (1, 1, 2, 2.5, 3, 3.5, 4, 5, 5)$. We can observe the scalings of the different basis functions. The two middle functions, $N_{3,2}$ and $N_{4,2}$, for example, are progressing differently than the other basis functions.*

2.1.1 Properties

The properties of the B-spline basis functions make them attractive mathematical tools: They offer *local support*, *linear independence*, and form a *partition of unity*. Later on, we will assume the same properties in a more general setting for subdivision algorithms.

This section only provides proofs for the most relevant properties. We will skip the more technical and tedious proofs as they do not provide any deeper insight.

Lemma 2.1. *$N_{i,p}(t)$, which is the ith B-spline basis function of degree p, is* combined *from the basis functions $N_{i,0}(t), \ldots, N_{i+p,0}(t)$.*

Proof. The proof is by induction over p. The claim holds by definition for $p = 0$. Let $p > 0$: From Equation 2.1, we see that $N_{i,p}(t)$ is combined from $N_{i,p-1}(t)$ and $N_{i+1,p-1}(t)$. By the induction hypothesis, $N_{i,p-1}(t)$ is combined from the basis functions $N_{i,0}(t), \ldots, N_{i+p-1,0}(t)$ and likewise, $N_{i+1,p-1}(t)$ is combined from the basis functions $N_{i+1,0}(t), \ldots, N_{i+1+p-1,0}(t)$. ∎

The next proposition is the basis for most of the other properties. It makes the B-spline basis functions very desirable for modelling. We will see that B-spline curves, for example, are only affected *locally* by changes in their control polygon. The same applies to B-spline surfaces and ultimately, as we will see, to the results of subdivision algorithms.

Proposition 2.1 (Local support)**.** *The support of any B-spline basis function does not extend over the whole interval, but only over a small part of the knot vector. More precisely,*

$$N_{i,p}(t) = 0 \text{ if } t \notin [t_i, t_{i+p+1}).$$

Proof. Lemma 2.1 states that the ith basis function is combined from the basis functions $N_{i,0}(t), \ldots, N_{i+p,0}$. Using Equation 2.1, we see that its support must be in the interval $[t_i, t_{i+p+1})$, as claimed. ∎

Figure 2.2 shows basis functions of degree 2. We can see the effect of the local support property—it limits the number of functions that are nonzero for a given knot vector. It is possible to state this observation more precisely:

Lemma 2.2. *For any knot span* $[t_j, t_{j+1})$*, at most* $p+1$ *of the* $N_{i,p}(t)$ *are nonzero, namely, the basis functions* $N_{j-p,p}(t), \ldots, N_{j,p}(t)$.

Proof. By Proposition 2.1, the support of $N_{j-p,p}(t)$ is in $[t_{j-p}, t_{j+1})$ and the support of basis functions with lower indices does *not* coincide with this knot span. Likewise, the support of $N_{j,p}(t)$ is in $[t_j, t_{j+p+1})$ and the support of basis functions with higher indices does *not* coincide with this knot span. The argument holds for the basis functions with indices between $j-p$ and p. ∎

Lemma 2.3 (Nonnegativity)**.** *Basis splines are* nonnegative*:*

$$N_{i,p}(t) \geq 0 \text{ for all } i, p, t.$$

Proof. The proof is by induction. The claim holds for $p = 0$ because the characteristic function of an interval is nonnegative. For $p > 0$, we have by Definition 2.1

$$N_{i,p}(t) = \frac{t - t_i}{t_{i+p} - t_i} N_{i,p-1}(t) + \frac{t_{i+p+1} - t}{t_{i+p+1} - t_{i+1}} N_{i+1,p-1}(t).$$

Proposition 2.1 states that $N_{i,p-1}(t) = 0$ for $t \notin [t_i, t_{i+p})$. For $t \in [t_i, t_{i+p})$, we see that

$$\frac{t - t_i}{t_{i+p} - t_i} \geq 0.$$

Since $N_{i,p-1}(t) \geq 0$ holds by the induction hypothesis, the first part of the equation above is nonnegative. For the second part, the analogous argument holds. ∎

Proposition 2.2 (Partition of unity)**.** *For any knot span* $[t_i, t_{i+1})$*, the basis functions of a fixed degree form a partition of unity:*

$$\sum_{j=i-p}^{i} N_{j,p}(t) = 1 \text{ for all } t \in [t_i, t_{i+1})$$

Proof. By definition,

$$\sum_{j=i-p}^{i} N_{j,p}(t) = \sum_{j=i-p}^{i} \frac{t - t_j}{t_{j+p} - t_j} N_{j,p-1}(t) + \sum_{j=i-p}^{i} \frac{t_{j+p+1} - t}{t_{j+p+1} - t_{j+1}} N_{j+1,p-1}(t)$$

According to Proposition 2.1, $N_{i-p,p-1}(t) = 0$ and $N_{i+1,p-1}(t) = 0$. This enables us to sum from $j = i-p+1$ to $j = i$ and change the index in the second sum from $N_{j+1,p-1}(t)$ to $N_{j,p-1}(t)$ so that we arrive at:

$$\begin{aligned}\sum_{j=i-p}^{i} N_{j,p}(t) &= \sum_{j=i-p+1}^{i} \underbrace{\left(\frac{t - t_j}{t_{j+p} - t_j} + \frac{t_{j+p} - t}{t_{j+p} - t_j} \right)}_{=1} N_{j,p-1}(t) \\ &= \sum_{j=i-p+1}^{i} N_{j,p-1}(t)\end{aligned}$$

By inserting the definition of $N_{j,p-1}(t)$, we can again change the summation indices. Doing this recursively, we get:

$$\sum_{j=i-p}^{i} N_{j,p}(t) = \sum_{j=i-p+1}^{i} N_{j,p-1}(t) = \sum_{j=i-p+2}^{i} N_{j,p-2}(t)$$
$$= \cdots = \sum_{j=i}^{i} N_{j,0}(t) = 1$$

■

This property will prove central later on. We will be able to prove invariance of the B-spline curve under affine transformations, as well as a statement about the graphical progression of the B-spline curve—it will turn out that the control points impose a limit on the curve. Furthermore, it will allow us to prove that a subdivision algorithm is well-defined and convergent.

Lemma 2.4 (Derivatives of the B-spline basis functions)**.** *The derivative of the ith B-spline basis function of degree p is given by:*

$$\frac{d}{dt}N_{i,p} = \frac{p}{t_{i+p} - t_i}N_{i,p-1}(t) - \frac{p}{t_{i+p+1} - t_{i+1}}N_{i+1,p-1}(t)$$

Proof. See Piegl and Tiller [PT96], pp. 59–61. ■

This lemma proves another interesting property about B-spline basis functions: The derivative of a basis function of degree p can be expressed in terms of basis functions of degree $p-1$. Later, when B-spline curves are introduced, this lemma enables us to see the derivative of a B-spline curve as a B-spline curve of lower degree.

So far, we did not take the knot vector into account. It turns out that by modifying the knot vector, the differentiability properties and the shape of B-spline basis functions can be regulated. The lemma will enable us to prove similar properties for B-spline *curves*.

Lemma 2.5. *Derivatives of $N_{i,p}$ exist in the interior of a knot span. At a knot of multiplicity k, the basis function $N_{i,p}(t)$ is $p-k$ times* continuously differentiable.

Proof. Due to the local support property from Proposition 2.1, we only need to check the differentiability in the interval $[t_i, t_{i+p+1})$. Using Lemma 2.4, we see that

$$\frac{d}{dt}N_{i,p} = \frac{p}{t_{i+p} - t_i}N_{i,p-1}(t) - \frac{p}{t_{i+p+1} - t_{i+1}}N_{i+1,p-1}(t)$$

and assume that the lower-degree basis functions are differentiable. The fractions

$$\frac{p}{t_{i+p} - t_i} \text{ and } \frac{p}{t_{i+p+1} - t_{i+1}}$$

are *not* well-defined anymore once $t_{i+p} = t_i$ or $t_{i+p+1} = t_{i+1}$. Thus, for a knot of multiplicity k, only $p-k$ derivatives exist. ■

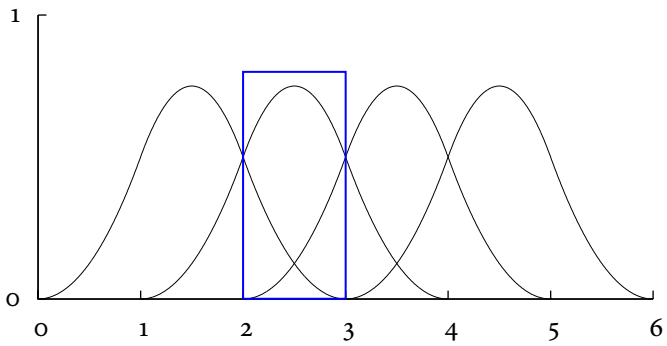


Figure 2.3: *B-spline basis functions $N_{1,2}, \ldots, N_{4,2}$ for the knot vector $T = (0, 1, \ldots, 6)$. We can observe the most important properties. The functions have* local support *and are* nonnegative. *Note how the functions form a* partition of unity *inside the blue rectangle. Furthermore, we can see that in comparison to Figure 2.2, the B-spline basis functions are shifted copies of each other. We will later prove this formally.*

The lemma above can also be used to show that the B-spline curve concept *works* in the sense that the curve defined by control points satisfies the desired continuity constraints.

From the last two properties, we see how the degree of the basis functions and the knot multiplicity are interlocked:

$$\text{increasing the} \left\{ \begin{array}{ll} \text{degree} & \text{increases} \\ \text{knot multiplicity} & \text{decreases} \end{array} \right\} \text{the continuity}$$

We will conclude this section by citing the most important property of the B-spline basis functions. Put briefly, the basis functions form a basis of the vector space of all piecewise polynomial functions of degree p that satisfy certain continuity constraints at the knots—following the previous discussion of knot multiplicities, these continuity constraints can be satisfied by the B-spline basis functions.

Theorem 2.1 (Linear independence)**.** *Let T be a* strictly increasing *knot vector. Then the B-spline basis functions of degree p are linearly independent, i.e.*

$$\sum_{i=0}^{n} \lambda_i N_{i,p}(t) = 0$$

for $\lambda_i \in \mathbb{R}$ if and only if $\lambda_i = 0$ for all i.

Proof. See Farin [Far96], p. 152. ∎

Figure 2.3 depicts all properties that were listed in this section. Note that the knot vector of this figure yields a certain symmetry—we will examine this property in the next section.

2.1.2 Uniform B-splines

Throughout this thesis, only knot vectors with *equidistant knots* need to be used. These knot vectors are called *uniform*. They yield simpler formulæ and allow us to concentrate on more important details.

Let a uniform knot vector be given, i.e. $t_i - t_{i-1} = c$ for all i and a fixed $c \in \mathbb{R}$. Without loss of generality (see the remarks concluding this section), we assume that the knots are given by $t_i = i$. Definition 2.1 is

now equivalent to

$$N_{i,p}(t) = \frac{t-i}{p} N_{i,p-1}(t) + \frac{i+p+1-t}{p} N_{i+1,p-1}(t), \tag{2.2}$$

which leads to the following conclusion:

Lemma 2.6. *Uniform B-splines are shifted copies of each other, i.e.*

$$N_{i,p}(t) = N_{0,p}(t-i).$$

Proof. We prove this by induction: For $p = 0$, the lemma is true because characteristic functions of shifted intervals are shifted copies of each other. For $p > 0$ and i fixed, we have by definition:

$$N_{i,p}(t) = \frac{t-i}{p} N_{i,p-1}(t) + \frac{i+p+1-t}{p} N_{i+1,p-1}(t)$$

Using the induction hypothesis, the terms are replaced by their copies:

$$\begin{aligned} N_{i,p}(t) &= \frac{t-i}{p} N_{0,p-1}(t-i) + \frac{i+p+1-t}{p} N_{0,p-1}(t-i-1) \\ &= N_{0,p}(t-i) \end{aligned}$$

■

The use of the "default" uniform knot vector with $t_i = i$ as introduced above may be justified by the following considerations: A uniform knot vector can be written in the form $t_i = \lambda i + \mu$ with λ and $\mu \in \mathbb{R}$. Substitution in Equation 2.1 then shows that *only* the parameter domain of the B-spline basis functions is changed by using different uniform knot vectors—the shape remains unchanged.

2.2 B-spline curves

We have now described the B-spline basis functions and several of their most important properties. In this section, we combine the basis functions to describe a variety of shapes. This leads to *B-spline curves*. The general idea behind these curves is to define a weighted sum of control points. In this sum, the scalar basis functions serve as the weights, whereas the control points are usually taken to be vectors in $\mathbb{R}^2$ or $\mathbb{R}^3$.

Definition 2.2 (B-spline curve)**.** Given a knot vector $T = (t_0, t_1, \ldots, t_k)$, a B-spline curve of degree p is defined by

$$C(t) = \sum_{i=0}^{n} N_{i,p}(t) P_i \tag{2.3}$$

with $a := t_0 \le t \le b := t_k$. The $P_i \in \mathbb{R}^3$ are the *control points* and $N_{i,p}(t)$ are the B-spline basis functions as defined earlier.

The effect of the control polygon is illustrated in Figure 2.4. It depicts a uniform B-spline curve and the behaviour of the B-spline curve near the control polygon.

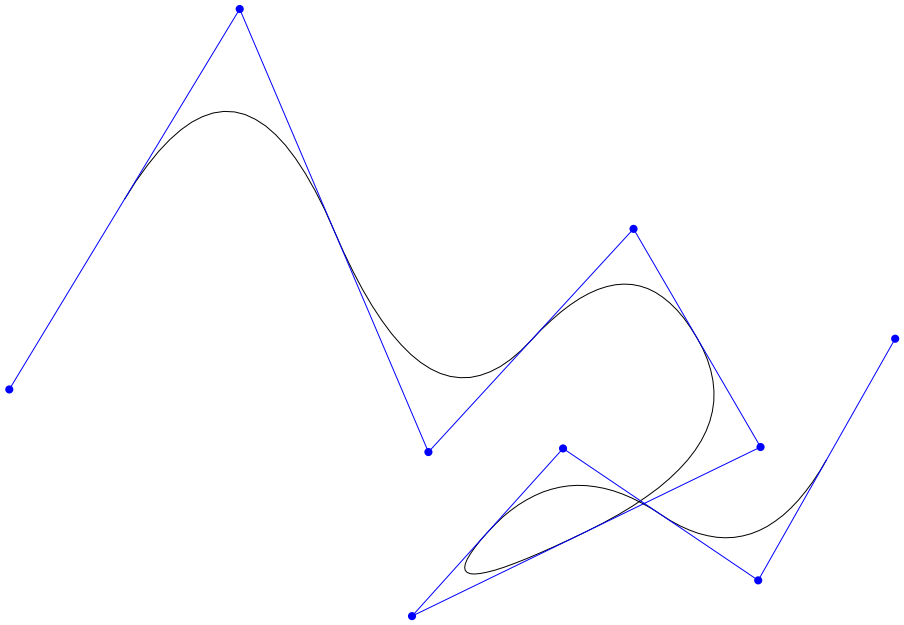


Figure 2.4: *A B-spline curve and its control polygon. For this example, a uniform knot vector has been used. Note how the B-spline curve is "pulled towards" the control polygon, while* not *interpolating it.*

2.2.1 Properties

In this section, we examine the properties of B-spline curves. Since B-spline curves use combinations of B-spline basis functions, we may expect that some of the properties for the basis functions also hold for B-spline curves. This is indeed the case—even more, properties of the B-spline basis functions give rise to important properties of B-spline curves. We will see that B-spline curves are *affine invariant* and satisfy the *convex hull property*.

Proposition 2.3 (Affine invariance)**.** *Affine transformations are applied to the B-spline curve by applying them to the control points.*

Proof. Let $\phi : \mathbb{R}^3 \to \mathbb{R}^3$ be an affine transformation, i.e. $\phi(x) = Ax + b$ for a real-valued 3×3 matrix A and a vector $b \in \mathbb{R}^3$. For a fixed value of t, let

$$x_t := \sum_{i=0}^{n} N_{i,p}(t) P_i.$$

Applying ϕ yields:

$$\begin{aligned} \phi(x_t) &= \phi\left(\sum_{i=0}^{n} N_{i,p}(t) P_i\right) \\ &= A\left(\sum_{i=0}^{n} N_{i,p}(t) P_i\right) + b \end{aligned}$$

Using Proposition 2.2, we see that $\sum_{i=0}^{n} N_{i,p}(t) = 1$, so the equation can be rewritten as:

$$\phi(x_t) = A\left(\sum_{i=0}^{n} N_{i,p}(t) P_i\right) + \sum_{i=0}^{n} N_{i,p}(t) b$$

$$= \sum_{i=0}^{n} N_{i,p}(t)(AP_i + b) \qquad \text{(by linearity)}$$
$$= \sum_{i=0}^{n} N_{i,p}(t)\phi(P_i)$$

Thus, an affine transformation applied to the curve needs only be applied to the control points. ∎

Affine invariance of a B-spline curve is the basis for almost all transformations in computer graphics: The invariance proves that *only* the control points are affected by affine transformations. Intuitively, this is the natural outcome—it implies that the B-spline curve simply approximates *any* control polygon.

Another statement about the shape of B-spline curves is made by the *strong convex hull property*. This property limits the distance of the B-spline curve to its control polygon.

Proposition 2.4 (Strong convex hull property)**.** *For $t \in [t_i, t_{i+1})$ and $p \le i \le k - p - 1$, the B-spline curve $C(t)$ is contained in the* convex hull *of the control points $P_{i-p}, \dots, P_i$. Consequently, the curve is also contained in the convex hull of all control points. This weaker statement is known as the* convex hull property.

Proof. Let $t \in [t_i, t_{i+1})$ be fixed. Using Lemma 2.2, we sum only the nonzero basis functions. The B-spline curve is then given by:

$$C(t) = \sum_{j=i-p}^{i} N_{j,p}(t)P_j$$

The basis functions form a partition of unity by Proposition 2.2, so $\sum_{j=i-p}^{i} N_{j,p}(t) = 1$, and furthermore, the basis functions are *nonnegative* by Lemma 2.3. Thus, $C(t)$ is a *convex combination* of the control points $P_{i-p}, \dots, P_i$ and by definition contained within the convex hull of these control points. ∎

Figure 2.5 depicts the convex hull property. Furthermore, it turns out that modifications of one control point do *not* affect the whole curve.

Lemma 2.7 (Local modification)**.** *If a control point P_i is moved, the B-spline curve $C(t)$ is only changed in the interval $[t_i, t_{i+p+1})$. As a consequence, all modifications are* local.

Proof. By Proposition 2.1, $N_{i,p}(t) = 0$ for $t \notin [t_i, t_{i+p+1})$. ∎

The previous lemma provides another advantage of B-spline-based methods: Contrary to other attempts for the interpolation of points, such as the naïve power basis approach, shape manipulation of B-spline curves does not require a recomputation of the whole curve.

2.3 B-spline surfaces

This section introduces B-spline *surfaces*. There are many ways to represent surfaces—the most common approach uses *tensor product surfaces*. We may think of these types of surfaces as being spanned by families of curves. As basis functions for these surfaces, we use products of univariate B-spline basis functions. The degrees of those basis functions are allowed to be different. This thesis only requires surfaces of the *same* degree in both directions, but the general definition and properties are nonetheless given below.

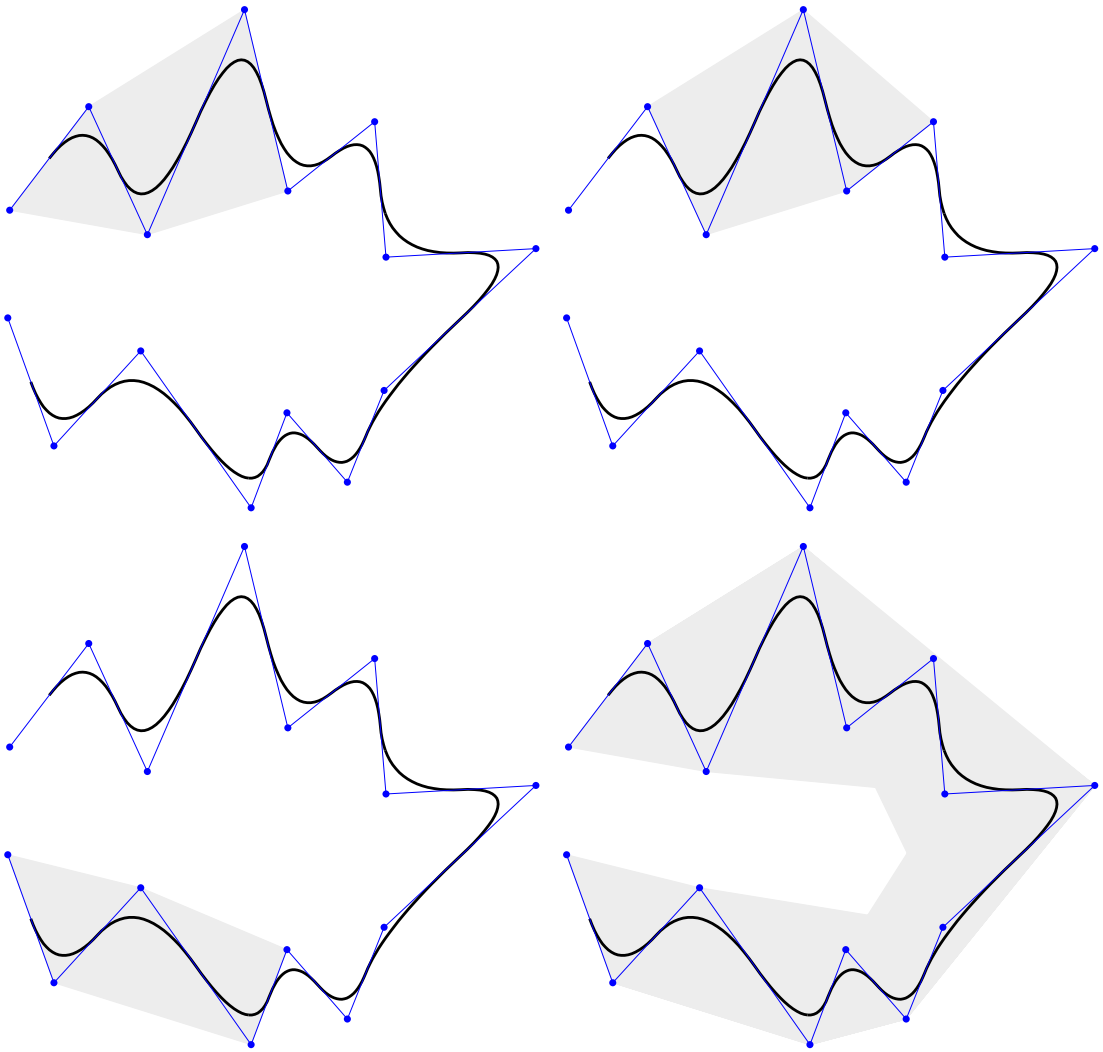


Figure 2.5: *A B-spline curve of degree* 4 *along with its control polygon. The convex hulls of the first, the second, and the last set of control points are shown. The picture on the lower right shows the union of the convex hulls for the respective sets of control points. Note that the convex hulls overlap as a consequence of Proposition 2.4.*

Definition 2.3 (B-spline surface)**.** Let the control points P_{ij} be arranged in a matrix. Let U and V be knot vectors with values in $[0,1]$. A B-spline surface of degree (p,q) is then defined as

$$S(u,v) = \sum_{i=0}^{n}\sum_{j=0}^{m} N_{i,p}(u)N_{j,q}(v)P_{ij}, \tag{2.4}$$

where $N_{i,p}$ and $N_{j,q}$ are the basis B-spline functions of degrees p and q.

2.3.1 Properties

In this section, we will prove some properties already known for B-spline curves, namely *affine invariance* and the *convex hull property*. Furthermore, the products of basis functions will turn out to have similar properties than the B-spline basis functions.

Proposition 2.5 (Partition of unity)**.** *The factors $N_{i,p}(u)N_{j,q}(v)$ form a* partition of unity*:*

$$\sum_{i=0}^{n} N_{i,p}(u)\sum_{j=0}^{m} N_{j,q}(v) = 1 \quad \textit{for all } (u,v) \in [0,1]^2$$

Proof. We prove this by applying Proposition 2.2 for each of the factors:

$$\begin{aligned}\sum_{i=0}^{n} N_{i,p}(u)\sum_{j=0}^{m} N_{j,q}(v) &= \sum_{i=0}^{n} N_{i,p}(u)\left(\underbrace{\sum_{j=0}^{m} N_{j,q}(v)}_{=1}\right)\\ &= \sum_{i=0}^{n} N_{i,p}(u)\\ &= 1\end{aligned}$$

■

Lemma 2.8 (Nonnegativity)**.** $N_{i,p}(u)N_{j,q}(v) \geq 0$ *for all i, j, p, q, u, v.*

Proof. This follows from the nonnegativity of the basis functions, which has been proven in Lemma 2.3. ■

Lemma 2.9 (Local support)**.** $N_{i,p}(u)N_{j,q}(v) = 0$ *if (u,v) is not inside the rectangle*

$$[u_i, u_{i+p+1}) \times [v_j, v_{j+q+1}).$$

Proof. The interval $[u_i, u_{i+p+1})$ can be seen as the union of the knot spans

$$[u_i, u_{i+1}), [u_{i+1}, u_{i+2}), \ldots, [u_{i+p}, u_{i+p+1}).$$

Using Lemma 2.2, we see that $N_{i,p}(u) = 0$ if u is not in any of these knot spans. An analogous statement holds for $N_{j,q}(v)$. Since the two factors are multiplied, the product vanishes whenever one of the values is outside the topological rectangle defined by the Cartesian product of the knot spans. ■

Using the previous lemma, we may partition a larger B-spline surface into smaller *patches*. As a corollary, we immediately see that, as in the univariate case, changes to the control points do *not* affect the whole surface:

Corollary 2.1 (Local modification)**.** *If a control point P_{ij} is moved, all the changes to the structure of the B-spline surface are contained inside the rectangle $[u_i, u_{i+p+1}) \times [v_j, v_{j+q+1})$.*

And again, the control polygon constrains the shape of a B-spline surface. This property will prove *central* for the analysis of subdivision algorithms.

Proposition 2.6 (Strong convex hull property)**.** *If the parameters (u, v) are contained inside a rectangle, namely,*

$$(u, v) \in [u_i, u_{i+1}) \times [v_j, v_{j+1}),$$

for some valid indices i, j, then $S(u, v)$ is contained in the convex hull *of the control points $P_{i'j'}$, where $i - p \le i' \le i$ and $j - q \le j' \le j$.*

Proof. By Lemma 2.9, we see that only the factors concerning the control points mentioned above are nonzero. As in the univariate case, the B-spline surface can be viewed as a linear combination of the control points for fixed parameters. Since the factors form a partition of unity by Lemma 2.5, the linear combination of the control points is an *affine combination*. Last, because the factors are nonnegative by Lemma 2.8, we see that the affine combination is a *convex combination* of the control points $P_{i'j'}$, with i' and j' as above. ∎

We have now the means to prove that the B-spline surface formulation as defined above provides us with a surface that has the desired continuity properties and can nonetheless be controlled and manipulated very easily:

Lemma 2.10. *At a knot of multiplicity k, the B-spline surface $S(u, v)$ is $p-k$ times continuously differentiable in u-direction and $q - k$ times continuously differentiable in v-direction.*

Proof. This is a consequence of the differentiability properties of the B-spline basis functions. We have proven them in Lemma 2.5. ∎

The lemma shows why setting $p = q$ makes sense: In most applications, derivatives of the same order should exist in *all directions*. To conclude the list of properties for B-spline curves, we prove that the surfaces are *affine invariant*. Again, this property will be relevant for the analysis of subdivision algorithms.

Proposition 2.7 (Affine invariance)**.** *An affine transformation is applied to the surface by applying it to the control points.*

Proof. The main argument requires Proposition 2.5. The rest of the proof is completely analogous to the proof of Proposition 2.3. ∎

2.4 Matrix representations

While brief and elegant, the recursive definition for the B-spline basis functions cannot be easily used for subdivision schemes because it requires many function evaluations. The following section describes a representation of *uniform* B-spline curves and B-spline surfaces by considering matrices. In later chapters, we will use these matrices to represent a subdivision scheme as a *linear map*, which is more approachable than recursively defined functions.

2.4.1 B-spline curves

Given a vector of control points, we can define a B-spline curve of degree p by using Equation 2.3. If t is inside any knot span, i.e. $t \in [t_i, t_{i+1})$, we know from Proposition 2.1 that the B-spline curve inside this knot span is given by

$$C_i(t) = \sum_{j=i-p}^{i} N_{j,p}(t) P_j.$$

This equation describes the ith *segment* of the B-spline curve $C(t)$. We can rewrite it to obtain

$$C_i(t) = \begin{pmatrix} N_{i-p}, \dots, N_i \end{pmatrix} \begin{pmatrix} P_{i-p} \\ P_{i-p+1} \\ \cdots \\ P_i \end{pmatrix},$$

which describes the segment as a multiplication of a matrix and a vector. The following theorem shows how to replace the B-spline basis functions by monomials.

Theorem 2.2 (Matrix representation of a B-spline curve segment)**.** *The uniform B-spline segment of degree p can be written as*

$$C_i(t) = (t^p, t^{p-1}, \dots, t, 1) M_p \begin{pmatrix} P_{i-p} \\ P_{i-p+1} \\ \cdots \\ P_i \end{pmatrix}, \tag{2.5}$$

where M_p is a $(p+1) \times (p+1)$ matrix with entries

$$m_{ij} = \frac{1}{p!} \binom{p}{i} \sum_{k=j}^{p} (p-k)^i (-1)^{k-j} \binom{p+1}{k-j}. \tag{2.6}$$

Proof. See Piegl and Tiller [PT96], pp. 265–271, or Yamaguchi [Yam88], pp. 327–329, for proofs. The basic idea is to express the B-spline curve as a piecewise combination of Bézier spline segments (Bézier splines are a specialization of B-spline—we will briefly introduce them in the next section). A change of basis is then performed so that these segments are expressed as *power basis functions*. After a reparametrization, the matrix expression from above is obtained. ∎

We can use the previous theorem to represent *B-spline curves*. To this end, we gather $p+1$ adjacent control points in a vector and define the curve segment-wise. In this thesis, however, we shall require matrices M_2 and M_3 only. Calculations according to Theorem 2.2 then yield

$$M_2 = \frac{1}{2}\begin{pmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{pmatrix} \quad \text{and} \quad M_3 = \frac{1}{6}\begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{pmatrix}. \tag{2.7}$$

2.4.2 B-spline surfaces

In order to find a matrix representation of a B-spline surface of degree (p,q), we can apply the reasoning from the previous section. For (u,v) fixed, with $(u,v) \in [u_i, u_{i+1}) \times [v_j, v_{j+1})$, we can start from Equation 2.4. Using Lemma 2.9, we know that a B-spline surface patch can be expressed by

$$S_{i,j}(u,v) = \sum_{i'=i-p}^{i} \sum_{j'=j-q}^{j} N_{i',p}(u) N_{j',q}(v) P_{i'j'},$$

which we can again rewrite in a more telling form, namely,

$$S_{i,j}(u,v) = \begin{pmatrix} N_{i-p}, \dots, N_i \end{pmatrix} P \begin{pmatrix} N_{j-q} \\ N_{j-q+1} \\ \dots \\ N_j \end{pmatrix},$$

where P is the matrix of a subset of control points that are relevant for the patch, i.e.

$$P = \left(P_{i'j'}\right)_{\substack{i-p \le i' \le i \\ j-q \le j' \le j}}.$$

From Theorem 2.2, we can derive the corresponding theorem for B-spline surface patches.

Theorem 2.3 (Matrix representation of a B-spline surface patch)**.** *The uniform B-spline surface patch of degree (p,q) can be written as*

$$\begin{aligned} S_{i,j}(u,v) &= \begin{pmatrix} u^p, u^{p-1}, \dots, u, 1 \end{pmatrix} M_p P M_q^T \begin{pmatrix} v^q \\ v^{q-1} \\ \dots \\ v \\ 1 \end{pmatrix} \\ &= U M_p P M_q^T V, \end{aligned}$$

where M_p and M_q are matrices as defined in Theorem 2.2.

Proof. The proof is analogous to the proof of Theorem 2.2: After expressing the B-spline surface patch as a Bézier spline surface patch, the B-spline basis functions can be changed into power basis functions, which will then yield the expression from above. ∎

2.5 Bézier splines

For the analysis of subdivision schemes, we shall require a *specialization* of the B-spline approach. The following sections provide a *tour de force* through the theory of Bézier curves and Bézier surfaces. We are only interested in some treats the theory offers and refer the reader to Farin [Far96, FHK02] for further details.

2.5.1 Definition and properties

Bézier curves and surfaces constitute another class of spline-based methods. Instead of the B-spline basis functions, they use *Bernstein polynomials*. Apart from that, their properties are similar to those of B-spline curves and surfaces. In fact, it turns out that Bézier splines may be viewed as a special case of B-splines.

More precisely, we take a knot vector of the form $T = (0, \dots, 0, 1, \dots, 1)$ with $2 \cdot (p+1)$ knots, and require the control polygon to have $p+1$ control points. Via this procedure, the B-spline curves become *Bézier curves* and the B-spline surfaces become *Bézier surfaces*. In particular, *all* the properties we have proved in this chapter still apply. We are especially interested in a reformulation of Proposition 2.6:

Proposition 2.8 (Convex hull property for Bézier surfaces)**.** *The Bézier surface of degree* (p, p) *is contained in the convex hull of its control points.*

Proof. This is exactly the same claim from Proposition 2.6. ∎

Furthermore, we will require *derivatives* of Bézier patches. Just as in the case of B-spline patches, derivatives of Bézier patches are Bézier patches of lower degree. We are solely interested in the *control points* of these derived patches and have the following proposition:

Proposition 2.9 (First-order partial derivatives of Bézier patches)**.** *The control points of first-order partial derivatives of a Bézier patch are calculated by* forward *differences. For the partial derivative in u-direction, the new control points are calculated as*

$$P'_{i,j} = P_{i+1,j} - P_{i,j}. \tag{2.8}$$

Likewise, for the partial derivative in v-direction, the new control points are

$$P'_{i,j} = P_{i,j+1} - P_{i,j}. \tag{2.9}$$

Proof. See Farin [Far96], pp. 241–242, or [FHK02], pp. 83–84. ∎

2.5.2 Matrix representations

In analogy to the matrix expressions of uniform B-spline patches, matrix expressions for Bézier patches exist. Again, we have basis matrices defined by the following equation.

Proposition 2.10 (Bézier basis matrix of degree p)**.** *The entries of the Bézier basis matrix of degree* p *are defined by*

$$n_{ij} = (-1)^{p+j-i} \binom{p}{j} \binom{p-j}{i},$$

where $i, j = 0, 1, \ldots, p$. We will refer to this matrix by N_p.

Proof. See Farin [Far96], pp. 59–60. ∎

For the analysis of subdivision schemes, we shall require only the basis matrices of degree 2 and 3. Proposition 2.10 yields

$$N_2 = \begin{pmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 0 & 0 \end{pmatrix} \quad \text{and} \quad N_3 = \begin{pmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}. \tag{2.10}$$

The basis matrices can be used to obtain a matrix expression of a Bézier surface patch. More precisely, we have the following theorem, which is reminiscent of the analogous theorem for B-spline patches.

Theorem 2.4. *A Bézier surface patch of degree (p, q) can be written as*

$$\begin{aligned} S_{i,j}(u, v) &= \left(u^p, u^{p-1}, \ldots, u, 1\right) N_p P N_q^T \begin{pmatrix} v^q \\ v^{q-1} \\ \ldots \\ v \\ 1 \end{pmatrix} \\ &= U N_p P N_q^T V, \end{aligned}$$

where N_p and N_q are basis matrices of degree p and q as defined by Proposition 2.10.

Proof. See Farin [Far96], pp. 248–249. ∎

2.5.3 Conversion between Bézier and B-spline control points

We now have expressions for both Bézier and B-spline surface patches. The B-spline concept, however, cannot produce "more" curves than the Bézier concept; see Böhm [Böh81] for an elaboration on this fact. Consequently, given a B-spline surface patch, it must be possible to obtain a Bézier patch that describes the *same* surface. In fact, we can easily express a given B-spline surface patch in terms of Bézier control points by equating the expressions from Theorem 2.3 and Theorem 2.4. Starting with

$$U M_p P_{\text{B-spline}} M_q^T V \stackrel{!}{=} U N_p P_{\text{Bézier}} N_q^t V,$$

this yields

$$P_{\text{Bézier}} \stackrel{!}{=} N_p^{-1} M_p P_{\text{B-spline}} M_q^T (N_q^T)^{-1}.$$

As an example, the biquadratic B-spline surface patch with control points P_{ij} can be expressed as a Bézier surface with control points P'_{ij}, where

$$P'_{ij} = \begin{pmatrix} \frac{P_{11}+P_{10}+P_{01}+P_{00}}{4} & \frac{P_{11}+P_{01}}{2} & \frac{P_{12}+P_{11}+P_{02}+P_{01}}{4} \\ \frac{P_{11}+P_{10}}{2} & P_{11} & \frac{P_{12}+P_{11}}{2} \\ \frac{P_{21}+P_{20}+P_{11}+P_{10}}{4} & \frac{P_{21}+P_{11}}{2} & \frac{P_{22}+P_{21}+P_{12}+P_{11}}{4} \end{pmatrix}. \tag{2.11}$$

We will use this conversion for the analysis of subdivision schemes: By converting a certain B-spline surface patch into a corresponding Bézier patch, we will be able to prove that the components of all control points are positive. Combined with the convex hull property of Bézier patches, this implies that the components of all points within the patch are also *positive*. From this simple fact, we will be able to deduce that a subdivision scheme creates C^1-continuous limit surfaces.

2.6 Summary

This chapter introduced B-splines and some of their most relevant properties. We saw the advantages of using them for geometric modelling, such as *affine invariance* and the *convex hull property*. After introducing a *matrix representation* of B-spline patches, we defined Bézier splines as a special case of B-splines. We analysed their properties and discussed how to represent them by matrices. It turned out that the matrix representation of Bézier splines allows a quick calculation of the control points defining the first-order partial derivatives. We finished the discussion of B-splines and Bézier splines by demonstrating a way how to convert B-spline patches to Bézier patches, and vice versa. In the following chapter, we will use the matrix representation to define a standard subdivision process for B-spline surface patches. Furthermore, we will see how to generalize this subdivision process for non-spline settings.

3 Subdivision schemes

In this chapter, we will derive *smoothing algorithms* that operate on *meshes* (for now, we may think of a mesh as a graph with vertices from $\mathbb{R}^n$). The algorithms use affine combinations of control points to calculate a set of *refined* control points. In each step, more refined control points are generated. Therefore, these algorithms are called *subdivision algorithms* or *subdivision schemes* (we will use these terms interchangeably).

The appeal of subdivision algorithms is that they provide easy means for transforming a *coarse* mesh into a *fine* and *smooth mesh*. In this chapter, we will thoroughly describe the Doo-Sabin and the Catmull-Clark subdivision algorithms; both schemes are a *generalization* of subdivision schemes for B-spline surfaces. Consequently, we will start with a description of biquadratic and bicubic B-spline surface subdivision and work towards inferring the formulæ of the Doo-Sabin and Catmull-Clark schemes. We will also perform a rudimentary analysis of some properties of those schemes. This analysis will serve as a preparation for a more rigorous treatment of both algorithms in Chapters 4 and 5.

3.1 Historical development

The first subdivision algorithm was introduced by Chaikin [Cha74]. Chaikin's approach was purely geometrical and did not make use of B-spline theory. The task of the algorithm was to *rapidly* generate curves in $\mathbb{R}^2$ and $\mathbb{R}^3$ by making maximum use of the hardware of this time. Riesenfeld [Rie75] proved that Chaikin's algorithm produces quadratic B-splines. Most authors of modern computer graphics text books, such as Farin [Far96] or Salomon [Salo5], describe a modified variant of Chaikin's algorithm that highlights the relationship to B-spline curves. Here, the existence of a control polygon P with control points P_i is assumed. A new control polygon P' with control points P'_i is then obtained from the old control points P_i by calculating

$$\begin{aligned} P'_{2i} &= \frac{3}{4}P_i + \frac{1}{4}P_{i+1} \\ P'_{2i+1} &= \frac{1}{4}P_i + \frac{3}{4}P_{i+1} \end{aligned} \tag{3.1}$$

and drawing line segments between P'_{2i} and P'_{2i+1}. Figure 3.1 depicts an example of this process. The idea of Chaikin's algorithm can be extended to B-spline curves of any degree. By splitting the parametric domain of the curve, expressions in the form of Equation 3.1 may be obtained, albeit with more control points. Subdivision of a bicubic B-spline curve, for example, results in equations that use three old control points to define one new control point.

In 1978, the two earliest subdivision methods for surfaces were introduced. The starting point, based on previous research by Catmull [Cat74], was an algorithm of Catmull and Clark [CC78] that general-

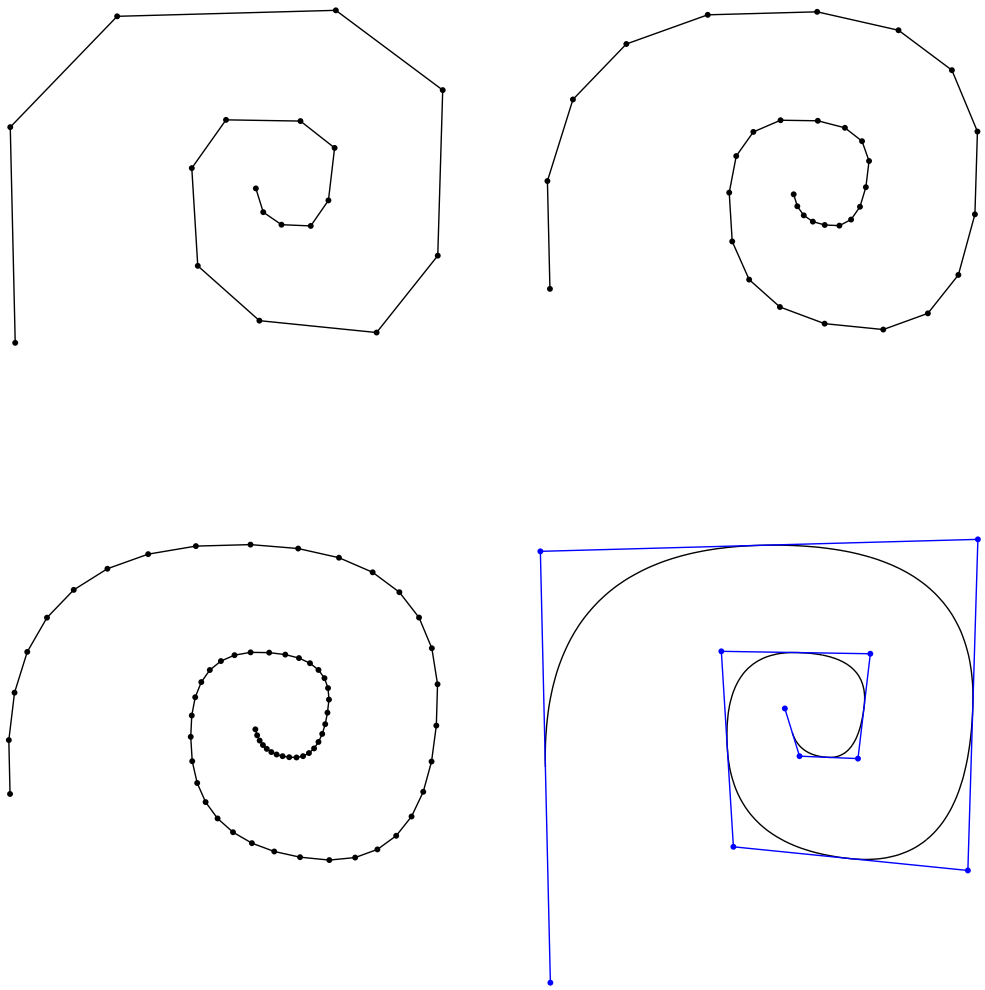


Figure 3.1: *Three steps of Chaikin's algorithm. Note how new control points are inserted in every step. The figure on the lower right shows the corresponding quadratic B-spline curve, which the algorithm converges against, along with its control polygon.*

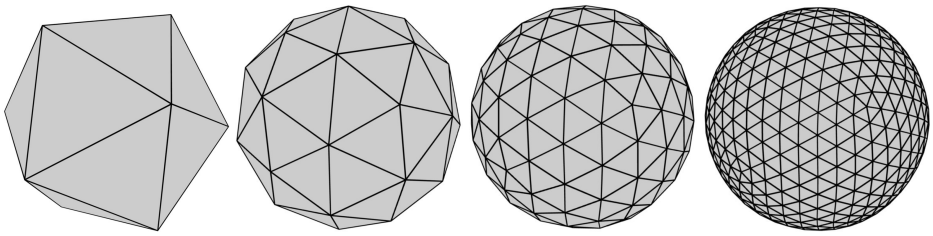


Figure 3.2: *Three steps of Loop subdivision applied to the mesh of an icosahedron. The first picture shows the initial mesh. Note how fast the individual faces shrink.*

izes *bicubic* B-spline surface subdivision. In the same year, Doo [Doo78] published an algorithm that generalizes *biquadratic* subdivision. This algorithm was subsequently analysed and extended by Doo and Sabin [DS78]. Throughout the years, more methods for developing subdivision algorithms were discovered. In 1987, Loop [Loo87] introduced a subdivision algorithm that generalizes subdivision of box splines over triangular control polygons—see Figure 3.2 for an example; a detailed derivation of this algorithm may be obtained from Portl [Por10]. Moreover, by extending an interpolatory scheme, Dyn, Levin, and Gregory [DLG90] presented the Butterfly subdivision scheme. Recently, the non-polynomial $\sqrt{3}$ subdivision scheme, which is based on topological splits, was established by Kobbelt [Kob00].

3.2 Preliminary definitions

In contrast to the previous algebraical approach to B-spline and B-spline surfaces, the following sections will be rather geometrically-flavoured. Accordingly, we shall require definitions concerning the geometrical structure of patches and surfaces.

Definition 3.1 (Mesh)**.** A *mesh* is a graph similar to the control polygon of a B-spline surface. The graph is constructed from a set of control points in $\mathbb{R}^3$ along with a description of their connectivity: The control points are the *vertices*, connections between adjacent control points are the *edges*, and each region that is enclosed by a loop of edges (which are not allowed to reach into the region) is a *face*. Contrary to the usual definition in graph theory, a mesh does not have *unbounded* faces.

A mesh is the most basic object encountered when discussing subdivision schemes: The schemes operate on meshes and make use of adjacency relations of vertices and edges. Consequently, these relations need to be classified. As a first step, different types of vertices are distinguished.

Definition 3.2 (Valency of a vertex)**.** The number of edges incident on a vertex is called its *valency*.

We may view the valency as an indicator of the *regularity* of meshes. A regular tiling of the plane using quadrangles, for example, has a valency of 4 everywhere. In most cases, the following rule holds: The more vertices with high valency, the less regular a mesh can be. This justifies the next definition.

Definition 3.3 (Ordinary and extraordinary vertices)**.** In a quadrangular tiling, a vertex with valency $k = 4$ is called *ordinary* vertex. If $k \neq 4$, the vertex is called *extraordinary*.

However, among the many flavours of meshes, some are distinguished. These are the *manifold meshes*. We may think of these meshes as looking locally like the Euclidean space.

Definition 3.4 (Manifold mesh of dimension n)**.** A mesh M is called *manifold mesh* of dimension n if for each point $x \in M$, there is a neighbourhood $U \subseteq M$ such that U is homeomorphic to $\mathbb{R}^n$.

The subdivision schemes in this thesis operate on 2-dimensional manifold meshes. In the following sections, we shall require only one consequence of this definition: Since a neighbourbood homeomorphic to $\mathbb{R}^2$ must exist, an edge in the interior of the mesh is shared by exactly two faces.

Definition 3.5 (B-spline topology)**.** If the vertices of a mesh can be arranged in a matrix, we may think of them as control points P_{ij} of a B-spline curve or a B-spline surface. By connecting control points with subsequent indices, the usual *control polygon* is formed. If such an arrangement is possible, we say that the mesh exhibits *B-spline topology*.

B-spline topology is quite restrictive: There is, for example, no possibility to model surfaces of arbitrary genus without splitting up the surface into several patches that need to be connected suitably. Moreover, the mesh cannot contain vertices of arbitrary valencies or faces with $k \neq 4$ sides. Overcoming these restrictions was the main incentive of early researchers of subdivision algorithms.

For the description of subdivision algorithms, a very intuitive approach emerged recently. It consists of using special masks, the *subdivision scheme stencils*. Unfortunately, stencils are not defined consistently in literature. For the purpose of this thesis, the following definition will prove useful.

Definition 3.6 (Subdivision scheme stencil)**.** A *subdivision scheme stencil* is a mask that can be applied to all parts of a mesh that match the topology of the stencil. Calculating the linear combination of vertices according to the weights of the stencil yields one new point for the subdivided mesh. Subdivision scheme stencils are *unique* up to rotations and reflections of control points.

Subdivision scheme stencils can be applied to *curves* and *surfaces*. As a simple example, we may derive the subdivision stencils for Chaikin's algorithm. By Equation 3.1, control points are either weighted with 3/4 or 1/4. This yields stencils as depicted by Figure 3.3.

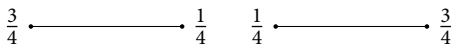


Figure 3.3: *Stencils for Chaikin's algorithm. Note that one stencil is sufficient in order to describe the algorithm because the topological situation remains the same.*

The appeal of stencils may not be obvious at first glance. Indeed, in the case of curves, the topology of the refined control points is fully defined by their indices—there are no ambiguities. However, the more complex subdivision algorithms for surfaces also yield more complex subdivision stencils; these stencils help improve the understanding of a given scheme. Nonetheless, this does *not* spare us from describing the new topology of the surface: Regrettably, stencils only describe the refinement rules for a *single* refined control point and not how to connect new control points.

3.3 B-spline surface subdivision

In this section, we will study subdivision techniques for B-spline surfaces. These techniques allow us to gain insights about subdivision algorithms for topologies different from the usual B-spline topology. Ultimately, we will be able to describe the Doo-Sabin and the Catmull-Clark subdivision algorithms.

The idea of subdividing the control polygon of B-spline curves in order to obtain the curve itself can be extended to B-spline surfaces. To this end, we need to split the parametrical domain $I = [0,1]^2$ over which a surface patch is defined. By performing the necessary calculations, we will obtain control polygons corresponding to *smaller* B-spline patches. Taking the limit, these control polygons will eventually converge to the B-spline patch that is defined by the initial control polygon.

3.3.1 Subdividing a biquadratic B-spline surface patch

It is sufficient to perform the subidivision for a biquadratic B-spline surface *patch* only. A general procedure can be derived from the resulting equations. Hence, we have a 3×3 matrix P that holds the control points P_{ij} defining the control polygon:

$$P = \begin{pmatrix} P_{00} & P_{01} & P_{02} \\ P_{10} & P_{11} & P_{12} \\ P_{20} & P_{21} & P_{22} \end{pmatrix}$$

Following the ideas of Chaikin's algorithm, we will split the quadrangular domain $I = [0,1]^2$ into four smaller congruent quadrangles. Splitting the domain corresponds to splitting the patch into four smaller patches. We will see that some of the new control points coincide so that these patches are described by 16 control points in total.

The parametrical domain of the B-spline surface patch is symmetrical, thus we only need to consider the subpatch $S(u,v)$ where $u,v \in [0,1/2]$. The equations for the other subpatches will not yield new insights. Let the new surface be defined as $S'(u,v)$. We use the matrix representation of the biquadratic B-spline patch as described by Theorem 2.3:

$$\begin{aligned} S'(u,v) &= S\left(\frac{u}{2}, \frac{v}{2}\right) \\ S'(u,v) &= \left(\frac{u^2}{4}, \frac{u}{2}, 1\right) M_2 P M_2^T \begin{pmatrix} v^2/4 \\ v/2 \\ 1 \end{pmatrix} \\ &= \left(u^2, u, 1\right) \underbrace{\begin{pmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix}}_{M'} M_2 P M_2^T \begin{pmatrix} \frac{1}{4} & 0 & 0 \\ 0 & \frac{1}{2} & 0 \\ 0 & 0 & 1 \end{pmatrix}^T \begin{pmatrix} v^2 \\ v \\ 1 \end{pmatrix} \end{aligned}$$

We now slip in auxiliary terms that simplify the equation:

$$\begin{aligned} S'(u,v) &= U M_2 M_2^{-1} M' M_2 P M_2^T M'^T (M_2^{-1})^T M_2^T V^T \\ &= U M_2 (M_2^{-1} M' M_2) P (M_2^T M'^T (M_2^{-1})^T) M_2^T V^T \\ &= U M_2 S P S^T M_2^T V^T \\ &= U M_2 P' M_2^T V^T, \end{aligned}$$

where $S = M_2^{-1} M' M_2$ and $P' = SPS^T$.

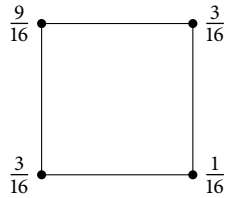


Figure 3.4: *Stencil for subdividing a biquadratic B-spline patch. The remaining three stencils have been omitted for symmetry reasons.*

Hence, the new patch can be written as $S'(u,v) = UM_2P'M_2^TV^T$. This equation, however, describes a *biquadratic B-spline patch* with a refined set of control points. In order to derive subdivision rules from this reparametrization, we solve $P' = SPS^T$ and obtain, for example,

$$P'_{00} = \frac{1}{16}\left(9P_{00} + 3P_{10} + 3P_{01} + P_{11}\right). \tag{3.2}$$

The same coefficients appear in the equations of all new points. We may interpret this equation geometrically: During the subdivision process, four points of any face in the control polygon are combined using normalized weights of $(9,3,3,1)$. This linear combination allows the calculation of *one* refined control point.

Equation 3.2 also allows us to determine the subdivision stencils. Since the weights of the equation are the same for all points and since subdivision stencils are unique up to rotations and reflections of control points, we only have *one* subdivision scheme stencil. It is depicted by Figure 3.4 and completely describes the algorithm.

Generalizing this scheme to *arbitrary* B-spline surface control polygons is possible by applying the stencil to every vertex of a face. In this case, the new topology of the refined mesh is *not ambiguous* because we may assume that the control points are indexed. Consequently, weighting a control point P_{ij} with a weight of $9/16$ when applying the stencil yields the control point P'_{ij}. Thus, the new control points may also be aligned in a matrix, which defines adjacency relations.

A shortcoming of this algorithm is that it can be applied to *quadrangular* faces only. As an advantage, though, the algorithm does not care about the valencies of vertices and can thus be applied to highly irregular quadrangular meshes.

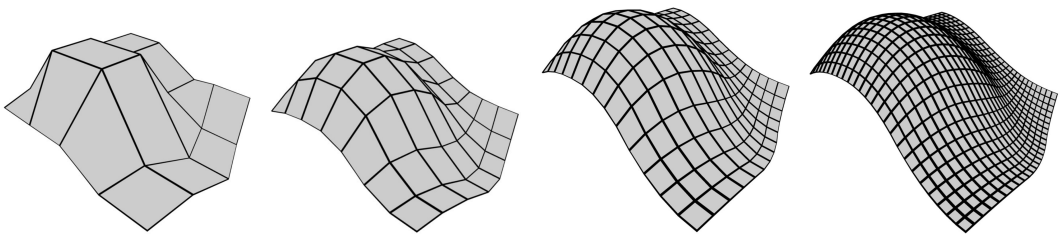


Figure 3.5: *Three steps of biquadratic B-spline subdivision being applied to a mesh with regular B-spline topology. Note that several patches are required in order to describe the surface created by the input control polygon. Decomposing meshes into B-spline patches becomes very clumsy with increasing mesh size. Hence, the* local operations *of subdivision stencils are appealing.*

3.3.2 Subdividing a bicubic B-spline surface patch

We can perform the procedure from the previous section for a bicubic B-spline surface patch. A bicubic B-spline surface patch is defined by a 4×4 matrix P of control points P_{ij}, where

$$P = \begin{pmatrix} P_{00} & P_{01} & P_{02} & P_{03} \\ P_{10} & P_{11} & P_{12} & P_{13} \\ P_{20} & P_{21} & P_{22} & P_{23} \\ P_{30} & P_{31} & P_{32} & P_{33} \end{pmatrix}.$$

As in the biquadratic case, we only need to consider the subpatch $S'(u,v)$ with $u, v \in [0, 1/2]$ due to symmetry reasons. Again, we start with the equation for the new subpatch and use the matrix representation of Theorem 2.3:

$$\begin{aligned} S'(u,v) &= S\left(\frac{u}{2}, \frac{v}{2}\right) \\ &= \left(\frac{u^3}{8}, \frac{u^2}{4}, \frac{u}{2}, 1\right) M_3 P M_3^T \begin{pmatrix} v^3/8 \\ v^2/4 \\ v/2 \\ 1 \end{pmatrix} \end{aligned}$$

The calculations are then similar to the ones of the previous section. We separate factors and introduce auxiliary terms in order to simplify the equation:

$$\begin{aligned} S'(u,v) &= \left(u^3, u^2, u, 1\right) \underbrace{\begin{pmatrix} \frac{1}{8} & 0 & 0 & 0 \\ 0 & \frac{1}{4} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{M'} M_3 P M_3^T M' \begin{pmatrix} \frac{1}{8} & 0 & 0 & 0 \\ 0 & \frac{1}{4} & 0 & 0 \\ 0 & 0 & \frac{1}{2} & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}^T \begin{pmatrix} v^3 \\ v^2 \\ v \\ 1 \end{pmatrix} \\ &= U M_3 M_3^{-1} M' M_3 P M_3^T M' (M_3^{-1})^T M_3^T V^T \\ &= U M_3 (M_3^{-1} M' M_3) P (M_3^T M'^T (M_3^{-1})^T) M_3^T V^T \\ &= U M_3 S P S^T M_3^T V^T \\ &= U M_3 P' M_3^T V^T, \end{aligned}$$

where $S = M_3^{-1} M' M_3$ and $P' = SPS^T$.

As a result of this reparametrization, we may write the subpatch as $S'(u,v) = UM_3P'M_3^TV^T$, which describes a bicubic B-spline patch with a matrix of refined control points. When solving the equations, we see that the bicubic case is more complicated than the biquadratic one: In the calculations for the new control points, three different weight patterns appear. The following expressions have been named in a suggestive manner in order to show their connection to the stencils we are going to derive.

Face points For each face in the mesh, the average of all its vertices, i.e. its *centroid*, is calculated. Here, P'_{00} is a face point. Face points are generated by the same stencil as in the biquadratic case.

$$P'_{00} = \frac{1}{4}\left(P_{00} + P_{01} + P_{10} + P_{11}\right)$$

Edge points For each edge that is adjacent to two faces in the mesh, a new point is created by averaging the average of the two face points and the midpoint of the edge. Here, P'_{01} is an edge point.

$$\begin{aligned} P'_{01} &= \frac{1}{2}\left(\frac{P'_{00} + P'_{02}}{2} + \frac{P_{01} + P_{11}}{2}\right) \\ &= \frac{1}{16}\left(P_{00} + 6P_{01} + P_{02} + P_{10} + 6P_{11} + P_{12}\right) \end{aligned}$$

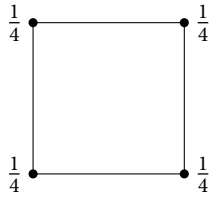
Vertex points For each vertex in the *interior* of the mesh, i.e. for each vertex that is shared by four faces, a new point is created by taking the average of the four face points of adjacent faces, the average of the four midpoints of incident edges, and the interior point itself. The new vertex is then calculated by weighting the three expressions with a factor of 1/4. Here, P'_{11} is a vertex point.

$$\begin{aligned} P'_{11} = \frac{1}{4}\Bigg[& \frac{1}{4}\left(P'_{00} + P'_{02} + P'_{20} + P'_{22}\right) \\ & + \frac{1}{4}\left(\frac{P_{11} + P_{01}}{2} + \frac{P_{11} + P_{10}}{2} + \frac{P_{11} + P_{12}}{2} + \frac{P_{11} + P_{21}}{2}\right) \\ & + P_{11}\Bigg] \\ = \frac{1}{64} & \left(P_{00} + 6P_{01} + P_{02} + 6P_{10} + 36P_{11} + 6P_{12} + P_{20} + 6P_{21} + P_{22}\right) \end{aligned}$$

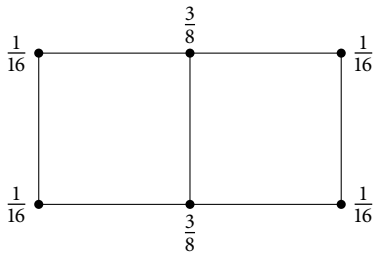
Using these terms and the weights from the calculations, we obtain the stencils as depicted in Figure 3.6. Several steps of the subdivision process are shown in Figure 3.7. Note how rapidly the refined surface seems to approach a limit. When drawing the B-spline surface corresponding to the initial mesh, there are almost *no* visual differences between the surface and the third subdivision step.

3.4 Almost arbitrary topologies

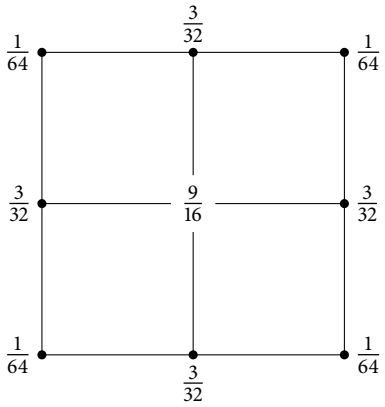
The subdivision methods that we have seen so far are limited. First, the algorithms are only applicable in the case of a B-spline topology. Second, the resulting surface will *always* be a B-spline surface of a certain degree. For most applications, however, the control polygons are *irregular*. There may be non-quadrangular faces or vertices with high valencies. The reason for the appearance of irregular meshes is a consequence of Euler's formula for convex polyhedrons (see Chapter A, Section A.1): Polyhedrons such as the platonic solids *cannot* be represented by quadrangular meshes without using vertices of valency $k \neq 4$. Consequently, we need to devise new subdivision rules for these types of meshes. In this section,



(a) *Bicubic subdivision scheme stencil for* face points



(b) *Bicubic subdivision scheme stencil for* edge points



(c) *Bicubic subdivision scheme stencil for* vertex points*. Note the symmetry of the stencil.*

Figure 3.6: *Stencils for subdividing a bicubic B-spline patch*

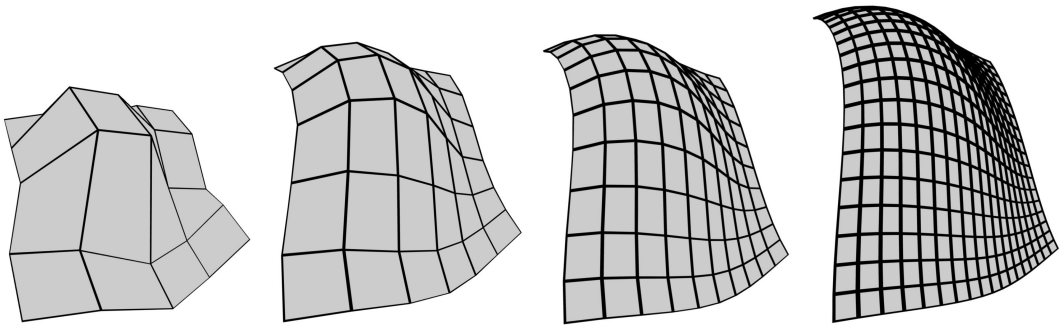


Figure 3.7: *Three steps of bicubic B-spline subdivision being applied to a mesh with regular B-spline topology. As in Figure 3.5, several bicubic patches are required for the description of the resulting B-spline surface.*

(a) *Doo-Sabin subdivision scheme stencil for regular vertices*

(b) *Doo-Sabin subdivision scheme stencil for the general case*

Figure 3.8: *Stencils for the Doo-Sabin subdivision scheme. Note how the stencil for the regular case corresponds to the stencil of the biquadratic B-spline subdivision scheme as depicted by Figure 3.4.*

we will take a look at the Doo-Sabin and the Catmull-Clark subdivision schemes. These schemes are generalizations of the biquadratic and bicubic B-spline surface subdivision schemes introduced previously and can be applied to *any* 2-dimensional manifold mesh. In general, the surface resulting from several steps of subdivision will *not* be a B-spline surface.

About the term "Arbitrary topology" An unfortunate error has crept into subdivision literature: The original paper of Catmull and Clark [CC78] introduced the term "arbitrary topology". This is *misleading*, though. It will become obvious from the description of subdivision algorithms that they only work for 2-dimensional manifold meshes. The Doo-Sabin scheme, for example, requires edges to be shared by no more than *two* faces. It is easy to define meshes where this is *not* the case. Consequently, this thesis defines subdivision schemes for *almost* arbitrary topologies. The term shall signify that the mesh is allowed to be highly irregular in terms of valencies and k-gons, but is *still* required to look locally like a 2-dimensional manifold according to Definition 3.4.

3.4.1 Doo-Sabin subdivision

The Doo-Sabin subdivision scheme generalizes biquadratic B-spline subdivision. A preliminary version was introduced by Doo [Doo78] in 1978. Following this publication, Doo and Sabin [DS78] gave a brief overview of an expanded version. This expanded version is the subdivision scheme in its currently used form.

The scheme is conceptually very simple. Since there is only one type of stencil for the biquadratic B-spline subdivision, the algorithm works on *faces* of the mesh and does not depend on the *valencies* of vertices. Consequently, the *extraordinary vertices* in the Doo-Sabin scheme are actually *extraordinary faces*, meaning k-gons with $k \neq 4$. For these kinds of faces, a stencil with user-definable weights is used. In Chapter 5, we will examine permissible weights for this stencil. The stencils for regular faces and irregular faces are depicted in Figure 3.8. As usual, computing the weights for $k = 4$ yields the weights of the biquadratic B-spline subdivision scheme. Algorithm 3.1 describes both the generation of the new points and the creation of the new topology for the mesh.

Following Algorithm 3.1, we now take a look at the properties of the Doo-Sabin subdivision scheme. When generating the topology of the new mesh, we have three kinds of faces. These are termed F-faces,

Algorithm 3.1 Doo-Sabin subdivision scheme

1: **for all** Vertices v in mesh **do**
2: **for all** Faces f that v is a part of **do**
3: Enumerate all vertices in f, starting with 0 for v and going in counter-clockwise order "around" the face in increasing numbers.
4: Let k be the number of vertices for f.
5: $\alpha_0 = (k + 5)/4k$.
6: $\alpha_j = (3 + 2\cos(2j\pi/k))/4k$
7: Create a new vertex v' by weighting the vertices of f according to $\alpha_0, \ldots, \alpha_{k-1}$.
8: **end for**
9: **end for**

10: **for all** Faces f in the mesh **do**
11: Connect the new vertices v' in the order of the old vertices v of f.
12: **end for**

13: **for all** Edges e in the mesh **do**
14: **if** Edge is part of exactly two faces **then**
15: Connect the new vertices that correspond to the start and the end vertex of the edge. Since a new vertex is computed for each face a vertex is part of, this yields four new vertices.
16: **end if**
17: **end for**

18: **for all** Vertices v in the mesh **do**
19: **if** Valency of v is greater than 2 **then**
20: **for all** Faces f that v is a part of **do**
21: Connect the new vertices corresponding to v and f.
22: **end for**
23: **end if**
24: **end for**

E-faces, and V-faces by Doo and Sabin—the terminology reflects from which elements of the mesh (faces, edges, and vertices) they are created.

F-faces An F-face is formed by the new vertices corresponding to the old vertices of the face. Thus, each face will be replaced by a smaller version of itself. In particular, k-gons with $k \neq 4$ are retained by the algorithm. Repeated application of the algorithm shrinks these faces to points.

E-faces For *manifold meshes*, non-boundary edges are always part of exactly two faces. Since we have two new vertices per face, E-faces will aways be quadrangular.

V-faces For every vertex v with valency $k > 2$, we connect the k new vertices corresponding to v. Thus, each V-face is a k-gon. For convex polyhedrons, Doo [Doo78] showed that all vertices of the new mesh have valency $k = 4$ (this is a simple consequence of Euler's formula).

Since irregular faces shrink to points and all new vertices have valency $k = 4$, the algorithm seems "reasonably well-behaved". In fact, we will prove later that the limit surface of the algorithm (for the weights chosen by Doo and Sabin) is a C^1-manifold for almost all input meshes.

Figure 3.9 depicts several steps of the Doo-Sabin scheme being applied to several platonic solids. The initial control polygons consist of triangular faces with less than 20 vertices. We can see that extraordinary areas of the mesh shrink but the shape remains unchanged. The mesh in total is quite smooth after three subdivision steps.

3.4.2 Catmull-Clark subdivision

The Catmull-Clark subdivision algorithm [CC78] has been introduced in 1978. It is a generalization of bicubic B-spline surface subdivision. Nowadays, the algorithm is one of the most common subdivision schemes. It is included in many proprietary programs (such as `ZBrush` and `LightWave 3D`), as well as in open-source software (such as `Blender` and `Wings 3D`).

The reason for the popularity of the Catmull-Clark scheme is twofold: First, the regular regions of the algorithm are locally a C^2-manifold, whereas the irregular regions are a C^1-manifold (we will expand on this fact later in more precise terms). Second, it can process meshes containing polygons with *any* number of sides and is not restricted to triangular or quadrangular meshes only.

In this section, we will retrace the steps of the original paper. Afterwards, a parametric version of the algorithm will be introduced—this is also the version we will be examining in later chapters.

Original description

The scheme is a natural extension of the bicubic B-spline subdivision scheme—it uses the same formulæ for new face points and new edge points. The stencils are slightly modified because they also need to be applicable to k-gons. The formula for vertex points, however, needs greater modifications in order to work with vertices of arbitrary valencies.

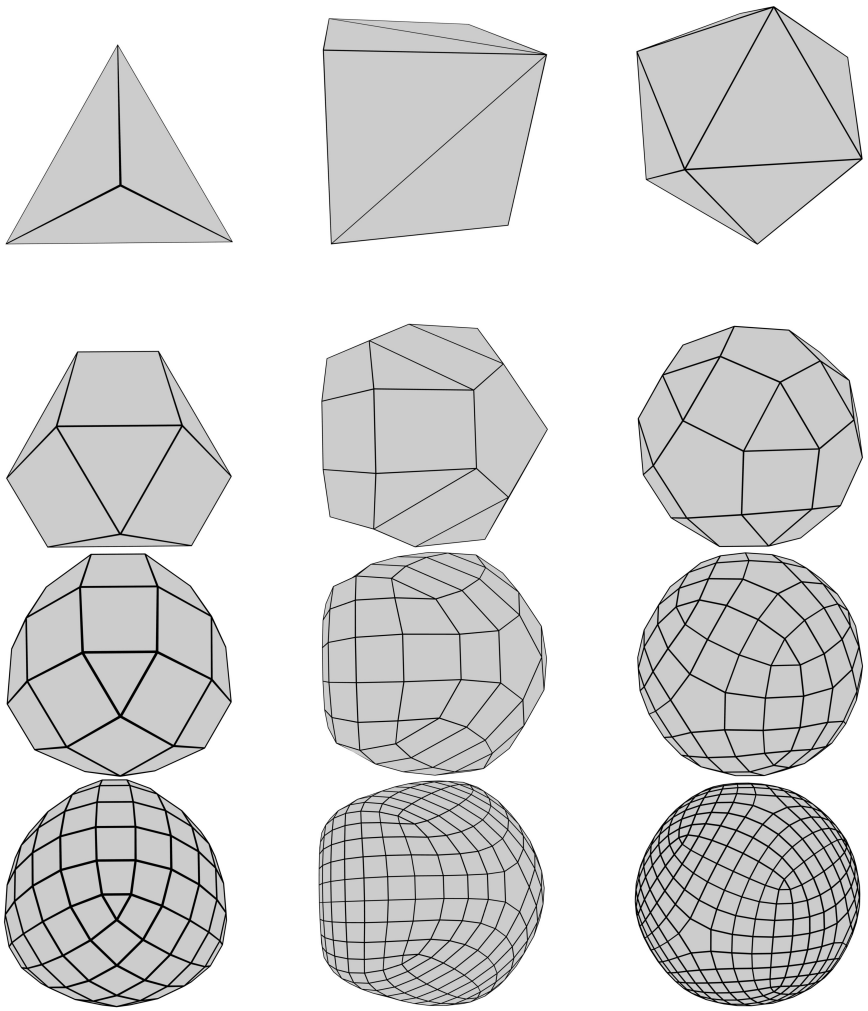
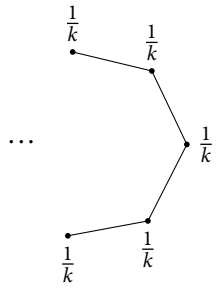
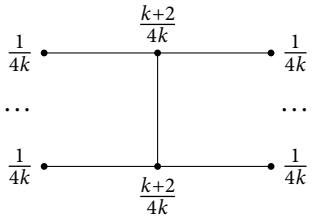


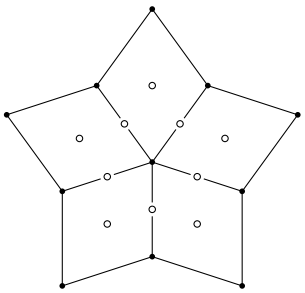
Figure 3.9: *Three steps of the Doo-Sabin algorithm applied to platonic solids (tetrahedron, hexahedron, icosahedron). The first row of images shows the input meshes.*



(a) *Catmull-Clark subdivision scheme stencil for* face points *of k-gons. As in the bicubic case, the face point is the* centroid *of the face.*



(b) *Catmull-Clark subdivision scheme stencil for* edge points *adjacent to* 2 *k-gons. Note that for* $k = 4$*, the weights coincide with the weights of the bicubic subdivision scheme as depicted in Figure 3.6.*



(c) *The "area of influence" in the mesh that is used in order to calculate* vertex points *corresponding to vertices of valency k. Here,* $k = 5$.

Figure 3.10: *Stencils for the Catmull-Clark subdivision scheme*

By the original suggestion of Catmull and Clark [CC78], a new vertex point v' corresponding to a vertex v of valency k is calculated as a linear combination of surrounding vertices,

$$v' = \frac{F}{k} + \frac{2E}{k} + \frac{v(k-3)}{k}, \tag{3.3}$$

where F is the average of the face points of the k faces adjacent to v, and E is the average of the midpoints of the k edges incident on v. For the regular case with $k = 4$, Equation 3.3 yields the vertex point stencil of the bicubic subdivision scheme as depicted in Figure 3.6. The stencils for irregular parts of the mesh are depicted by Figure 3.10. Note that the vertex point stencil is shown without any weights—we will soon derive a weighted representation.

Having described the stencils, the new vertices still need to be connected correctly—this is explained by Algorithm 3.2. Since the scheme uses more stencils than the Doo-Sabin scheme, we may expect that the Catmull-Clark scheme takes greater advantage of the underlying structure of the mesh. This is indeed the case: We can observe that the subdivision scheme only creates *quadrangular faces* (which are not planar in general), and a k-gon with $k \neq 4$ will yield an extraordinary vertex of valency k. Furthermore, extraordinary vertices of quadrangles retain their valency. As a consequence, the number of *extraordinary vertices* in the mesh remains fixed after the initial subdivision step, whereas the size and number of regular parts of the mesh grows with each subdivision step.

Figure 3.11 shows several steps of the Catmull-Clark scheme being applied to several platonic solids. Again, the initial control polygons consist of triangular faces with less than 20 vertices. After a mere three

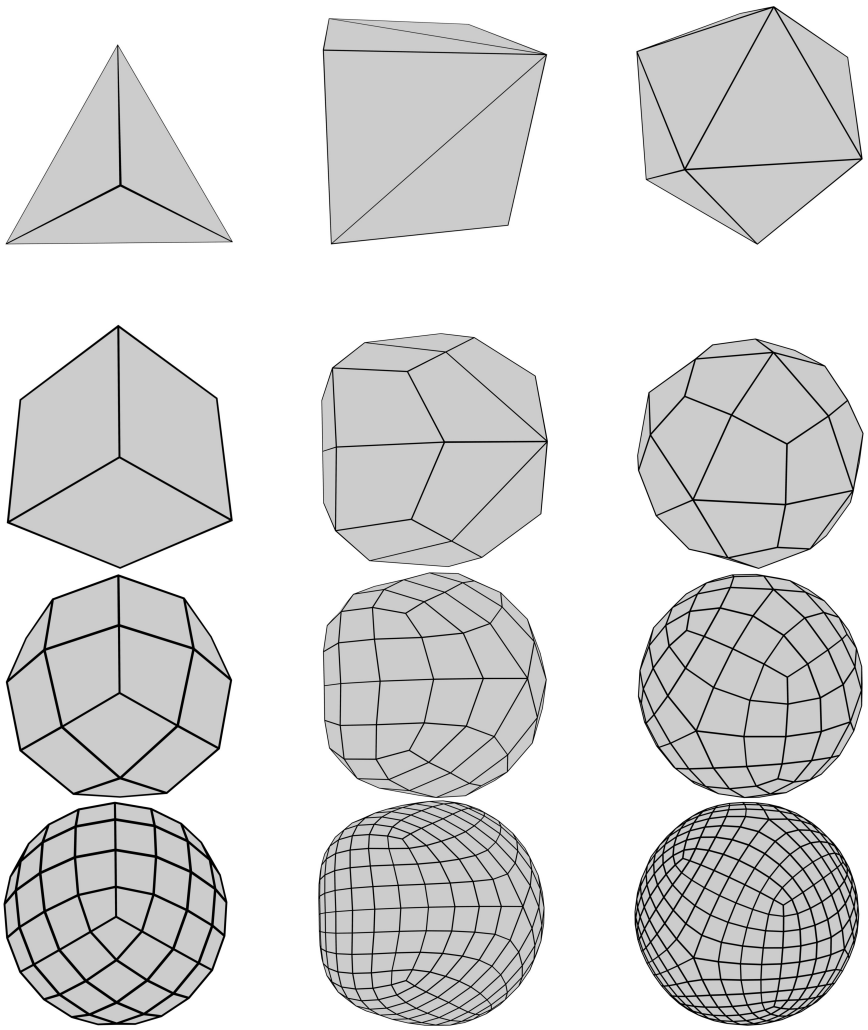


Figure 3.11: *Three steps of the Catmull-Clark algorithm applied to platonic solids (tetrahedron, hexahedron, icosahedron). The first row of images shows the input meshes.*

Algorithm 3.2 Catmull-Clark subdivision scheme

1: **for all** Faces f in the mesh **do**
2: Create a *face point* as the geometrical centre of the face.
3: **end for**
4: **for all** Edges e in the mesh **do**
5: Create an *edge point* as the average of the average of the two *face points* that correspond to the two faces e is a part of and the midpoint of e.
6: **end for**
7: **for all** Vertices v in the mesh **do**
8: Create a *vertex point* using Equation 3.3.
9: **end for**
10: **for all** Vertices v in the mesh **do**
11: **for all** Faces f that v is a part of **do**
12: Connect the *vertex point* corresponding to v with an *edge point* of an edge that is incident on v and part of f.
13: Connect the *edge point* used before with the *face point* corresponding to f.
14: Connect the *face point* used before with the remaining *edge point* of an edge that is incident on v and part of f.
15: Make this a face in the refined mesh.
16: **end for**
17: **end for**

subdivision steps, the mesh is sufficiently smooth and the irregular regions corresponding to extraordinary vertices have become very small.

Parametrical description

Figure 3.10 depicts the "area of influence" for the calculation of vertex points. Since the weights of this stencil depend on both the valency of the vertex *and* the number of edges of adjacent faces, it is rather unwieldy. We now derive equations for the vertex point stencil in the special case that *all* adjacent faces are quadrangular. Since all faces of the mesh become quadrangles after one step of the Catmull-Clark algorithm, this condition is *not* a loss of generality. Hence, let v be a vertex of valency k. Using Equation 3.3, we see that all vertices that are not part of an incident edge of v are weighted with $(4k^2)^{-1}$ (because they only appear within the face points equation). Vertices that are part of an incident edge of v, however, are part of two faces and one edge. Thus, they are weighted with $(4k^2)^{-1} + (4k^2)^{-1} + 2(2k^2)^{-1} = 3(2k^2)^{-1}$. Finally, since v is part of k faces and k edges, it is weighted with $(4k)^{-1} + 2(2k)^{-1} + (k-3)/k = 1 - 7(4k)^{-1}$.

Consequently, we may describe subsequent steps of the Catmull-Clark scheme by using three weights for all vertices around an extraordinary vertex: The vertex itself is weighted with α, all directly incident vertices are weighted with β/k, and all remaining vertices are weighted with γ/k. For the original scheme of Catmull-Clark, we have

$$\alpha = 1 - \frac{7}{4k},\ \beta = \frac{3}{2k},\ \gamma = \frac{1}{4k}, \tag{3.4}$$

and obviously, $\alpha + \beta + \gamma = 1$ (otherwise, affine invariance would not hold). The resulting stencil is depicted in Figure 3.12. By picking arbitrary values for α, β, and γ, the smoothness properties of the algorithm can be changed. In Chapter 5, we will discuss feasible weights and the consequences of erroneous weights.

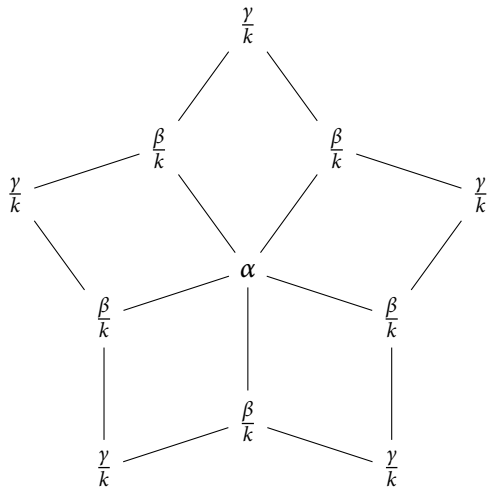


Figure 3.12: *Catmull-Clark subdivision scheme stencil for* vertex points *corresponding to a vertex of valency k (see text for a detailed derivation). Here, $k = 5$. If one of the faces is not yet quadrangular, Equation 3.3 needs to be used for the calculation. The weights α, β, and γ must sum to unity. Catmull and Clark suggested $\alpha = 1-7(4k)^{-1}$, $\beta = 3(2k)^{-1}$, and $\gamma = (4k)^{-1}$.*

3.5 Summary

This chapter gave a brief overview of the history of subdivision algorithms for curves and surfaces. We saw how to split the parameter domain of B-spline surface patches in order to obtain *refined* patches from the initial control polygon. The refined patches converge against a B-spline surface of a certain degree. Furthermore, we took a look at *generalizations* of B-spline surface subdivision, i.e. algorithms that may be used to smooth patches of almost arbitrary topology. These generalizations ultimately led us to the *Doo-Sabin* and the *Catmull-Clark* subdivision schemes. We introduced both schemes and took a brief look at their properties. In addition, we extended their original definitions by introducing user-definable *weights*, which we will analyse in Chapter 5.

4 Analysing subdivision algorithms

This chapter introduces methods for analysing the smoothness properties of subdivision schemes. For this purpose, we are required to switch between two different viewpoints: Instead of thinking of subdivision algorithms as refinement schemes that are applied to meshes, we will also consider a subdivision algorithm to be a *linear map* in the space of control points. The main idea is to represent the surface that results from repeated application of this linear map as the graph of a *regular injective* function. If this function is known, smoothness properties can be determined rather easily.

In the following sections, we will derive criteria that are essential for thoroughly analysing the smoothness properties of a given subdivision scheme: We will start by defining the problem in rigorous mathematical terms. Next, we will derive an invariant, the *characteristic map*, of a subdivision scheme. Briefly, this is a function that *solely* depends on the subdivision algorithm itself and not on any input data. Using the characteristic map, we will prove a theorem that guarantees smooth limit surfaces, provided the characteristic map is *regular* and *injective*. Following this result, the remainder of this chapter is dedicated to deriving conditions under which the characteristic map will be regular and injective. We will show that the analysis of the characteristic map can be limited to a *single segment* in the complex plane. Furthermore, we will see that in some cases, checking for regularity and injectivity is as simple as checking signs of certain functions.

This chapter is based on a paper by Reif [Rei95], a paper by Peters and Reif [PR98], and Reif's habilitation thesis [Rei98]. Care has been taken to use a *consistent* notation. Most of the original proofs have been extended to provide more details.

4.1 Preliminary definitions

In order to derive tools for analysing subdivision schemes, a solid mathematical foundation is essential. This section presents suitable parametrizations for spline surfaces and subdivision algorithms. Since we only require a small subset of the theory, many details will only be briefly touched upon.

4.1.1 Parameter space and representations

Let Σ_0 be a parametrically smooth spline surface consisting of quadrangular patches. In addition, we assume that Σ_0 is an infinite planar surface with a single n-sided hole as its only boundary—this enables us to *ignore* the outer boundary of the surface without loss of generality, thereby simplifying the analysis to some extent.

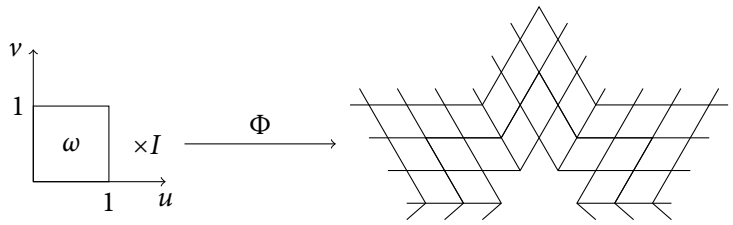


Figure 4.1: *Representation* Φ *of the parameter space* Ω *for* Σ_0.

Definition 4.1 (Parameter space of a spline surface)**.** Let I be a finite subset of $\mathbb{N}$. The parameter space Ω of a parametrically smooth spline surface is defined by

$$\Omega := \omega \times I,$$

where ω is the unit square, i.e. $\omega = [0,1] \times [0,1]$. Furthermore, a *neighbourhood relation* is defined for pairs (ω, i) and $i \in I$ by identifying adjacent edges of unit squares. Throughout this thesis, equality in the parameter space will only be defined modulo the neighbourhood relation.

In order to visualize the parameter space, we define a *representation* of Ω. The representation is an embedding of Ω and its topology in the real plane. A representation does not need to exist in every case (take a cube, for example—there is obviously *no* planar embedding), but it certainly exists for Σ_0.

Definition 4.2 (Representation of Ω)**.** Let Φ^i with $i \in I$ be a set of smooth injective functions from the unit square to $\mathbb{R}^2$. The functions are required to have the property that adjacent edges have a *common image*, i.e. Φ^i needs to preserve the neighbourhood relation. With these conditions satisfied, a *representation* is a map Φ with

$$\Phi : \Omega \ni (\omega, i) \mapsto \Phi^i(\omega) \in \mathbb{R}^2.$$

We call the image $\Gamma := \Phi(\Omega)$ of the parameter space under Φ a *mesh*. Figure 4.1 shows an example for the representation of a parameter space.

We can consider the mesh Γ as a mesh in the usual sense of computer graphics, as introduced by Definition 3.1: The faces of the mesh correspond to the image of the unit squares, the edges correspond to the image of the edges of the unit squares, and the vertices correspond to the image of the four corner points of the unit square. In Chapter 2 and Chapter 3, we already saw that meshes with four faces meeting at every vertex allow us to use them as control polygons of B-spline surfaces that satisfy certain smoothness conditions. This motivates the next definition.

Definition 4.3 (Regular mesh)**.** We call an interior vertex of a quadrangular mesh Γ *regular* if four faces join at the vertex. Otherwise, we call the vertex *irregular*. If all interior vertices of a mesh are regular, we call the mesh Γ *regular*. Following Chapter 3, we will also use the terms *ordinary* vertex and *extraordinary* vertex.

The control net of any B-spline surface is a typical example of a regular mesh. In Chapter 3, we have already seen that irregular meshes inevitably appear when more complex objects need to be modelled. Peters and Reif [PR08] motivate the definition of regularity by defining generalized spline functions that feature *singularities* at irregular vertices.

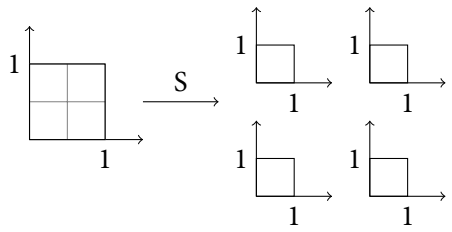


Figure 4.2: *Subdividing the parameter space*

4.1.2 Prolongations

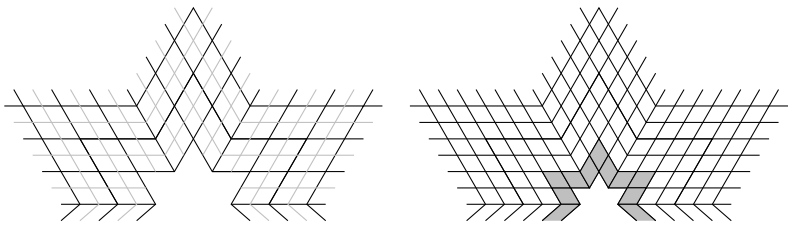
In our current setting, the resulting mesh is *regular*. However, it is impossible to add more patches to the mesh without breaking symmetry or regularity. Hence, in order to add more patches, the parameter space Ω needs to be refined. Refinement means that a surface $\Sigma_0(\Omega)$ is represented as a surface $\widetilde{\Sigma}(\widetilde{\Omega})$ that uses a set of *finer* patches. We can achieve this by splitting the unit square into four smaller squares and normalizing them. An example of this process is depicted in Figure 4.2. Reif [Rei98] showed that subdividing the spline domain is an isomorphism, i.e. the original spline domain and the refined spline domain are *isomorphic*. Furthermore, this isomorphism can be shown to induce an embedding of the original spline surface in the refined spline surface.

Subdivision of Ω also yields a new mesh $\widetilde{\Gamma}$, generated by some representation $\widetilde{\Phi}$. The new mesh is finer than the old mesh and can be *prolonged*, i.e. there is a new layer $\mathrm{pr}(\Gamma)$ of patches such that the subdivided mesh $S(\Gamma) := \widetilde{\Gamma} \cup \mathrm{pr}(\Gamma)$ remains *regular*. We will soon give a more rigorous definition of prolongations.

Due to the subdivision process, the scale of $S(\Gamma)$ is halved and further prolongations will extend the regular parts of the mesh, whereas the irregular parts of the mesh shrink. Thus, without loss of generality, every mesh can be assumed to contain only *one* irregular vertex in a sufficiently large neighbourhood. Furthermore, since Γ and $S(\Gamma)$ are topologically equivalent, $S(\Gamma) =: S(\Phi)(\Omega)$ may be viewed as a new representation of the parameter space. See Figure 4.3 for an example.

Since $S(\Gamma)$ remains regular, we can find a parametrically smooth spline surface $\Sigma_1 := S(\Sigma_0)$ that is parametrized over $S(\Gamma)$ such that the restriction of Σ_1 to the subdivided mesh $\widetilde{\Gamma}$ coincides with the original surface Σ_0. This yields a new part of the surface, namely

$$\mathrm{pr}(\Sigma_0) := \mathrm{cl}\left(\Sigma_1 \setminus \Sigma_0\right),$$



(a) $\Gamma \subset \widetilde{\Gamma}$ **(b)** $S(\Gamma) = \widetilde{\Gamma} \cup \mathrm{pr}(\Gamma)$

Figure 4.3: *The left figure (a) shows the old mesh (black lines) as a subset of the new mesh (grey lines). The right figure (b) shows the prolonged mesh that is the union of the subdivided mesh and a* prolongation *(shown in grey) of the old mesh.*

which we call the *prolongation* of Σ_0. The prolongation of Σ_0 is not uniquely defined because we only require it to join smoothly with Σ_0 and depend on the *innermost* layer of patches of Σ_0, i.e.

$$\mathrm{pr}(S(\Sigma_0)) = \mathrm{pr}(\mathrm{pr}(\Sigma_0)) = \mathrm{pr}^2(\Sigma_0).$$

We can therefore choose *any* subdivision algorithm in order to generate the prolongations. As a convenience, we will also write

$$x_m := \mathrm{pr}(\Sigma_m) = \mathrm{cl}\,(\Sigma_{m+1} \setminus \Sigma_m) = \mathrm{pr}^m(\Sigma_0)$$

for the prolongation. If the subdivision algorithm is reasonable, each iteration will create more layers that fill the n-sided hole completely when taking the limit—we will state this in more precise terms shortly.

To sum up our results so far: Subdividing the parameter space creates an ascending sequence of smooth surfaces, i.e.

$$\Sigma_0 \subset \Sigma_1 \subset \ldots,$$

which are supposed to converge to the *limit surface*

$$\Sigma = \bigcup_{m\in\mathbb{N}} \Sigma_m,$$

which we can also represent as

$$\Sigma = \Sigma_0 \cup \bigcup_{m\in\mathbb{N}} x_m$$

by a result of Reif [Rei95]. In the subsequent analysis of subdivision schemes, we will examine the properties of the limit surface Σ.

4.1.3 Convergence and continuity

Before elaborating on the precise parametrization of the subdivision process and the prolongations, we require some definitions about convergence and continuity.

Definition 4.4 (Convergence of subdivision procedures)**.** Let S be a subdivision procedure in the general sense, as outlined above. We say that S is *convergent* if there is a *unique* limit point p such that

$$\lim_{m\to\infty} p_m = p$$

for *any* sequence of points $(p_m)_{m\in\mathbb{N}}$ with $p_m \in x_m = pr^m(\Sigma_0)$. In other words, S is convergent if all prolongations converge to a single, unique point.

Definition 4.5 (Closure of the surface)**.** The *closure* of the limit surface Σ is the surface

$$\overline{\Sigma} := \Sigma \cup p.$$

For *convergent* algorithms, the closure is a surface of genus 0.

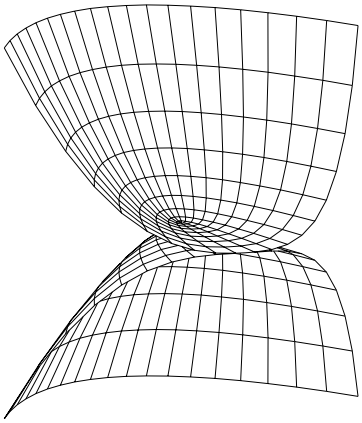


Figure 4.4: *Following Peters and Reif [PR08], p. 25, this surface is parametrized by* $\Sigma(u,v) = (u^2 - v^2, uv, u^3)$. *Calculating the limit of the normalized cross product of the partial derivatives yields a limit point of* $(0,0,1)$ *at the origin. Thus, the surface is tangent plane continuous. Yet, the surface is not* regular *at the origin because of the two coinciding sheets.*

Definition 4.6 (Tangent plane continuity)**.** The closure $\overline{\Sigma}$ is *tangent plane continuous* in p if S converges and if there is a unique limit point $\mathrm{n}(p)$ for *any* sequence of normal vectors, i.e.

$$\lim_{m\to\infty} \mathrm{n}(p_m) = \mathrm{n}(p),$$

for *any* sequence of points $(p_m)_{m\in\mathbb{N}}$ with $p_m \in x_m = pr^m(\Sigma_0)$.

It is important to note that $\mathrm{n}(p)$ is simply the *limit* of the normal vectors and *not* the normal vector of $\overline{\Sigma}$ at p, which may not even exist.

Definition 4.7 (Regular surface)**.** A tangent plane continuous surface $\overline{\Sigma}$ is called *regular* at p if there is a regular smooth parametrization of $\overline{\Sigma}$ in a neighbourhood around p, which means that

$$\overline{\Sigma} = \overline{\Sigma}(u,v)$$

for parameters u, v such that $\overline{\Sigma}$ is 1-times continuously differentiable and the partial derivatives

$$\frac{\partial\overline{\Sigma}(u,v)}{\partial u} \quad \text{and} \quad \frac{\partial\overline{\Sigma}(u,v)}{\partial v}$$

are *linearly independent*.

From the point of view of *differential topology*, regular surfaces are differentiable manifolds. Consequently, we will also employ this term whenever the importance of the topology of the surface is stressed. In particular, we only require (regular) C^1-manifolds, i.e. manifolds whose transition maps are 1-times continuously differentiable.

The notion of tangent plane continuity is distinctly weaker than the notion of regularity: A surface may be tangent plane continuous and still contain self-intersections, which do not permit a regular smooth parametrization. See Figure 4.4 for an example.

4.1.4 Parametrizing the prolongations

We now return to the prolongations defined above. In both Reif's habilitation thesis [Rei98] and a paper of Peters and Reif [PR98], it was observed that the prolongations x_m can be defined more conveniently than by using the parameter space Ω. The key observation is that a prolongation consists of a *ring* of patches around the n-sided hole. A parameter space that reflects this fact is better suited as a parametrization. This leads us to the following definition:

Definition 4.8 (Parameter space for prolongations)**.** Let $\omega_0 := [0,2]^2 \setminus [0,1)^2$ as depicted in Figure 4.5. The parameter space Ω_0 for prolongations is defined as

$$\Omega_0 := \omega_0 \times \mathbb{Z}_n,$$

where $\mathbb{Z}_n$ is the ring of congruence classes modulo n.

Definition 4.9 (Representation of Ω_0)**.** We can define representations for Ω_0 in complete analogy with Definition 4.2 and write $\Gamma_0 := \Phi(\Omega_0)$ for the corresponding mesh.

For the analysis of subdivision algorithms, we assume that it is possible to parametrize the prolongations x_m over Ω_0 by using K control points $B_m^k \in \mathbb{R}^3$ and K piecewise polynomial functions N^k such that

$$x_m : \Omega_0 \ni (u,v,j) \mapsto x_m^j(u,v) = \sum_{k=0}^{K-1} N^k(u,v,j) B_m^k. \tag{4.1}$$

The functions N^k are assumed to lie in $C^1(\Omega_0)$, which is the space of parametrically smooth functions over Ω_0. Furthermore, they are required to be linearly independent and form a *partition of unity*, i.e. $\sum_{k=0}^{K-1} N^k(u,v,j) \equiv 1$. We call N^k the *basis functions* and B_m^k the *control points*. As an abbreviation, we use vectors and rewrite Equation 4.1 as

$$x_m^j(u,v) = N(u,v,j) B_m, \tag{4.2}$$

where $N(u,v,j)$ is a row vector collecting the basis functions and B_m is a column vector of points in $\mathbb{R}^3$, which we may think of as a $K \times 3$ matrix of entries from $\mathbb{R}$. See Figure 4.6 for an illustration of the previous definitions and equations.

It should be noted that this type of parametrization does not need to exist for every subdivision algorithm, but it *does* exist in the case of algorithms generalizing B-spline subdivision, such as the schemes analysed in this thesis: We have already seen in Chapter 2 that the B-spline basis functions for curves and surfaces are linearly independent and form a partition of unity. In addition, we saw in Chapter 3 that both subdivision algorithms create B-spline surface patches for regular parts of the mesh. Consequently, they can be expressed in the form of Equation 4.2. Further details may be obtained from Peters and Reif [PR08].

With small modifications, Equation 4.2 also holds for subdivision schemes on purely triangular meshes. Umlauf [Uml00], for example, applied the methods of this chapter to the Loop subdivision algorithm, which is a generalization of *box spline* subdivision.

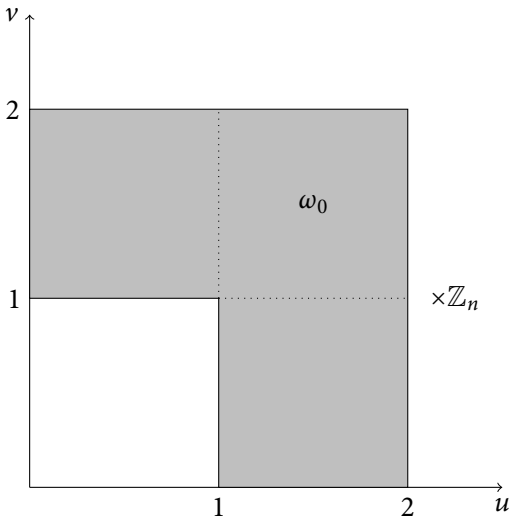


Figure 4.5: *Parameter space Ω_0 of the prolongations. The dotted lines are added for visualization purposes only.*

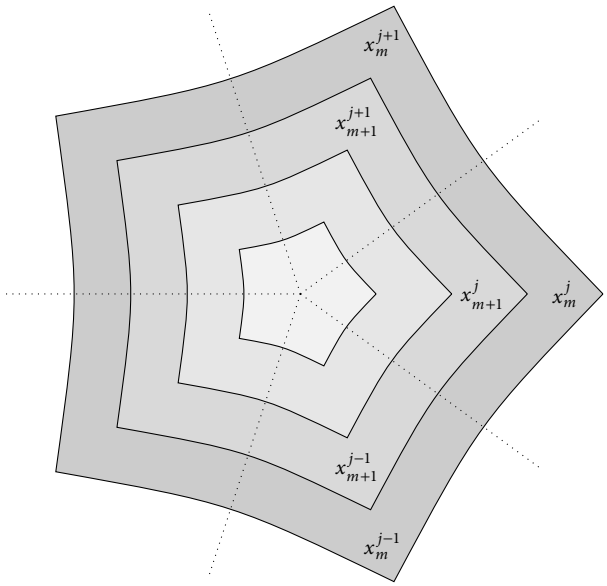


Figure 4.6: *Multiple surface layers for a mesh with a single hole of valency $n > 4$. The indices are defined modulo n.*

4.1.5 Subdivision algorithms

Having defined the prolongations x_m in terms of control points and basis functions, we are now able to state the notion of a subdivision algorithm in precise terms.

Definition 4.10 (Subdivision algorithm)**.** A subdivision algorithm is a linear and stationary map in the space of control points, i.e.

$$B_m = AB_{m-1} = A^2B_{m-2} = \cdots = A^{m-1}B_0,$$

where A is the $K \times K$ *subdivision matrix*. *Stationary* means that A does not depend on m, the subdivision level. This assumption is reasonable and does hold for the most common subdivision schemes. Furthermore, all rows of A must sum to unity. This implies that AB_0 is an affine combination of control points and consequently, the scheme is affine invariant.

In the subsequent analysis, we will find more requirements for subdivision matrices such that the resulting surfaces satisfy certain smoothness conditions—the next section will expand on this. In addition, it needs to be stressed that the subdivision matrix is only applied to the control points of the prolongations. We do *not* require A to be applicable to all control points of the mesh. Hence, the subdivision matrix A is a *finite* quadratic matrix that generates a vector B_{m+1} of new control points from a vector B_m of old control points. Since most subdivision algorithms depend on a small set of control points only, A is a *sparse matrix* in most cases.

4.2 Smoothness conditions

This section will put the definitions of the preceding sections to use. We will derive *necessary* and *sufficient* conditions for the smoothness of a subdivision scheme. To this end, we first discuss how to determine whether a subdivision scheme is *convergent*. Following this, we will introduce the *characteristic map* and prove that its properties determine the regularity of the limit surface.

Definition 4.11 (Order of sequences)**.** Let $\lambda \in \mathbb{R} \setminus \{0\}$. A sequence $x_1, x_2, \ldots$, in $\mathbb{R}^n$ is of order $o(\lambda^m)$ if

$$\lim_{m\to\infty} \frac{\|x_m\|}{|\lambda|^m} = 0$$

with respect to some norm $\|\cdot\|$ of $\mathbb{R}^n$. Since norms of finite real vector spaces are equivalent, the norm can be chosen arbitrarily. Furthermore, sequences of functions converge *uniformly* by this definition because all domains introduced in this chapter are compact.

Lemma 4.1. *Let $\lambda_0, \lambda_1, \ldots, \lambda_{K-1}$ be the eigenvalues of the subdivision matrix A such that*

$$|\lambda_0| \geq |\lambda_1| \geq \cdots \geq |\lambda_{K-1}|$$

and let $\psi_0, \psi_1, \ldots, \psi_{K-1}$ be corresponding eigenvectors. If r is chosen as the smallest *index such that $|\lambda_r| > |\lambda_{r+1}|$, then*

$$A^m\psi_s = o(\lambda_r^m)$$

for $m \to \infty$ and all $s > r$.

Proof. It is a standard fact of numerical analysis that there is a norm in $\mathbb{R}^K$ that differs from the spectral radius of the matrix by an arbitrarily small ϵ only (see Stoer and Bulirsch [SB96], Theorem 6.9.2, pp. 407–408). Thus, $\|A^m \psi_s\| < |\lambda_r|^m + \epsilon$ and it follows that $A^m \psi_s = o(\lambda_r^m)$. ∎

Theorem 4.1 (Convergence of a subdivision algorithm)**.** *Let A be a subdivision algorithm and let its eigenvalues be sorted by modulus. If $1 = \lambda_0 > |\lambda_1|$, then A converges.*

Proof. By Definition 4.10, the rows of the subdivision matrix A sum to unity. Thus, $\lambda_0 = 1$ is an eigenvalue of A and $\psi_0 = (1, 1, \ldots, 1, 1)^T$ is a corresponding eigenvector. Let ψ_1, ψ_2, …, ψ_{K-1} be the remaining eigenvectors of A. Since eigenvectors form a basis, every component B_0^j, with $j \in \{0, 1, 2\}$, of the vector B_0 of control points can be written as

$$B_0^j = \sum_{i=0}^{K-1} \mu_i^j \psi_i$$

with coefficients $\mu_i^j \in \mathbb{R}$. Combining these coefficients into a vector of $\mathbb{R}^3$, i.e.

$$p_i = \begin{pmatrix} \mu_i^0 \\ \mu_i^1 \\ \mu_i^2 \end{pmatrix},$$

allows us to write the sequence of vectors of control points as

$$B_0 = \sum_{i=0}^{K-1} \psi_i p_i, \tag{4.3}$$

and by Lemma 4.1,

$$B_m = \psi_0 p_0 + o(1). \tag{4.4}$$

Using the equations from above, we can simplify Equation 4.1 and Equation 4.2, thereby obtaining:

$$\begin{aligned} x_0^j(u, v) &= \sum_{i=1}^{k} N(u, v, j) \psi_i p_i \\ x_m^j(u, v) &= \underbrace{N(u, v, j) \psi_0}_{=1} p_0 + o(1) = p_0 + o(1) \end{aligned}$$

This works because the functions $N(u, v, j)$ are assumed to form a *partition of unity*. Since the order function converges uniformly in (u, v), we get

$$\lim_{m \to \infty} x_m^j(u, v) = p_0.$$

Thus, $p := p_0$ is the limit point and the algorithm converges. ∎

The key observation of Reif [Rei95] is that all smoothness properties of $\overline{\Sigma}$ can be derived from the leading eigenvalues of the subdivision matrix A and a map Ψ, which we will describe below. The map Ψ depends on the eigenvectors that correspond to the leading eigenvalues and the basis functions for the subdivision algorithm, but *not* on the control points. Consequently, Ψ is called *characteristic map* of the subdivision scheme.

Definition 4.12 (Characteristic map)**.** Let A be a subdivision matrix with eigenvalues

$$\lambda_0 = 1 > |\lambda_1| = |\lambda_2| > |\lambda_3|$$

and let ψ_1, ψ_2 be the eigenvectors corresponding to $\lambda_1 = \lambda_2$. The characteristic map $\Psi : \Omega_0 \to \mathbb{R}^2$ is defined by

$$\Psi : (u, v, j) \mapsto N(u, v, j)V := N(u, v, j)\,[\psi_1, \psi_2]\,,$$

where V is a $K \times 2$ matrix with vectors ψ_1 and ψ_2 as its columns. Ψ is called *regular* if

$$\Delta(u, v, j) := \det \frac{\partial \Psi(u, v, j)}{\partial(u, v)} \neq 0$$

for all $(u, v, j) \in \Omega_0$. In other words, Ψ is regular if its Jacobian matrix is regular.

We shall also require a *complex* form of the characteristic map. It will be used later on in this chapter when we derive smoothness criteria. The complex form of the characteristic map is given by

$$\Psi_* : \Omega_0 \ni (u, v, j) \mapsto N(u, v, j)\psi_* \in \mathbb{C},$$

where $\psi_* := \psi_1 + i\psi_2$. Since the properties of the characteristic map are not changed by this complexification ($\mathbb{R}^2$ and $\mathbb{C}$ may be viewed as isomorphic vector spaces), we will switch between the real variant of the characteristic map and its complex variant without further notice.

Following the notation for the prolongations around an n-sided hole, we will index different *segments* of the characteristic map by superscripts: Ψ^j signifies the jth segment of Ψ and Ψ_*^j denotes the complexification of said segment.

For visualization purposes, Ψ is best considered a 2-dimensional B-spline function with control points from $\mathbb{R}^2$ that are determined by the rows of V. The characteristic map may not seem well-defined at a first glance because there is still some choice in the eigenvectors ψ_1 and ψ_2 that are used in its definition. The following lemma, however, proves that a particular choice of the eigenvectors does *not* change any important properties of the characteristic map.

Lemma 4.2. *Injectivity and regularity of the characteristic map do not depend on a particular choice of ψ_1 and ψ_2.*

Proof. Let $\widetilde{V} := [\widetilde{\psi}_1, \widetilde{\psi}_2]$ be another feasible matrix for defining the characteristic map. Since the eigenvectors $\widetilde{\psi}_1$ and $\widetilde{\psi}_2$ span the same space as ψ_1 and ψ_2, the columns of $\widetilde{V}$ span the same linear space as the columns of V. Thus, a change of basis can be performed and $\widetilde{V}$ can be written as

$$\widetilde{V} = T^{-1}VT$$

for some regular matrix T. Using this equation, we obtain:

$$\widetilde{\Psi}(u, v, j) := N(u, v, j)\widetilde{V} = T^{-1}\Psi(u, v, j)T \tag{4.5}$$

$$\widetilde{\Delta}(u, v, j) := \det T^{-1}\Delta(u, v, j) \det T = \Delta(u, v, j) \tag{4.6}$$

Since T is regular, the properties of the composite functions remain unchanged. In particular, injectivity and regularity are preserved. ∎

Lemma 4.3. *If the characteristic map Ψ is regular, then*

$$\inf_{\Delta \in \Omega_0} |\Delta| > 0.$$

Proof. The determinant function is continuous, so Δ and $|\Delta|$ are also continuous. Since Ω_0 is compact, $|\Delta|$ attains its minimum in Ω_0. This minimum is greater than zero by assumption. ∎

Using the characteristic map, we can obtain our first result concerning the smoothness of $\overline{\Sigma}$: We will use the next theorem as the starting point towards better smoothness criteria.

Theorem 4.2. *Let $\lambda_1 = \lambda_2$ be a real eigenvalue with algebraic and geometric multiplicity 2 such that $\lambda_0 = 1 > |\lambda_1| = |\lambda_2| > |\lambda_3|$. If the characteristic map is regular, then $\overline{\Sigma}$ is* tangent plane continuous *for almost every initial vector B_0 of control points.*

Proof. Tangent plane continuity depends on the normals of the surface. Thus, we try to find a closed expression for the normals and show that it is well-defined. For this purpose, we start with the sequence B_m of control points as defined by Equation 4.2, derive expressions for the surface, and, as a last step, calculate the limit of the normal vectors of the surface.

Let $\lambda := \lambda_1 = \lambda_2$. Then the sequence of refined control points can be expanded as

$$B_m = \psi_0 p_0 + \lambda^m \left(\psi_1 p_1 + \psi_2 p_2\right) + o(\lambda^m),$$

where we have used that $\lambda_0^m = 1$ by assumption. The coefficients p_1 and p_2 are defined as in the proof of Theorem 4.1. Using these control points for the parametrization of the prolongations $x_m^j(u,v)$, we get

$$x_m^j(u,v) = \underbrace{N(u,v,j)\psi_0}_{=1} p_0 + \lambda^m N(u,v,j)(\psi_1 p_1 + \psi_2 p_2) + o(\lambda^m). \tag{4.7}$$

The partial derivatives are given by:

$$x_{m,u}^j(u,v) = \lambda^m N_u(u,v,j)(\psi_1 p_1 + \psi_2 p_2) + o(\lambda^m) \tag{4.8}$$

$$x_{m,v}^j(u,v) = \lambda^m N_v(u,v,j)(\psi_1 p_1 + \psi_2 p_2) + o(\lambda^m) \tag{4.9}$$

In order to calculate the normal at (u,v), we need to take the cross product:

$$\begin{aligned} x_{m,u}^i(u,v) \times x_{m,v}^i(u,v) = \lambda^{2m}(&N_u\psi_1 p_1 \times N_v\psi_1 p_1 + N_u\psi_2 p_2 \times N_v\psi_1 p_1 + \\ &N_u\psi_1 p_1 \times N_v\psi_2 p_2 + N_u\psi_2 p_2 \times N_v\psi_2 p_2 + o(1)) \end{aligned}$$

The last term contains all terms of higher order, which we can ignore. By using the anticommutativity of the cross product, we arrive at

$$\begin{aligned} x_{m,u}^i(u,v) \times x_{m,v}^i(u,v) &= \lambda_2^{2m}\left(\left(N_u\psi_1 \cdot N_v\psi_2 - N_v\psi_2 \cdot N_u\psi_1\right)(p_1 \times p_2) + o(1)\right) \\ &= \lambda_2^{2m}\left(\Delta(u,v,j)(p_1 \times p_2) + o(1)\right). \end{aligned} \tag{4.10}$$

By definition, the normal vectors of the surface are given by

$$n_m(u,v,j) = \frac{x^j_{m,u} \times x^j_{m,v}}{\left\| x^j_{m,u} \times x^j_{m,v} \right\|}.$$

This yields

$$n_m(u,v,j) = \frac{p_1 \times p_2}{\|p_1 \times p_2\|} + \frac{o(1)}{|\Delta(u,v,j)|\,\|p_1 \times p_2\|},$$

where the order function in the denominator has been ignored because its norm converges uniformly.

Since $\Delta(u,v,j) \geq \inf_{\Delta \in \Omega_0} |\Delta| > 0$ by Lemma 4.3 because the characteristic map is assumed to be *regular*, we obtain for the limit of normal vectors:

$$\lim_{m\to\infty} n^j_m(u,v) = \frac{p_1 \times p_2}{\|p_1 \times p_2\|} =: n(p) \tag{4.11}$$

The norm of the cross product of p_1 and p_2 is nonzero for almost every set of control points (see the following discussion). Hence, the surface is tangent plane continuous for almost every vector B_0 of control points. ■

In the previous lemma, the term "almost every" denotes a set of Lebesgue measure zero. If the cross-product is zero for p_1 and p_2, we know that the vectors are linearly dependent, i.e.

$$p_2 = \sigma p_1$$

for some scalar $\sigma \in \mathbb{R}$. If we fix p_1, the set of all multiples of p_1 forms a linear subspace, which has Lebesgue measure zero. Thus, ignoring these kinds of input data is justified.

4.2.1 A sufficient condition for regularity

At this point, we are ready to derive the main result of Reif's paper [Rei95]. It gives us conditions under which a subdivision scheme is *guaranteed* to produce smooth surfaces.

We will start with several auxiliary results that are required for the proof. First of all, we define a partial order for Jordan curves. This will be used for setting up a sequence of *boundary curves* of the characteristic map. Next, we show how to transform a parametrically smooth function by using the characteristic map. It will turn out that the transformed function lies in $C^1(\Gamma_0)$, which is the space of continuously differentiable functions over the set Γ_0. This fact enables us to use tools from real analysis. Last, we show that the set of all regular injective functions is open in $C^1(\Gamma_0)$. This will be useful for approximating a certain map in the proof of Theorem 4.3.

Definition 4.13 (A relation for Jordan curves)**.** Let c_1 and c_2 be Jordan curves in $\mathbb{R}^2$. By the Jordan curve theorem, the *interior* I of the curves is well-defined—see Fulton [Ful95], Theorem 5.10, pp. 68–69. We define a relation between c_1 and c_2 by writing $c_1 < c_2$ if and only if $c_1 \subseteq I(c_2)$.

Lemma 4.4. *The previous definition describes a* partial order *for Jordan curves.*

Proof. A partial order is reflexive, antisymmetric, and transitive. Reflexivity holds since the curve itself is a subset of its interior (more precisely, it is the *boundary* of the interior by the Jordan curve theorem).

For antisymmetry, we assume that we have $c_1 < c_2$ and $c_2 < c_1$ for some curves c_1 and c_2. Since $c_1 \subseteq I(c_2)$ and c_1 is the boundary of its interior (by the Jordan curve theorem), we have $I(c_1) \subseteq I(c_2)$. Likewise, we have $I(c_2) \subseteq I(c_1)$. Putting everything together, we arrive at $c_2 \subseteq I(c_1) \subseteq I(c_2) \subseteq I(c_1)$, which implies $I(c_1) = I(c_2)$. Since c_1 and c_2 do not have any self-intersections, this can only happen if $c_1 = c_2$.

In order to prove transitivity, let curves c_1, c_2, c_3 be given such that $c_1 < c_2$ and $c_2 < c_3$. Now $c_2 \subseteq I(c_3)$ by definition, which implies $I(c_2) \subseteq I(c_3)$ as above. Since $c_1 \subseteq I(c_2) \subseteq I(c_3)$, transitivity follows. Thus, Definition 4.13 defines a *partial order*. ∎

Lemma 4.5. *Let Ψ be regular and injective. If f is a parametrically smooth function over Ω_0, then $\widetilde{f} := f \circ \Psi^{-1}$ lies in $C^1(\Gamma_0)$, the space of continuously differentiable functions over the compact set Γ_0.*

Proof. Parametrical smoothness requires the cross boundary derivatives of surface segments sharing a common boundary to be equal up to sign; see for example Farin [Far96], Section 15.6, pp. 241–242, for more details.

Without loss of generality, we assume that the boundary curves $([1,2], 0, j)$ and $([1,2], 0, k)$ of some patches j and k have been identified by the neighbourhood relation (other boundary curves can be treated analogously). Let f be a parametrically smooth function over Ω_0. Since the neighbourhood relation identifies adjacent patches, we have

$$\begin{aligned} f(u,0,j) &= f(u,0,k) \\ Df(u,0,j) &= Df(u,0,k) \underbrace{\begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}}_{=:M}, \end{aligned} \tag{4.12}$$

where Df is the Jacobian matrix of f. This holds for the characteristic map Ψ, as well:

$$\begin{aligned} c(u) := \Psi(u,0,j) &= \Psi(u,0,k) \\ D\Psi(u,0,j) &= D\Psi(u,0,k)M \end{aligned} \tag{4.13}$$

Let $\widetilde{f}$ be the composition of the inverse of the characteristic map and f, i.e. $\widetilde{f} := f \circ \Psi^{-1}$. The *existence* of $D\widetilde{f}$ is plain because $\widetilde{f}$ is the composition of differentiable functions. Consequently, we only need to show that $D\widetilde{f}$ is a *continuous* function over Γ_0. In the interior of the patches, the inverse function theorem implies that Ψ is a diffeomorphism. Therefore, the chain rule can be applied and yields

$$D\widetilde{f}(x,y) = Df\left(\Psi^{-1}(x,y)\right)\left(D\Psi\left(\Psi^{-1}(x,y)\right)\right)^{-1}.$$

If we now let (x,y) approach the boundary curve $c(u)$ from patch j, the limit is given by

$$D\widetilde{f}(c(u)) = Df(u,0,j)\left(D\Psi(u,0,j)\right)^{-1}. \tag{4.14}$$

If (x,y) approaches $c(u)$ from patch k, however, the limit is given by

$$D\widetilde{f}(c(u)) = Df(u,0,k)\left(D\Psi(u,0,k)\right)^{-1}. \tag{4.15}$$

Starting with Equation 4.14, and using Equation 4.12 and Equation 4.13, which define the relationship between derivatives of adjacent surface segments, we obtain

$$\begin{aligned} Df(u,0,j)\left(D\Psi(u,0,j)\right)^{-1} &= Df(u,0,k)M\left(M^{-1}\left(D\Psi\left(u,0,k\right)\right)^{-1}\right) \\ &= Df(u,0,k)\left(D\Psi\left(u,0,k\right)\right)^{-1}, \end{aligned}$$

where we used that taking the inverse of matrix expressions changes the order of their arguments. Since the limits coincide, $D\widetilde{f}$ is well-defined and continuous on Γ_0 as a consequence of the inverse function theorem. ∎

The next lemma is a rather technical result. We will use it to show that a uniformly convergent sequence of functions with an injective limit function will consist of injective functions for all but finitely many elements of the sequence.

Lemma 4.6. *The set of all regular injective functions is open in* $C^1(\Gamma_0)$.

Proof. We continuously embed $C^1(\Gamma_0)$ in $C^{0,1}(\Gamma_0)$, which is the space of Lipschitz continuous functions. By definition of a continuous embedding, there is a constant $M \in \mathbb{R}$ such that

$$\sup \frac{\|g(x) - g(\widetilde{x})\|_2}{\|x - \widetilde{x}\|_2} =: \|g\|_{C^{0,1}(\Gamma_0)} \leq M \|g\|_{C^1(\Gamma_0)}$$

for all functions $g \in C^1(\Gamma_0)$. A regular and injective function $f \in C^1(\Gamma_0)$ can be inverted and its inverse will be continuously differentiable by the inverse function theorem. Hence, we have

$$f^{-1} \in C^1(f(\Gamma_0)) \subset C^{0,1}(f(\Gamma_0)).$$

By definition of Lipschitz continuity, the norm of f^{-1} must be finite, i.e.

$$\left\|f^{-1}\right\|_{C^{0,1}(f(\Gamma_0))} := \sup \frac{\left\|f^{-1}(y) - f^{-1}(\widetilde{y})\right\|_2}{\|y - \widetilde{y}\|_2} =: K < \infty.$$

As a consequence, by setting $x = f^{-1}(y)$, where $y = f(x)$, we obtain

$$\inf \frac{\|f(x) - f(\widetilde{x})\|_2}{\|x - \widetilde{x}\|_2} = \frac{1}{K}. \tag{4.16}$$

If we now perturb the function f by adding another function $g \in C^1(\Gamma_0)$, i.e. $\widetilde{f} := f + g$, we have

$$\begin{aligned} \inf \frac{\left\|\widetilde{f}(x) - \widetilde{f}(\widetilde{x})\right\|_2}{\|x - \widetilde{x}\|_2} &\geq \inf \frac{\|f(x) - f(\widetilde{x})\|_2}{\|x - \widetilde{x}\|_2} + \underbrace{\inf \frac{\|g(x) - g(\widetilde{x})\|_2}{\|x - \widetilde{x}\|_2}}_{\geq 0} \\ &\geq \inf \frac{\|f(x) - f(\widetilde{x})\|_2}{\|x - \widetilde{x}\|_2} - \sup \frac{\|g(x) - g(\widetilde{x})\|_2}{\|x - \widetilde{x}\|_2} \\ &\geq \frac{1}{K} - \|g\|_{C^{0,1}(\Gamma_0)} \\ &\geq \frac{1}{K} - M \|g\|_{C^1(\Gamma_0)}. \end{aligned}$$

Using Equation 4.16, we know that the last expression needs to be greater than zero in order for $\widetilde{f}$ to be an injective function. This is certainly satisfied if

$$\|g\|_{C^1(\Gamma_0)} = \left\|\widetilde{f} - f\right\|_{C^1(\Gamma_0)} < \frac{1}{MK}.$$

As a result, $\widetilde{f}$ remains *injective* for a function g that is sufficiently close to f.

To conclude, we prove regularity by straightforward continuity arguments: Since the determinant is a continuous function, the set of regular functions is open in $C^1(\Gamma_0)$. The intersection of two open sets remains open, so a suitable function g can always be found such that $\widetilde{f}$ is a *regular injective* function. ■

The next theorem is the main result of this section. It constitutes the first precise prediction of the behaviour of subdivision algorithms. Following Reif's approach [Rei95], we choose the characteristic map Ψ as a representation of the parameter space Ω_0 and apply the transformation described by Lemma 4.5. Consequently, the characteristic map will be transformed to the identity function on Γ_0 and we write

$$\widetilde{\Psi}(\xi, \eta) = (\xi, \eta)$$

in order to clarify this. Since $\Gamma_0 \subset \mathbb{R}^2$, the topology on Γ_0 is well-defined and we can use all the tools from real analysis.

Theorem 4.3. *Let $\lambda_1 = \lambda_2$ be a* real *eigenvalue with algebraic and geometric multiplicity 2 such that $1 > |\lambda_1| = |\lambda_2| > |\lambda_3|$. If the characteristic map is* regular *and* injective*, $\overline{\Sigma}$ is regular at p for almost every initial vector B_0 of control points.*

Proof. We first transform the problem to a more canonical form. Namely, by using an affine transformation we can always achieve that

$$p_0 = 0, \quad p_1 = e_1, \quad p_2 = e_2,$$

where 0 is the origin of the coordinate system and e_1, e_2, e_3 are the unit vectors of $\mathbb{R}^3$. Let $\lambda := \lambda_1 = \lambda_2$. As in the proof of Theorem 4.2, we use the parametrization of the prolongation and arrive at

$$x_m^j(u, v) = \begin{pmatrix} x(u, v, j, m) \\ y(u, v, j, m) \\ z(u, v, j, m) \end{pmatrix} = \begin{pmatrix} \lambda^m N(u, v, j)\psi_1 + o(\lambda^m) \\ \lambda^m N(u, v, j)\psi_2 + o(\lambda^m) \\ o(\lambda^m) \end{pmatrix},$$

where $x(u, v, j, m)$, $y(u, v, j, m)$, and $z(u, v, j, m)$ refer to the coordinate functions of the prolongation $x_m^j(u, v)$. Applying the transformation via Ψ^{-1} yields

$$\widetilde{x}_m^j(\xi, \eta) = \begin{pmatrix} x(\xi, \eta, m) \\ y(\xi, \eta, m) \\ z(\xi, \eta, m) \end{pmatrix} = \begin{pmatrix} \lambda^m \xi + o(\lambda^m) \\ \lambda^m \eta + o(\lambda^m) \\ o(\lambda^m) \end{pmatrix}.$$

In a small neighbourhood around the origin, the surface is located in the xy-plane because of the affine transformation. By Equation 4.11 from Theorem 4.2, the limit of the normal vectors is given by the unit vector e_3, i.e. $n(p) = e_3$. Thus, we try to find a *smooth parametrization* of the surface $\overline{\Sigma}$ in a

neighbourhood around the origin. This is possible if the projection function

$$\pi_M : \Gamma_0 \times \mathbb{N} \ni (\xi, \eta, m) \mapsto (x(\xi, \eta, m), y(\xi, \eta, m)) \in \mathbb{R}^2, \tag{4.17}$$

with $m \geq M$, is *injective*. If this is the case, π_M can be inverted locally, which yields

$$\pi_M^{-1} : \mathbb{R}^2 \ni (x, y) \mapsto (\xi, \eta, m) \in \Gamma_0. \tag{4.18}$$

A local parametrization is then given by setting

$$z = h(x, y) := z(\xi, \eta, m) = z(\pi_M^{-1}(x, y)). \tag{4.19}$$

For proving injectivity for π_M, we return to the characteristic map: The basis functions $N(u, v, j)$ and their partial derivatives are *continuous* functions over the *compact* set Γ_0. This implies that the basis functions attain their minimum and maximum, making them *bounded* functions. As a consequence, the order functions converge with respect to the norm on $C^1(\Gamma_0)$. Thus, we have

$$\lim_{m \to \infty} \left\| \lambda^{-m} \pi_M(\cdot, m) - \widetilde{\Psi} \right\|_{C^1(\Gamma_0)} = 0. \tag{4.20}$$

Put differently, π_M converges to $\widetilde{\Psi}$, which is *regular* and *injective* by assumption because Ψ is assumed to be regular and injective. Since the set of all regular injective functions is open by Lemma 4.6, π_M is injective for a fixed $m \geq M$.

In order to conclude the proof, we still need to show that the images of $\pi_M(\cdot, m)$ are *essentially disjoint*, i.e. that the intersection of images of maps m and $m + 1$ is equal to the common boundary curve—otherwise, the local parametrizations would coincide. Since Γ_0 is the image of a *representation* of the prolongations, only two closed disjoint boundary curves exist as a consequence of Definition 4.2.

Let α be the outer curve and β be the inner curve. The projection function $\pi_M(\cdot, m)$ is *regular*, so the boundary of π_M consists solely of the disjoint curves $\pi_M(\alpha, m)$ and $\pi_M(\beta, m)$ by the inverse function theorem. Therefore, all function values of $\pi_M(\cdot, m)$ lie in the region between the two curves. Furthermore, $\pi_M(\Gamma_0, m)$ is a *compact* and *connected* set. Hence, either $\pi_M(\alpha, m) < \pi_M(\beta, m)$ or $\pi_M(\beta, m) < \pi_M(\alpha, m)$. By definition, $\beta < \alpha$, and we have

$$\begin{aligned} \lim_{m \to \infty} \left\| \lambda^{-m} \pi_M(\alpha, m) - \widetilde{\alpha} \right\|_{C^1(\Gamma_0)} &= 0 \\ \lim_{m \to \infty} \left\| \lambda^{-m} \pi_M(\beta, m) - \widetilde{\beta} \right\|_{C^1(\Gamma_0)} &= 0 \end{aligned} \tag{4.21}$$

because $\lambda^{-m} \pi_M(\cdot, m)$ converges against $\widetilde{\Psi}$, as we have seen above. Thus, for $m \geq M$ sufficiently large, $\pi_M(\beta, m) < \pi_M(\alpha, m)$. Using Equation 4.21, we can define a sequence of boundary curves, namely,

$$\pi_M(\alpha, m), \pi_M(\beta, m) = \pi_M(\alpha, m + 1), \pi_M(\beta, m + 1) = \ldots$$

This sequence is *strictly* decreasing and converges to the origin because the subdivision algorithm is assumed to converge. Each of the regions defined by two curves corresponds to the image of π_M for some M. This implies that π_M is injective. The image Π_M of π_M is the interior of the outermost curve

$\pi_M(\alpha, M)$ without the origin, i.e.

$$\Pi_M = I(\pi_M(\alpha, M)) \setminus \{(0,0)\},$$

so $\Pi_M \cup \{(0,0)\}$ is a neighbourhood of the origin. Thus, we can parametrize $\overline{\Sigma}$ by setting

$$z = h(x,y) := \begin{cases} z(\pi_M^{-1}(x,y)) & \text{for } (x,y) \in \Pi_M \\ 0 & \text{for } (x,y) = (0,0) \end{cases}.$$

The function h is continuously differentiable for $(x,y) \neq (0,0)$ by the inverse function theorem because Ψ is assumed to be regular and Σ is assumed to be a smooth surface. Hence, the local parametrization of $\overline{\Sigma}$ around the origin is given by

$$\mathbb{R}^3 \ni (x,y,z) \mapsto \begin{pmatrix} x \\ y \\ h(x,y) \end{pmatrix}. \tag{4.22}$$

Using Equation 4.22 and Theorem 4.2, we can calculate the normal vector of Σ as (x,y) approaches the origin:

$$\begin{aligned} \lim_{(x,y)\to(0,0)} \mathrm{n}(x,y) &= \lim_{(x,y)\to(0,0)} \begin{pmatrix} 1 \\ 0 \\ h_x(x,y) \end{pmatrix} \times \begin{pmatrix} 0 \\ 1 \\ h_y(x,y) \end{pmatrix} \\ &= \lim_{(x,y)\to(0,0)} \frac{1}{\sqrt{1 + \|\nabla h(x,y)\|^2}} \begin{pmatrix} -h_x(x,y) \\ -h_y(x,y) \\ 1 \end{pmatrix} \\ &= e_3 \quad \text{(by Equation 4.11)} \end{aligned}$$

Hence, $h_x(x,y)$ and $h_y(x,y)$ tend to zero. Furthermore, the limit of the gradient in the equation above exists and is evaluated as

$$\lim_{(x,y)\to(0,0)} \nabla h(x,y) = (0,0).$$

Consequently, $h(x,y)$ is continuously differentiable for all $(x,y) \in \Pi_M \cup \{(0,0)\}$, and we have found a *smooth parametrization* of the limit surface for almost every initial vector of control points. ■

Having a sufficient smoothness condition at hand, we assume that the eigenvalues of the subdivision matrix satisfy the conditions of Theorem 4.3 from this point on. This leads to the following definition:

Definition 4.14 (Subdominant eigenvalue)**.** Let $\lambda_0, \lambda_1, \ldots$ be the eigenvalues of A, ordered by modulus. If $|\lambda_0| > |\lambda_1| = |\lambda_2| > |\lambda_3|$, we set $\lambda := \lambda_1 = \lambda_2$ and call λ the *subdominant eigenvalue* of A. From this point on, we shall employ this notation for all remaining proofs.

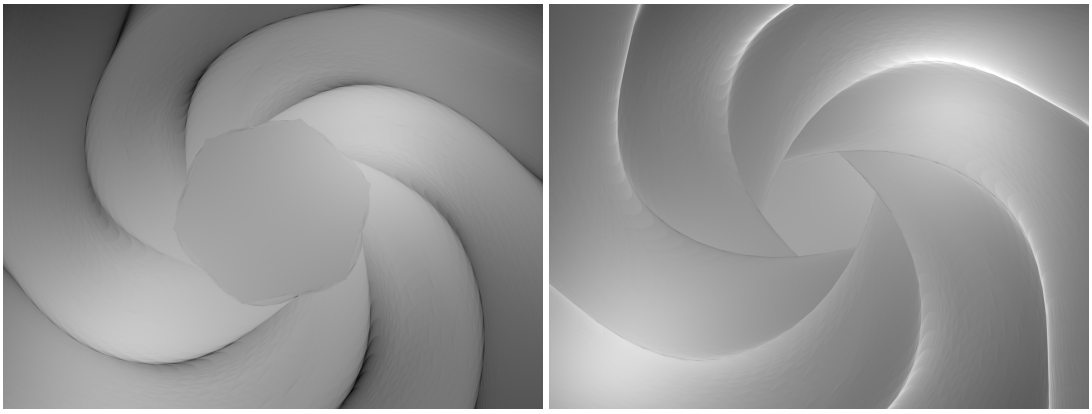


Figure 4.7: *Non-injective, irregular surface that results from using the Doo-Sabin algorithm with deliberately disturbed weights. The initial mesh is regular with a single extraordinary 6-sided face. Repeated application of the subdivision algorithm yields a surface that is wound up around the origin. The left image depicts the front side of the surface (lit from above) around the extraordinary face; the large face in the middle of the surface is not rendered properly due to self-intersections. The right image depicts the back side of the surface. Multiple sheets are approximating the central face while intersecting each other.*

4.2.2 A necessary condition for regularity

Theorem 4.3 describes sufficient requirements for obtaining smooth limit surfaces. In this section, we will see that injectivity of the characteristic map is also a *necessary* condition for smooth limit surfaces. Figure 4.7 depicts a non-injective, irregular surface, which is the consequence of violating this condition.

Theorem 4.4. *If $\Psi(u,v,j)$ is a point in the* interior *of $\Psi(\Omega_0)$ such that the characteristic map is non-injective at (u,v,j), then the limit surface Σ is not a regular C^1-manifold for almost every choice of initial data B_0.*

Proof. Since Ψ is not injective at (u,v,j), there is another point $(u',v',j') \neq (u,v,j)$ such that we have $\Psi(u,v,j) = \Psi(u',v',j')$. Let V_ϵ be an ϵ-neighbourhood of $\Psi(u,v,j)$ such that $V_\epsilon \subseteq \Psi(\Omega_0)$. This neighbourhood certainly exists because $\Psi(u,v,j)$ is assumed to lie in the interior. Since Ψ is a continuous function, there are neighbourhoods V and V' of (u,v,j) and (u',v',j') with

$$\Psi(V) = \Psi(V') = V_\epsilon.$$

Let $\widetilde{\Psi}$ be a continuous map that is sufficiently close to Ψ, i.e. $\left\|\Psi - \widetilde{\Psi}\right\|_\infty < \epsilon/2$, then, as a consequence of the continuity of $\widetilde{\Psi}$,

$$\widetilde{\Psi}(V) \cap \widetilde{\Psi}\left(V'\right) \neq \varnothing.$$

Thus, $\widetilde{\Psi}$ is also *not injective*. Expressing B_m in terms of eigenvectors yields:

$$\begin{aligned} B_0 &= \sum_{i=0}^{L} \psi_i p_i \\ &\vdots \\ B_m &= p_0 + \lambda^m \left(\psi_1 p_1 + \psi_2 p_2\right) + o(\lambda^m) \end{aligned}$$

We have already seen that the coefficients p_1 and p_2 are linearly independent for almost every choice of initial data. Thus, we can apply an affine transformation such that $p_0 = 0$ and $p_1 = e_1$ and $p_2 = e_2$. Since $\lambda \neq 0$, the surface layers can be rescaled in order to obtain simpler equations:

$$\widetilde{x}_m := \lambda^{-m} x_m$$
$$\lim_{m\to\infty} \widetilde{x}_m = e_1\psi_1 + e_2\psi_2 + o(1) = \begin{pmatrix} \Psi \\ 0 \end{pmatrix} + o(1)$$

We now assume that Σ is a *regular* C^1-manifold. The equation above implies that the tangent space at the origin is the xy-plane. Since the projection of $\widetilde{x}_m$ on the xy-plane converges to Ψ, it is non-injective for sufficiently large values of m. Thus, the projection of x_m to the xy-plane is non-injective near the origin for almost every m. But the projection of a regular C^1-manifold to its tangent space is *locally injective* because the partial derivatives are linearly independent (this allows application of the implicit function theorem). Consequently, Σ is *not* a regular C^1-manifold. ∎

4.3 Discrete Fourier transform and block-circulant matrices

The following is a digression about the discrete Fourier transform (DFT). Readers already well-versed with the DFT may skip it without remorse.

The DFT will prove to be an indispensable tool for the analysis of the subdivision matrix: Instead of analysing the complete matrix A, the DFT will calculate a matrix $\widehat{A}$ that is *similar* to A. Consequently, eigenvalues of $\widehat{A}$ will also be eigenvalues of A and we may calculate eigenvectors of A by a simple transformation of the corresponding eigenvectors of $\widehat{A}$.

From this point on, if not specifically mentioned otherwise, i denotes the *imaginary unit* of $\mathbb{C}$ rather than an index, i.e.

$$i := \sqrt{-1}.$$

In the following paragraphs, proofs were deliberately *not included* in order not to distract from the main material. The reader is referred to Davis [Dav94] for more details on circulant matrices and the DFT or to Lipson [Lip81] for a general introduction to several Fourier techniques.

Definition 4.15 (Fourier block matrix)**.** Let w_n be a *primitive root of unity*, i.e.

$$w_n := \exp\left(\frac{2\pi i}{n}\right) = \cos\left(\frac{2\pi}{n}\right) + i \sin\left(\frac{2\pi}{n}\right) =: c_n + i s_n, \tag{4.23}$$

and $\mathbb{1}$ be the *identity matrix* of sufficient size (depending on the subdivision matrix and n). Then the *Fourier block matrix*, which we will use in order to transform the subdivision matrix A, is defined as

$$\mathcal{W} := \left(w_n^{-jk}\right)_{j,k\in\mathbb{Z}_n} = \begin{pmatrix} \mathbb{1} & \mathbb{1} & \mathbb{1} & \dots & \mathbb{1} \\ \mathbb{1} & w_n^{-1}\mathbb{1} & w_n^{-2}\mathbb{1} & \dots & w_n^{1}\mathbb{1} \\ \mathbb{1} & w_n^{-2}\mathbb{1} & w_n^{-4}\mathbb{1} & \dots & w_n^{2}\mathbb{1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \mathbb{1} & w_n^{1}\mathbb{1} & w_n^{2}\mathbb{1} & \dots & w_n^{-1}\mathbb{1} \end{pmatrix}. \tag{4.24}$$

The inverse of $\mathcal{W}$ is given by

$$\mathcal{W}^{-1} = \frac{1}{n}\left(w_n^{jk}\mathbb{1}\right)_{j,k\in\mathbb{Z}_n} = \frac{1}{n}\overline{\mathcal{W}}. \tag{4.25}$$

Hence the jth block column of $\mathcal{W}^{-1}$ can be expressed by

$$\mathcal{W}_j^{-1} = \frac{1}{n}\begin{pmatrix} \mathbb{1} \\ w_n^j\mathbb{1} \\ \vdots \\ w_n^{(n-1)j}\mathbb{1} \end{pmatrix}. \tag{4.26}$$

Definition 4.16 (Discrete Fourier transform)**.** Let A be a matrix. The discrete Fourier transform (DFT) of A is defined as

$$\widehat{A} := \mathcal{W}A\mathcal{W}^{-1}.$$

We now take a look at a special class of matrices, the *block-circulant* matrices. These matrices have the helpful property that applying the DFT to them results in a *block-diagonal* matrix, i.e. a matrix that consists of block matrices of equal size around the main diagonal. The characteristic feature of block-circulant matrices is that they are fully determined by their first column.

Definition 4.17 (Block-circulant matrix)**.** Let $A_0, A_1, \ldots, A_{n-1}$ be $n \times n$ matrices. A matrix A is *block-circulant* if it can be written as

$$A = \begin{pmatrix} A_0 & A_{n-1} & \ldots & A_1 \\ A_1 & A_0 & \ldots & A_2 \\ \vdots & \vdots & \ddots & \vdots \\ A_{n-1} & A_{n-2} & \ldots & A_0 \end{pmatrix}.$$

More formally, we will also write $A = \operatorname{circ}(A_0, \ldots, A_{n-1})$. In some textbooks, this notation is employed to signify a *circulant* matrix, i.e. a matrix where the A_j have been replaced by vectors.

We may construct a block-circulant matrix by writing down the initial column of matrices and shifting each entry one row down.

Theorem 4.5. *Let $A = \operatorname{circ}(A_0, \ldots, A_{n-1})$ be a block-circulant matrix and $\widehat{A}$ the result of the DFT. Then $\widehat{A}$ is decomposed into blocks of matrices around the main diagonal, i.e.*

$$\widehat{A} = \operatorname{diag}(\widehat{A}_0, \ldots, \widehat{A}_{n-1}) = \begin{pmatrix} \widehat{A}_0 & & 0 \\ & \ddots & \\ 0 & & \widehat{A}_{n-1} \end{pmatrix}.$$

with $n \times n$ matrices $\widehat{A}_0, \widehat{A}_1, \ldots, \widehat{A}_{n-1}$. Put differently, a block-circulant matrix is "block-diagonalized" by applying a DFT.

Proof. See Davis [Dav94], Theorem 3.2.2, pp. 72–73. ∎

If A is a block-circulant matrix, the blocks on the diagonal of $\widehat{A}$ are obtained by applying the Fourier matrix $\mathcal{W}$ to the first block-column of A, i.e.

$$\begin{pmatrix} \widehat{A}_0 \\ \vdots \\ \widehat{A}_{n-1} \end{pmatrix} = \mathcal{W} \begin{pmatrix} A_0 \\ \vdots \\ A_{n-1} \end{pmatrix}.$$

Written more compactly, we have

$$\widehat{A}_k = \sum_{j \in \mathbb{Z}_n} w_n^{-jk} A_j. \tag{4.27}$$

Since the matrices A and $\widehat{A}$ are *similar* by definition, we can restrict the analysis of a subdivision algorithm to the *blocks* $\widehat{A}_j$ rather than having to examine the whole matrix. We still need to see how to determine an eigenvector of A from eigenvectors of $\widehat{A}_j$. To this end, we refer to a calculation by Peters and Reif from [PR08], Section 5.4, p. 99: Let $\widehat{v}$ be an eigenvector of a block $\widehat{A}_j$. The corresponding eigenvector of A is given by

$$v = \frac{1}{n} \begin{pmatrix} w_n^0 \widehat{v} \\ w_n^j \widehat{v} \\ \dots \\ w_n^{(n-1)j} \widehat{v} \end{pmatrix}, \tag{4.28}$$

which can be abbreviated to $v = \mathcal{W}_j^{-1} \widehat{v}$ by using the expression for the jth block-column of the inverse transformation matrix.

Consequences of block-circulant matrices If the subdivision matrix of a subdivision algorithm is *block-circulant*, the equation of a subdivision process as introduced by Definition 4.10 is simplified to

$$B_{m+1}^k = \sum_{j=0}^{n-1} A_{k-j} B_m^j. \tag{4.29}$$

There is an intuitive approach to this equation: Let several patches of control points be given. We assume that they are ordered counter-clockwise. Then, starting from a patch j, the patch to the left (with coefficients $j+1$) will be weighted with the matrix A_{n-1}, whereas the patch to the right (with coefficients $j-1$) will be weighted with the matrix A_1. At last, the central patch (with coefficients j) will be weighted with the matrix A_0. See Figure 4.6 for an illustration. This point of view will come in handy when analysing subdivision schemes.

4.4 Symmetric subdivision algorithms

In the following discussions, we shall take advantage of two basic properties of characteristic maps, which follow from their definition.

Definition 4.18 (Properties of characteristic maps)**.** Ψ and $A\Psi = \lambda\Psi$ join *smoothly*. Thus, we have for $t \in [0,1]$:

$$\Psi^j(1,t) = \lambda\Psi^j(2,2t) \tag{4.30}$$

$$\Psi^j(t,1) = \lambda\Psi^j(2t,2) \tag{4.31}$$

Moreover, all adjacent segments of the characteristic map join *continuously*. Hence, we have for $t \in [1,2]$:

$$\Psi^j(0,t) = \Psi^{j+1}(t,0) \tag{4.32}$$

The subdivision algorithms analysed in this thesis are reasonable in the sense that the current labelling of the control points is *irrelevant*: In order to calculate the refined control points, only the neighbourhood of the point is taken into account and not, for example, the *direction* of traversal. This property simplifies calculations—consequently, we define *symmetric subdivision algorithms* for which it holds. Most subdivision algorithms, such as the Doo-Sabin, the Catmull-Clark, and the Loop scheme, are symmetric subdivision algorithms.

Definition 4.19 (Symmetric subdivision algorithms)**.** A subdivision algorithm is *symmetric* if it is invariant under both shifts and reflections of the labelling of the vector of control points B_m. Shifts and reflections are described by matrices S and R that are characterized by:

$$N(u,v,j+1)B_m = N(u,v,j)SB_m \tag{4.33}$$

$$N(v,u,-j)B_m = N(u,v,j)RB_m \tag{4.34}$$

Since a symmetric subdivision algorithm is invariant under both shifts and reflections, the subdivision matrix A needs to commute with R and S:

$$SA = AS \tag{4.35}$$

$$RA = AR \tag{4.36}$$

The matrix S can be expressed as

$$S = \begin{pmatrix} 0 & 0 & \dots & 0 & \mathbb{1} \\ \mathbb{1} & 0 & \dots & 0 & 0 \\ 0 & \mathbb{1} & \dots & 0 & 0 \\ & & \ddots & & \\ 0 & 0 & \dots & \mathbb{1} & 0 \end{pmatrix}, \tag{4.37}$$

where $\mathbb{1}$ is the identity matrix. For the matrix R, the closed form depends on the subdivision scheme. Fortunately, we only need to assume that a matrix R does *exist*—its precise description is *completely* irrelevant and all proofs will only require the properties outlined above.

The reason for restricting our analysis to symmetric subdivision schemes is illustrated by the next proposition: The matrix of a symmetric subdivision algorithm is block-circulant, which implies that we can diagonalize it by the DFT. More intuitively, block-circulance means that the algorithm performs the same calculations regardless of the choice of start vertex.

Proposition 4.1. *The subdivision matrix of a symmetric subdivision algorithm is* block-circulant.

Proof. Since the algorithm is symmetric, matrices A and S commute. This results in

$$(AS)_{j,i} = A_{j,i+1} = A_{j-1,i} = (SA)_{j,i} \tag{4.38}$$

for $(i, j) \in \mathbb{Z}_n$, where AS and SA have been decomposed into block matrices $(AS)_{j,i}$. Using this equality, we see that

$$A_j := A_{j,0} = A_{j+1,1} = A_{j+2,2} = \cdots = A_{j+i,i}.$$

Thus, $A = \operatorname{circ}(A_0, \ldots, A_{n-1})$. ∎

By limiting the analysis to symmetric schemes, a necessary condition for the subdominant eigenvalue emerges. The next theorem states this in more precise terms. Following Peters and Reif [PR08], we will call the subdominant eigenvalue the *Fourier index* in this context.

Theorem 4.6. *Let the characteristic map Ψ of a symmetric scheme be regular. Let $\widehat{A}_0, \ldots, \widehat{A}_{n-1}$ denote the diagonal blocks of the transformed subdivision matrix $\widehat{A}$. If the subdominant eigenvalue λ is* not *an eigenvalue of $\widehat{A}_1$ and $\widehat{A}_{n-1}$, then Ψ is* non-injective.

Proof. Matrices A and $\widehat{A}$ are similar by definition of the DFT. Furthermore, $\widehat{A}$ contains only block matrices along its diagonal. Thus, λ is an eigenvalue of A if and only if it is an eigenvalue $\widehat{A}_k$ for $k \in \{0, \ldots, n-1\}$. Since A is real, we have $\widehat{A}_{n-k} = \widehat{A}_k^*$, where in this case, $\widehat{A}_k^*$ denotes the complex conjugate of $\widehat{A}_k$ (for typographical reasons). Consequently, λ is also an eigenvalue of $\widehat{A}_{n-k}$. Let $\widehat{\psi}$ be an *eigenvector* of $\widehat{A}_k$ corresponding to the eigenvalue λ, then

$$\psi_* := [w_n^0 \widehat{\psi}, w_n^k \widehat{\psi}, \ldots, w_n^{k(n-1)} \widehat{\psi}] \tag{4.39}$$

is a complex eigenvector of A by Equation 4.28.

Since ψ_* contains roots of unity as coefficients for $\widehat{\psi}$ and the components of ψ_* serve as control points for the characteristic map, *all* segments Ψ_*^j of the characteristic map can be represented as multiples of the first segment. More precisely,

$$\Psi_*^j = w_n^{jk} \Psi_*^0. \tag{4.40}$$

Let τ_* be the restriction of Ψ_* to the outer boundary of Ω_0. Then τ_* consists of segments $\tau_*^0, \ldots, \tau_*^{n-1}$. If we assume that Ψ_* is *injective*, τ_* parametrizes a Jordan curve in $\mathbb{C}$. Furthermore,

$$\Psi_*^0(2,2) \neq \Psi_*^0(1,1) \overset{\text{Eq. 4.30}}{=} \lambda \Psi_*^0(2,2)$$

because Ψ_* is assumed to be injective. Therefore, the image of τ_* does not contain the origin. As a consequence, we can apply the residue theorem and calculate the *winding number* of τ_* with respect to the origin as

$$\frac{1}{2\pi i} \oint_{\tau_*} \frac{dz}{z} = \frac{1}{2\pi i} \sum_{j \in \mathbb{Z}_n} \int_{\tau_*^j} \frac{dz}{z} = \frac{n \ln w_n^k}{2\pi i} = k.$$

If $k \notin \{0, 1, n-1\}$, the curve has self-intersections ($k = n-1$ would imply that the surface winds around the origin *once* in clockwise direction), thereby contradicting the injectivity of Ψ_*. The case of $k = 0$ can be ignored because by Equation 4.40, all segments would have to be equal. ∎

We have already seen that injectivity is a requirement for smooth surfaces, so we can postulate that, given all the ambiguity in choosing eigenvectors for the subdominant eigenvalue, one characteristic map is singled out. We call this map the *normalized characteristic map*.

Definition 4.20 (Normalized characteristic map)**.** We say that the characteristic map Ψ is *normalized* if the eigenvector $\widehat{\psi}$ is scaled such that

$$\Psi^0(2,2) = (d, 0) \tag{4.41}$$

with $d > 0$. If Ψ is injective, normalization is always possible because of the relation

$$\Psi^0(2,2) \neq \Psi^0(1,1) = \lambda \Psi^0(2,2),$$

which implies that $\Psi^0(2,2) \neq 0$. Thus, there is $\rho \in \mathbb{C}$ such that $\rho\Psi^0(2,2) = (d, 0)$ with $d > 0$.

The appeal of normalized characteristic maps is explained by the following lemma. Note that for $j = 0$, the lemma implies symmetry about the real axis.

Lemma 4.7. *If Ψ_* is a* normalized characteristic map *corresponding to a symmetric subdivision scheme, then*

$$\Psi_*^j(u, v) = \overline{\Psi}_*^{-j}(v, u). \tag{4.42}$$

Proof. The subdivision scheme is supposed to be symmetric. Using Definition 4.19, the following equations can be obtained:

$$N(u, v, j)S^{-1}RB_m = N(u, v, j-1)RB_m = N(v, u, 1-j)B_m \tag{4.43}$$

$$N(u, v, j)RSB_m = N(v, u, -j)SB_m = N(v, u, 1-j)B_m \tag{4.44}$$

Since the basis functions are assumed to be *linearly independent*, Equation 4.43 and Equation 4.44 imply

$$RS = S^{-1}R, \tag{4.45}$$

which will be useful later on.

Let ψ_* be an eigenvector of A. Then $R\psi_*$ is also an eigenvector of A or the zero vector because $RA = AR$ for symmetric algorithms, so

$$AR\psi_* = RA\psi_* = R\lambda\psi_* = \lambda R\psi_*. \tag{4.46}$$

Since $\psi_* := \psi_1 + i\psi_2$, there must be coefficients $a, b \in \mathbb{C}$ such that

$$R\psi_* = a\psi_* + b\overline{\psi}_*. \tag{4.47}$$

Furthermore, shifting the eigenvector ψ_* corresponds to a multiplication with w_n because w_n is a primitive root of unity, so

$$S\psi_* = w_n\psi_*. \tag{4.48}$$

Using Equations 4.47 and 4.48, we try to determine the coefficients a, b for $R\psi_*$:

$$\begin{aligned} S^{-1}R\psi_* &= S^{-1}\left(a\psi_* + b\overline{\psi}_*\right) && \text{(by Equation 4.47)} \\ &= a\overline{w}_n\psi_* + bw_n\overline{\psi}_* && \text{(by Equation 4.48)} \\ RS\psi_* &= Rw_n\psi_* && \text{(by Equation 4.48)} \\ &= aw_n\psi_* + bw_n\overline{\psi}_* && \text{(by Equation 4.47)} \end{aligned}$$

By Equation 4.45, the expressions above must be equal, which yields

$$a\overline{w}_n\psi_* + bw_n\overline{\psi}_* = aw_n\psi_* + bw_n\overline{\psi}_*,$$

from which we conclude that $a = 0$ because roots of unity are nonzero and $\overline{w}_n \neq w_n$. As an intermediate result, we now know that

$$R\psi_* = b\overline{\psi}_*. \tag{4.49}$$

In order to determine b, we take a look at $\Psi_*^0(2,2)$. Since the characteristic map is assumed to be *normalized*, there is $d \in \mathbb{R}$ with $d > 0$ such that $d = N(2,2,0)\psi_*$. By definition of symmetric schemes, R can be applied and we can use the expression for $R\psi_*$ derived above:

$$\begin{aligned} d &= N(2,2,0)R\psi_* \\ &= bN(2,2,0)\overline{\psi}_* \\ &= bd \qquad \text{(since the basis functions are real)} \end{aligned}$$

Thus, $b = 1$ and $R\psi_* = \overline{\psi}_*$. This allows us to conclude the proof, starting with the definition of the jth segment of the characteristic map:

$$\Psi_*^j(u,v) = N(u,v,j)\psi_*$$

Again, R can be applied and $R\psi_* = \overline{\psi}_*$ can be used. This yields

$$\begin{aligned} \Psi_*^j(u,v) &= N(u,v,-j)R\psi_* \\ &= N(v,u,-j)\overline{\psi}_*, \end{aligned}$$

which, by definition of Ψ, can be written as

$$\Psi_*^j(u,v) = \overline{\Psi}_*^{-j}(v,u).$$

■

Corollary 4.1. *For $j = 0$, the previous lemma states that $\Psi_*^0(u,v) = \overline{\Psi}_*^0(v,u)$.*

We now have seen that the first segment of normalized characteristic maps is symmetric with respect to the real axis. Consequently, from this point on, we assume that the characteristic map of a subdivision scheme is *normalized*.

4.5 Criteria for regularity and injectivity of the characteristic map

The proofs in the remaining part of this chapter all concern the *regularity* and *injectivity* of the characteristic map. We will derive several criteria that are easily verified without having to resort to long and tedious computations.

The next lemma is very technical: We define a function μ that assigns each point the number of its preimages. Afterwards, we proceed to prove several properties of this function—this will help in later proofs. In particular, if $\mu = 1$ for the image of the characteristic map, injectivity holds.

Lemma 4.8. *For regular Ψ, the function μ that assigns each point in $\Psi^0(\Omega_0)$ its number of preimages is upper semi-continuous everywhere and lower semi-continuous on the interior. μ is defined by:*

$$\begin{aligned}\mu : \Psi^0(\Omega_0) &\to \mathbb{N} \\ (x, y) &\mapsto \#\left\{(u, v) \in \Omega_0 \mid \Psi^0(u, v) = (x, y)\right\}\end{aligned}$$

Proof. Let y be a point in $\Psi(\Omega_0)$ and x_i its preimages, i.e.

$$\Psi(x_i) = y, \text{ where } i \in \{1, \dots, \mu(y)\}.$$

By the inverse function theorem, there is an open neighbourhood V of y and open neighbourhoods U_i of x_i such that $\Psi(U_i) = V$. Assume there is a sequence $(\widetilde{y}_n)$ that converges against y such that $\mu(\widetilde{y}_i) > \mu(y)$ for $i \in \mathbb{N}$. We can then consider the corresponding preimages $\widetilde{x}_i$ with $\Psi(\widetilde{x}_i) = \widetilde{y}_i$. Let $U_* := \bigcup U_i$ be the union of the open sets U_i as defined above. The preimages $\widetilde{x}_i$ are not elements of U_* because, by the inverse function theorem, this would imply $\mu(\widetilde{y}_i) = \mu(y)$. Since Ψ is continuous, the accumulation point $\widetilde{x}_i^*$ of the sequence of preimages must be mapped to y, i.e. $\Psi(\widetilde{x}_i^*) = y$. But $\widetilde{x}_i^*$ cannot be an element of U_* because $\Omega_0 \setminus U_* \ni \widetilde{x}_i$ is closed. This means that we have found *another* preimage of y. Thus, $\mu(y)$ is larger than we assumed—which is a contradiction.

We now show that μ is lower semi-continuous at the interior of the image. Let x be a point in the interior of $\Psi(\Omega_0)$. Since the images of the interior and of the boundary do not coincide by the inverse function theorem, we can choose a preimage which lies in the interior. Furthermore, the neighbourhood V as introduced above can be chosen very small such that all open sets around the preimages also lie in the interior of $\Psi^0(\Omega_0)$. Hence, $\mu(V) \geq \mu(x)$. ∎

The function μ as defined above will be useful for the next proof. In essence, we can show that, given a *regular* characteristic map, it is sufficient to prove injectivity on the *boundary* only.

Lemma 4.9. *Let $\partial\Omega_0$ be the boundary of Ω_0 and Ψ^0_∂ be the restriction of Ψ^0 to $\partial\Omega_0$. If $\Psi^0 = [\Psi^0_1, \Psi^0_2]$ is* regular, *then Ψ^0 is injective if and only if Ψ^0_∂ is injective. Recalling the definition of regularity, this means*

that the Jacobian determinant is nonzero everywhere, i.e.

$$\det D\Psi^0 = \det \begin{pmatrix} \Psi^0_{1,u} & \Psi^0_{1,v} \\ \Psi^0_{2,u} & \Psi^0_{2,v} \end{pmatrix} \neq 0.$$

Proof. We assume that Ψ^0 is *regular* and Ψ^0_∂ is *injective*. By the inverse function theorem, points in the *interior* of Ω_0, which we denote by $\mathring{\Omega}_0$, are mapped into the interior of $\Psi^0(\Omega_0)$, i.e.

$$\partial\Psi^0(\Omega_0) \cap \Psi^0(\mathring{\Omega}_0) = \varnothing. \tag{4.50}$$

Let μ be as defined in Lemma 4.8. Since Ψ^0_∂ is assumed to be injective, we may use Equation 4.50 to obtain $\mu(\partial\Psi^0(\Omega_0)) = 1$. Furthermore, since μ is upper semi-continuous by Lemma 4.8, $\mu(\Psi^0(\Omega_0)) = 1$, and thus, Ψ is *injective*. ∎

The next lemma is crucial for showing injectivity for *all* segments of the characteristic map. Its claim is that the segment Ψ^0_* lies in a sector of the complex plane. Since all segments of the characteristic map are related by *rotations*, this lemma will be used as a stepping stone towards proving global injectivity under certain assumptions.

Lemma 4.10. *If Ψ^0 is regular and both of the partial derivatives $\Psi^0_{1,v}(1,t)$ and $\Psi^0_{2,v}(1,t)$ are* positive *for $t \in [0,1]$, then Ψ^0 is located in a sector of angle $2\pi/n$ in the complex plane, i.e.*

$$\Psi^0(u,v) \subset S_n := \{x \in \mathbb{C} \mid -\pi/n \leq \arg x \leq \pi/n\}$$

for all $(u,v) \in \Omega_0$.

Proof. By Corollary 4.1, $\Psi^0(u,v) = \overline{\Psi}^0(v,u)$. Hence, the components of the characteristic map are related by

$$\begin{aligned} \Psi^0_1(u,v) &= \Psi^0_1(v,u) \\ \Psi^0_2(u,v) &= -\Psi^0_2(v,u) \end{aligned} \tag{4.51}$$

and in particular, $\Psi^0_2(t,t) = 0$ for $t \in [1,2]$.

We define $p_*(t) := p_1(t) + ip_2(t) : \Psi^0_*(1,t)$ for $t \in [0,1]$ in order to decompose the characteristic map. Since $\Psi^0_{2,v} > 0$ and p_2 describes the second component of Ψ_*, the function p_2 must be *strictly increasing*. We have the following estimate for its values:

$$\Psi^0_2(1,0) = p_2(0) < p_2(t) < p_2(1) = \Psi^0_2(1,1) = 0 \tag{4.52}$$

Keeping in mind that multiplying by w_n corresponds to a rotation by $2\pi/n$ in the complex plane, we now calculate the argument of $\Psi^0_*(t,0)$ for $t \in [1,2]$:

$$w_n\Psi^0_*(t,0) \overset{\text{Eq. 4.40}}{=} \Psi^1_*(t,0) \overset{\text{Eq. 4.32}}{=} \Psi^1_*(0,t) \overset{\text{Cor. 4.1}}{=} \overline{\Psi}^0_*(t,0)$$

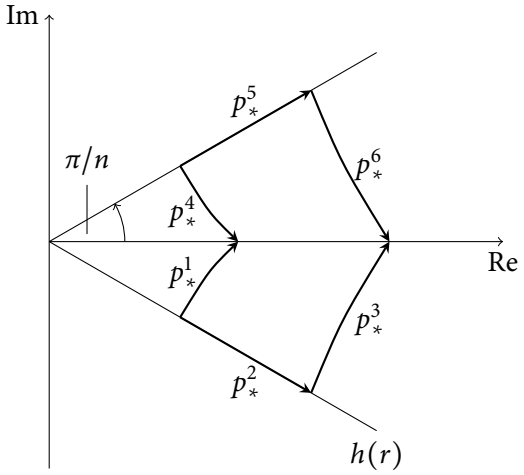


Figure 4.8: *A sketch of the curves $p_*^1, \ldots, p_*^6$ that are used in the proof of Lemma 4.11. The ray $h(r)$ is defined in the proof of Lemma 4.10. Note that the upper and lower boundary curves have been slightly contorted because, in general, they do* not *lie on a straight line.*

This implies that either $\arg \Psi_*^0(t,0) = -\pi/n$ or $\arg \Psi_*^0(t,0) = \pi - \pi/n$. The latter case implies that $p_2(0) > 0$, which contradicts Equation 4.52. Consequently,

$$\arg \Psi_*^0(t,0) = -\pi/n \tag{4.53}$$

$$\arg \overline{\Psi}_*^0(0,t) = \pi/n, \tag{4.54}$$

so $\Psi^0(t,0)$ is a part of the ray $h(r) := r\exp(-i\pi/n)$ with $r > 0$.

Since the partial derivatives are positive, both $p_1(t)$ and $p_2(t)$ are *strictly increasing* (we know this already for p_2). Thus, p_* only intersects h at the origin and it follows that

$$-\pi/n = \arg p_*(0) < \arg p_*(t) < \arg p_*(1) \tag{4.55}$$

for $t \in (0,1)$.

Summing up, we have shown that $\Psi^0(t,0) \in S_n$ for $t \in [1,2]$ and $\Psi^0(1,t) \in S_n$ for $t \in [0,1]$. Since $\overline{\Psi}_*^0(t,1) = \Psi_*^0(1,t)$ by Corollary 4.1 and complex conjugation corresponds to reflection at the real axis, we have $\overline{\Psi}_*^0(t,1) \subset S_n$. Similarly, by using the scaling properties from Equation 4.30 and Equation 4.31 as well as symmetry about the real axis, we can prove that $\Psi^0(\partial\Omega_0) \subset S_n$.

It remains to be shown that $\Psi^0(\Omega_0) \subset S_n$. Since Ψ^0 is assumed to be *regular*, the inverse function theorem shows that $\partial\Psi^0(\Omega_0) \subseteq \Psi^0(\partial\Omega_0)$. However, $\Psi^0(\partial\Omega_0) \subset S_n$, as we have already shown. Let us assume that there is a point in the *interior* of $\Psi^0(\Omega_0)$ that does not lie in the sector. As a consequence, *all* points from the interior of $\Psi^0(\Omega_0)$ would need to lie *outside* the sector because $\partial\Psi^0(\Omega_0) \subset S_n$. Since $\mathbb{C} \setminus S$ is *unbounded*, this would contradict the *compactness* of $\Psi^0(\Omega_0)$. Hence $\Psi^0(u,v) \subset S_n$ for all (u,v). ∎

At this point, we have almost enough tools in order to prove global injectivity of the characteristic map. In fact, there is a handy criterion for showing that the segment Ψ^0 is injective. By using the rotational symmetry, as outlined above, global injectivity will be proven afterwards.

Lemma 4.11. *If* Ψ^0 *is* regular *and* $\Psi^0_{1,\nu}(1,t)$ *and* $\Psi^0_{2,\nu}(1,t)$ *are* positive *for* $t \in [0,1]$*, then* Ψ^0 *is* injective.

Proof. By Lemma 4.9, it is sufficient to show that the restriction of Ψ^0 to the boundary of Ω_0 is injective. We denote this restriction by Ψ^0_∂. In order to prove injectivity, we define several *boundary curves*:

$$p^1_* := \Psi^0_*(1,t), \quad p^2_* := \Psi^0_*(1+t,0), \quad p^3_* := \Psi^0_*(2,2t)$$
$$p^4_* := \Psi^0_*(t,1), \quad p^5_* := \Psi^0_*(0,1+t), \quad p^6_* := \Psi^0_*(2t,2)$$

for $t \in [0,1]$. Figure 4.8 depicts a sketch of these curves. Recalling the definition of $p_*(t) = p_1 + ip_2$ from the proof of the previous lemma, the following relations hold:

$$p_* \overset{\text{Def.}}{=} p^1_* \overset{\text{Lem. 4.7}}{=} \overline{p}^4_* \overset{\text{Eq. 4.30}}{=} \lambda p^3_* \overset{\text{Lem. 4.7}}{=} \lambda \overline{p}^6_*$$
$$p^2_* \overset{\text{Lem. 4.7}}{=} \overline{p}^5_*$$

Using Equations 4.53 and 4.54, we see that

$$\arg p^2_* = -\frac{\pi}{n} \quad \text{and} \quad \arg p^5_* = \frac{\pi}{n},$$

and consequently, the curves p^2_* and p^5_* do *not* have any intersections. The curves also do not have any self-intersections because, as we have seen in the proof of Lemma 4.10, they are regularly parametrized parts of rays in the complex plane.

In order to state properties of the other curves, we require that $\arg p_*(t)$ is monotone. By Equation 4.52, $p_2(0) < 0$ and $p_2(1) = 0$. Using Equation 4.53, we see that $p_1(0) \geq 0$ must hold. Due to the assumptions about the partial derivatives, $p_1(t)$ and $p_2(t)$ are *strictly increasing*; hence $p_1(t) \geq 0$ and $p_2(t) \leq 0$. Since $p_1(t)$ and $p_2(t)$ do not vanish simultaneously,

$$\frac{d}{dt}(\arg p_*) = \frac{d}{dt} \arctan\left(\frac{p_2(t)}{p_1(t)}\right) = \frac{p_1 p_2' - p_1' p_2}{p_1^2 + p_2^2} > 0.$$

The monotonicity of $\arg p_*$ has several consequences: Since $\arg p_* = \arg p^1_* = \arg p^3_*$ by definition of the curves and $\arg p_* = -\arg p^4_* = -\arg p^6_*$ due to the complex conjugation, the curves p^1_*, p^3_*, p^4_*, p^6_* cannot have self-intersections. Additionally, the curves p^1_* and p^3_* cannot intersect because $p^1_* = \lambda p^3_*$ with $\lambda \neq 1$ and their arguments are monotonically increasing. The same argument also holds for the curves p^4_* and p^6_*. Furthermore, since the argument of p^2_* remains *fixed*, the only intersections of p^1_* and p^3_* with p^2_* are at the beginning and the end of the curve, i.e.

$$\Psi^0_*(1,0) = p^1_*(0) = p^2_*(0)$$
$$\Psi^0_*(2,0) = p^3_*(0) = p^2_*(1).$$

Likewise, the curves p^4_* and p^6_* only coincide with p^5_* at its beginning and its end.

As a last step, we see that due to the monotonicity, the union of all *lower* boundary curves, $p^1_* \cup p^2_* \cup p^3_*$, only has 2 intersections with the union of all *upper* boundary curves, $p^4_* \cup p^5_* \cup p^5_*$, namely,

$$\Psi^0_*(1,1) = p^1_*(1) = p^4_*(1) \quad \text{and} \quad \Psi^0_*(2,2) = p^3_*(1) = p^6_*(1).$$

In conclusion, we have shown that it is possible to parametrize the boundary of the first segment of the characteristic map by non-intersecting injective curves. Thus, Ψ^0_∂ is *injective*. ∎

As a culmination of the efforts in the previous sections, we obtain a theorem that guarantees injectivity provided that the partial derivatives are positive. Since this fact can be checked quite easily, the theorem will become very useful when analysing a subdivision algorithm.

Theorem 4.7. *If* Ψ^0 *is* regular *and both* $\Psi^0_{1,v}(1,t)$ *and* $\Psi^0_{2,v}(1,t)$ *are* positive *for* $t \in [0,1]$*, then the characteristic map* Ψ *is* regular *and* injective.

Proof. By Lemma 4.11, the first segment Ψ^0 of the characteristic map is regular and injective. In Equation 4.40, we have already seen that the segment Ψ^j of the characteristic map can be obtained from Ψ^0 by a rotation around the origin with angle $2\pi j/n$. As a consequence of this rotational symmetry, the segments Ψ^j with $j \in \mathbb{Z}_n$ are regular and injective. By Lemma 4.10, the segments Ψ^j do not overlap because

$$(2j-1)\,\frac{\pi}{n} \le \arg \Psi^j_* \le (2j+1)\,\frac{\pi}{n}$$

for $j \in \mathbb{Z}_n$.

Since the common boundaries of adjacent segments are identified by the neighbourhood relation, the union of all segments Ψ^j remains regular and injective. Thus, the characteristic map Ψ is *regular* and *injective*. ∎

Using this theorem, we can obtain a corollary that determines *regularity* and *injectivity* simultaneously while having rather weak requirements.

Corollary 4.2. *If* $\Psi^0_{1,v}$ *and* $\Psi^0_{2,v}$ *are* positive *for all values in* Ω_0*, then the characteristic map* Ψ *is* regular *and* injective.

Proof. We first prove that the Jacobian determinant does not vanish: Using Equation 4.51, the Jacobian matrix can be rewritten as

$$D\Psi^0(u,v) = \begin{pmatrix} \Psi^0_{1,v}(v,u) & \Psi^0_{1,v}(u,v) \\ -\Psi^0_{2,v}(v,u) & \Psi^0_{2,v}(u,v) \end{pmatrix}, \tag{4.56}$$

which has a *positive* determinant if $\Psi^0_{1,v} > 0$ and $\Psi^0_{2,v} > 0$. Thus, Ψ^0 is regular and the conditions for applying Theorem 4.7 are met. ∎

4.6 Summary

In this chapter, we derived methods for analysing the smoothness properties of linear stationary subdivision schemes. To this end, we introduced the *characteristic map* of a subdivision scheme and proved that the scheme generates regular surfaces as long as the characteristic map is *regular* and *injective*. We then proceeded to show that injectivity is *mandatory*—otherwise, the limit surface will be irregular. Moreover, we introduced the discrete Fourier transform (DFT) and showed how it simplifies the analysis of the subdivision matrix. It turned out that the *subdominant* eigenvalue of a subdivision algorithm needs to be an eigenvalue of the transformed matrices $\widehat{A}_1$ and $\widehat{A}_{n-1}$. Peters and Reif [PR08] designate this the

Fourier index of the eigenvalue. We then ended the chapter by deriving some conditions for checking *regularity* and *injectivity* of the characteristic map. As a result, we saw that in some cases, only the sign of the partial derivatives needs to be checked.

5 Case studies of subdivision algorithms

In this chapter, the methods that we have previously derived will be put to use: We will analyse the smoothness properties of the Doo-Sabin and the Catmull-Clark subdivision algorithms. To this end, we will first study the schemes in their originally described form. Following this, we will examine the consequences of using degenerated weights and calculate permissible ranges for the weights of both algorithms. For the Catmull-Clark scheme, we will derive a graphical representation that describes sufficient conditions for the weights such that the limit surfaces are smooth.

We will use the problem setting as introduced in Chapter 4: A mesh with a single n-sided hole as its only boundary. The mesh around the hole is assumed to consist of quadrilaterals only. We will represent the hole as *either* an n-sided face (for the Doo-Sabin scheme) *or* a vertex with valency n (for the Catmull-Clark scheme).

Note that in this chapter, we will have to use *double indices* for the matrices. The upper indices will refer to rows and columns of the matrix, whereas the lower indices will distinguish the matrices.

5.1 Preliminary definitions

We recall the parametrization of the prolongations, as described by Equation 4.2: For both the Doo-Sabin and the Catmull-Clark scheme, we have a block vector of control points, namely $B := (B^0, B^1, \dots, B^{n-1})^T$. Each block B^k is a vector of control points, i.e.

$$B^k := \begin{pmatrix} B^{k,1} \\ B^{k,2} \\ \dots \\ B^{k,j} \end{pmatrix},$$

where j is the number of control points required for expressing one patch of the prolongations and $k \in \{0, 1, \dots, n-1\}$. When considering subdivision schemes that generalize B-spline surface subdivision of degree (p, p), for example, usually $j = (p+1)^2$. The Catmull-Clark scheme, however, will turn out to be an exception to this rule: In order to force block-circulance for the matrix, each B^k needs to consist of 13 control points—we will clarify this later.

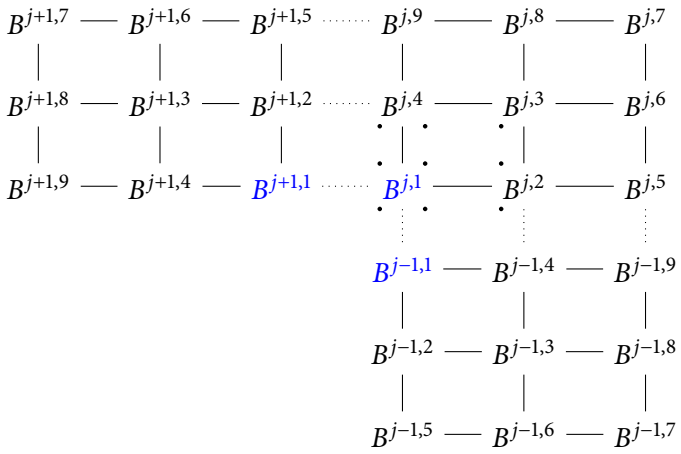


Figure 5.1: *The* local *part of the mesh describing an n-sided hole for the Doo-Sabin subdivision algorithm. Dotted lines between adjacent patches indicate faces that belong to two adjacent patches. The highlighted control points define a part of the n-sided hole, which constitutes the* only *extraordinary vertex in the mesh. Small circles indicate the* new control points *for the refined mesh that correspond to the jth patch.*

5.2 Doo-Sabin subdivision scheme

The Doo-Sabin scheme as introduced in Section 3.4.1 is an extension of the *biquadratic* B-spline subdivision scheme. It does not distinguish between ordinary and extraordinary vertices but between ordinary and extraordinary *faces*. Consequently, we represent the hole as an n-sided face. Following the remarks of Section 5.1, we describe the inner layer of the hole by n blocks of control points that contain 9 elements each. A *local* part of this mesh is shown in Figure 5.1.

Taking a look at the stencils of the Doo-Sabin scheme as depicted by Figure 3.8, we see that the Doo-Sabin algorithm is a symmetric subdivision algorithm in the sense of Definition 4.19 for all *regular* parts of the mesh—there is only one stencil for calculating new control points and neither shifts nor reflections of the control points change the results of this stencil. Thus, as long as the weights α_j for the vertices of an extraordinary face are nonnegative, satisfying $\sum_{i=0}^{n-1} \alpha_i = 1$ (to ensure affine invariance) and $a_j = a_{n-j}$ for $j \in \mathbb{Z}_n$, the algorithm will also be symmetric for *irregular* parts of the mesh.

5.2.1 Calculating the subdivision matrix

Under the assumptions from above, we now apply the Doo-Sabin subdivision algorithm to the mesh. All faces in the mesh, except for the n-sided hole, are *regular*. Therefore, the new control points corresponding to the refined jth patch can be determined from patches $j-1$, j, $j+1$ only. From the remaining patches, only the control point corresponding to one of the vertices of the n-sided hole is used. Applying the Doo-Sabin stencils results in the following matrices (recall the notation of ranges for rows and columns for matrices, as introduced in Chapter 1):

$$A_0^{1:9,1:4} = \begin{pmatrix} \alpha_0 & 0 & 0 & 0 \\ \frac{9}{16} & \frac{3}{16} & 0 & 0 \\ \frac{9}{16} & \frac{3}{16} & \frac{1}{16} & \frac{3}{16} \\ \frac{9}{16} & 0 & 0 & \frac{3}{16} \\ \frac{3}{16} & \frac{9}{16} & 0 & 0 \\ \frac{3}{16} & \frac{9}{16} & \frac{3}{16} & \frac{1}{16} \\ \frac{1}{16} & \frac{3}{16} & \frac{9}{16} & \frac{3}{16} \\ \frac{3}{16} & \frac{1}{16} & \frac{3}{16} & \frac{9}{16} \\ \frac{3}{16} & 0 & 0 & \frac{9}{16} \end{pmatrix}, \quad A_1^{1:9,1:4} = \begin{pmatrix} \alpha_1 & 0 & 0 & 0 \\ \frac{3}{16} & 0 & 0 & \frac{1}{16} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \frac{1}{16} & 0 & 0 & \frac{3}{16} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \quad A_{n-1}^{1:9,1:2} = \begin{pmatrix} \alpha_{n-1} & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{3}{16} & \frac{1}{16} \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ 0 & 0 \\ \frac{1}{16} & \frac{3}{16} \end{pmatrix} \tag{5.1}$$

The remaining matrices A_2, A_3, …, A_{n-2} only contain 1 entry that is nonzero, namely $A_j^{1,1} = \alpha_j$, for $j \in \{2, 3, \ldots, n-2\}$. We now apply the DFT to this system of matrices. By Equation 4.27, the kth transformed matrix is calculated as

$$\widehat{A}_k = \begin{pmatrix} \widehat{\alpha}_k & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ \frac{9}{16} + \frac{3}{16}\overline{w}_k & \frac{3}{16} & 0 & \frac{1}{16}\overline{w}_k & 0 & 0 & 0 & 0 & 0 \\ \frac{9}{16} & \frac{3}{16} & \frac{1}{16} & \frac{3}{16} & 0 & 0 & 0 & 0 & 0 \\ \frac{9}{16} + \frac{3}{16}w_k & \frac{1}{16}w_k & 0 & \frac{3}{16}w & 0 & 0 & 0 & 0 & 0 \\ \frac{3}{16} + \frac{1}{16}\overline{w}_k & \frac{9}{16} & 0 & \frac{3}{16}\overline{w}_k & 0 & 0 & 0 & 0 & 0 \\ \frac{3}{16} & \frac{9}{16} & \frac{3}{16} & \frac{1}{16} & 0 & 0 & 0 & 0 & 0 \\ \frac{1}{16} & \frac{3}{16} & \frac{9}{16} & \frac{3}{16} & 0 & 0 & 0 & 0 & 0 \\ \frac{3}{16} & \frac{1}{16} & \frac{3}{16} & \frac{9}{16} & 0 & 0 & 0 & 0 & 0 \\ \frac{3}{16} + \frac{1}{16}w_k & \frac{3}{16}w_k & 0 & \frac{9}{16} & 0 & 0 & 0 & 0 & 0 \end{pmatrix}, \tag{5.2}$$

where $w_k := \exp(2\pi i k/n)$ and $\overline{w}_k$ designates the complex conjugate of w_k. The *transformed weights* $\widehat{\alpha}_k$ are then given by

$$\widehat{\alpha}_k := \sum_{j=0}^{n-1} \alpha_j w_n^{-jk}. \tag{5.3}$$

Lemma 5.1. *If $k \neq 0$ and at least two weights of the Doo-Sabin scheme are nonzero, the transformed weight $\widehat{\alpha}_k$ has complex modulus less than one.*

Proof. We have $\sum_{j=0}^{n-1} \alpha_j = 1$ and $\alpha_j \geq 0$. Hence, Equation 5.3 is a *convex combination* of roots of unity. Since the convex hull of roots of unity is an n-sided polygon, $\widehat{\alpha}_k$ will be an *inner point*, which has modulus less than 1—the case that $\widehat{\alpha}_k$ is a *vertex* of the convex hull cannot occur because this would imply that there is a weight $\alpha_j = 1$ and the remaining weights are zero. ∎

5.2.2 Eigenvalues and convergence

In this section, we will perform spectral analysis of the subdivision matrix from Equation 5.2. The transformed weights will play a decisive role in determining the smoothness properties of the algorithm.

Convergence

Due to the similarity of the subdivision matrix A and the transformed matrix $\widehat{A}$, we can determine all eigenvalues of A by calculating the eigenvalues of each $\widehat{A}_k$, which turn out to be $\widehat{\alpha}_k$, $1/4$, $1/8$, $1/16$, and 0. Knowing the eigenvalues, we can prove that the Doo-Sabin algorithm converges for *almost all* symmetric weights that sum to 1. This is our first *nontrivial* result based on the machinery introduced in the previous chapter.

Proposition 5.1 (Convergence of the Doo-Sabin scheme)**.** *The Doo-Sabin subdivision algorithm is a* convergent *subdivision scheme according to Definition 4.4 as long as there are at least two nonzero weights.*

Proof. By Equation 5.3, we have $\widehat{\alpha}_0 = \sum_{j=0}^{n-1} \alpha_j = 1$, hence 1 is always an eigenvalue of the subdivision matrix. All other eigenvalues have modulus less than one either by Lemma 5.1 (for the transformed weights) or by the preceding paragraph. Consequently, convergence follows from Theorem 4.1. ∎

Necessary condition for eigenvalues

Returning to the general weights α_j, we can find a necessary condition for the eigenvalues. By Theorem 4.6, the subdominant eigenvalue must be

$$\lambda := \widehat{\alpha}_1 = \widehat{\alpha}_{n-1} \in \left(\frac{1}{4}, 1\right).$$

Otherwise, the characteristic map would not be injective. Using Theorem 4.4, this would imply that the algorithm does not produce *regular* limit surfaces. Thus, we obtain a constraint for the range of permissible weights:

$$1 > \widehat{\alpha}_1 > \max\left\{\frac{1}{4}, \left|\widehat{\alpha}_2\right|, \ldots, \left|\widehat{\alpha}_{n/2}\right|\right\} \tag{5.4}$$

5.2.3 Original weights

We now examine the smoothness properties for the *original weights* of Doo and Sabin [DS78], i.e.

$$\alpha_j = \frac{\delta_{0,j}}{4} + \frac{3 + 2\cos(2\pi j/n)}{4n}, \tag{5.5}$$

where n is the number of vertices of the face and $\delta_{0,j}$ denotes the Kronecker delta. As a first step, we show that the weights satisfy Equation 5.4. For this purpose, we need to calculate $\widehat{\alpha}_k$. Inserting Equation 5.5 into Equation 5.3 yields

$$\begin{aligned}\widehat{\alpha}_k &= \sum_{j=0}^{n-1} \alpha_j w_n^{-jk} \\ &= \sum_{j=0}^{n-1} \left(\frac{\delta_{0,j}}{4} + \frac{3}{4n} + \frac{2\cos(2\pi j/n)}{4n}\right) w_n^{-jk}.\end{aligned}$$

We split the previous equation into three sums and treat them independently. For the first sum, we have

$$\sum_{j=0}^{n-1} \frac{\delta_{0,j}}{4} w_n^{-jk} = \frac{1}{4}.$$

Roots of unity sum to zero, thus the second sum evaluates to

$$\sum_{j=0}^{n-1} \frac{3}{4n} w_n^{-jk} = \frac{3}{4n} \sum_{j=0}^{n-1} w_n^{jk} = 0.$$

Hence, $\widehat{\alpha}_1$ is given by

$$\widehat{\alpha}_k = \frac{1}{4} + \frac{1}{2n} \sum_{j=0}^{n-1} \cos(2\pi j/n) w_n^{-jk}. \tag{5.6}$$

In order to evaluate the sum of cosines, we shall require a lemma.

Lemma 5.2. *The sum of cosines from Equation 5.6 is nonzero for $k = 1$ and $k = n-1$ only. More precisely, we have*

$$\sum_{j=0}^{n-1} \cos(2\pi j/n) w_n^{-jk} = \begin{cases} n/2 & \text{for } k = 1 \text{ and } k = n-1 \\ 0 & \text{else} \end{cases}. \tag{5.7}$$

Proof. By Euler's formula and the symmetry properties of the cosine and sine functions, we have

$$\begin{aligned} w_n^{-jk} &= \cos(-2\pi jk/n) + i \sin(-2\pi jk/n) \\ &= \cos(2\pi jk/n) - i \sin(2\pi jk/n) \\ &= \overline{w}_n^{jk}. \end{aligned} \tag{5.8}$$

We first consider the real part of Equation 5.7. By using Equation 5.8, we obtain:

$$\begin{aligned} \sum_{j=0}^{n-1} \cos(2\pi j/n) w_n^{-jk} = \frac{1}{2} \sum_{j=0}^{n-1} \Big(& \cos(2\pi(k+1)j/n) \\ & + \cos(2\pi(k-1)j/n) \Big) \end{aligned} \tag{5.9}$$

If $k \neq (n-1)$, we can apply the summation formula for cosines with arguments in arithmetic progression (see Euler [EB88], pp. 225–226, for a proof) to the first sum of cosines from Equation 5.9. This results in

$$\begin{aligned} \sum_{j=0}^{n-1} \cos(2\pi(k+1)j/n) &= \frac{\sin(2\pi(k+1)) \cos(\pi(k+1)(n-1)/n)}{\sin(\pi(k+1)/n)} \\ &= 0. \end{aligned}$$

If $k = (n-1)$, however, the first sum from Equation 5.9 evaluates to $n/2$ because only multiples of $\cos(2\pi)$ are summed. An analogous consideration can be done for the second sum of cosines. Hence the second sum also evaluates to 0 (for $k \neq 1$) or to $n/2$ (for $k = 1$).

Finally, application of the summation formula for sines with arguments in arithmetic progression (see Euler [EB88], pp. 224–225, for a proof) shows that the imaginary part of Equation 5.7 is always zero. ∎

We now return to the calculation of $\widehat{\alpha}_k$ and apply Lemma 5.2 to Equation 5.6. For $\widehat{\alpha}_k$, we obtain:

$$\begin{aligned}\widehat{\alpha}_k &= \frac{1}{4} + \frac{1}{2n}\sum_{j=0}^{n-1}\cos(2\pi j/n)w_n^{-jk} \\ &= \frac{1}{4} + \frac{1}{2n}\cdot\begin{cases} n/2 & \text{for } k = 1 \text{ and } k = n-1 \\ 0 & \text{else} \end{cases} \\ &= \begin{cases} 1/2 & \text{for } k = 1 \text{ and } k = n-1 \\ 1/4 & \text{else} \end{cases}\end{aligned}$$

Consequently, $\widehat{\alpha}_k$ satisfies Equation 5.4. In particular, we have $\widehat{\alpha}_1 = \widehat{\alpha}_{n-1} = 1/2 > 1/4$, which implies that $\lambda := \widehat{\alpha}_1 = \widehat{\alpha}_{n-1}$ is the subdominant eigenvalue. Moreover, it turns out that the subdivision algorithm indeed creates regular surfaces.

Theorem 5.1 (Smoothness of the original Doo-Sabin subdivision scheme)**.** *If the weights are given by*

$$\alpha_j = \frac{\delta_{0,j}}{4} + \frac{3 + 2\cos(2\pi j/n)}{4n},$$

then the limit surface of the Doo-Sabin subdivision scheme will be a C^1-manifold for almost all input data.

Proof. We first calculate the eigenvector ψ_* of the transformed matrix $\widehat{A}_1$ corresponding to the subdominant eigenvalue $\lambda := \widehat{\alpha}_1 = \widehat{\alpha}_{n-1} = 1/2$. Arranging the components of ψ_* in a matrix according to the labelling shown in Figure 5.1 (the labels have been "rotated" because of the parametrization of the characteristic map, which requires ψ_3, ψ_6, ψ_7, and ψ_8 to determine the image of the boundary according to Figure 4.5), we obtain

$$\begin{aligned}\psi_* &= \begin{pmatrix} \psi_1 & \psi_4 & \psi_9 \\ \psi_2 & \psi_3 & \psi_8 \\ \psi_5 & \psi_6 & \psi_7 \end{pmatrix} \\ &= \begin{pmatrix} 7 & 14 + 7w_n & 21 + 14w_n \\ 14 + 7\overline{w}_n & 21 + 6c_n & 28 + 9w_n + 2\overline{w}_n \\ 21 + 14\overline{w}_n & 28 + 9\overline{w}_n + 2w_n & 35 + 12c_n \end{pmatrix},\end{aligned} \tag{5.10}$$

where $c_n := \cos(2\pi/n)$ and $s_n := \sin(2\pi/n)$.

We have already seen that all segments of the characteristic map are related by rotations in the complex plane. More precisely, using Equation 4.40 we can compute *all* control points by multiplying ψ_* with powers of w_n. Hence, we consider the first segment of the characteristic map only. The first segment can be obtained by calculating control points for patches 0, 1, and $n-1$, which are then connected suitably: In order to connect segments 0 and 1, for example, we need to use one column of control points from patch 1 and two columns of control points from patch 0.

Connecting the segments yields three biquadratic B-spline patches that define Ψ^0, the first segment of the characteristic map. The real and imaginary parts of the control points are given by:

$$\begin{array}{ccccccc}
 & & & & \begin{matrix}7c_n\\7s_n\end{matrix} & \begin{matrix}7+14c_n\\14s_n\end{matrix} & \begin{matrix}14+21c_n\\21s_n\end{matrix}\\
 & & & & \begin{matrix}7\\0\end{matrix} & \begin{matrix}14+7c_n\\7s_n\end{matrix} & \begin{matrix}21+14c_n\\14s_n\end{matrix}\\
 & & & & \begin{matrix}14+7c_n\\-7s_n\end{matrix} & \begin{matrix}21+6c_n\\0\end{matrix} & \begin{matrix}28+11c_n\\7s_n\end{matrix}\\
\begin{matrix}7c_n\\-7s_n\end{matrix} & \begin{matrix}7\\0\end{matrix} & \begin{matrix}14+7c_n\\7s_n\end{matrix} & \begin{matrix}7\\0\end{matrix} & \begin{matrix}14+7c_n\\7s_n\end{matrix} & \begin{matrix}21+14c_n\\14s_n\end{matrix}\\
\begin{matrix}7+14c_n\\-14s_n\end{matrix} & \begin{matrix}14+7c_n\\-7s_n\end{matrix} & \begin{matrix}21+6c_n\\0\end{matrix} & \begin{matrix}14+7c_n\\-7s_n\end{matrix} & \begin{matrix}21+6c_n\\0\end{matrix} & \begin{matrix}28+11c_n\\7s_n\end{matrix}\\
\begin{matrix}14+21c_n\\-21s_n\end{matrix} & \begin{matrix}21+14c_n\\-14s_n\end{matrix} & \begin{matrix}28+11c_n\\-7s_n\end{matrix} & \begin{matrix}21+14c_n\\-14s_n\end{matrix} & \begin{matrix}28+11c_n\\-7s_n\end{matrix} & \begin{matrix}35+12c_n\\0\end{matrix}
\end{array}$$

We now convert the B-spline control points to the corresponding Bézier-spline control points according to Equation 2.11. This step is a preparation for the application of the criteria derived in the preceding chapter. Note that some of the Bézier-spline control points coincide, thus we can represent the three patches more compactly. Converting the control points and scaling them by 1/2 yields:

$$\begin{array}{ccccccc}
 & & & \begin{matrix}14+14c_n\\14s_n\end{matrix} & \begin{matrix}21+21c_n\\21s_n\end{matrix} & \begin{matrix}28+28c_n\\28s_n\end{matrix} & \\
 & & & \begin{matrix}21+7cn\\7s_n\end{matrix} & \begin{matrix}28+14c_n\\14s_n\end{matrix} & \begin{matrix}35+21c_n\\21s_n\end{matrix} & \\
\begin{matrix}14+14c_n\\-14s_n\end{matrix} & \begin{matrix}21+7c_n\\-7s_n\end{matrix} & \begin{matrix}28+10c_n\\0\end{matrix} & \begin{matrix}35+13c_n\\7s_n\end{matrix} & \begin{matrix}42+19c_n\\14s_n\end{matrix} & * \frac{1}{2}\\
\begin{matrix}21+21c_n\\-21s_n\end{matrix} & \begin{matrix}28+14c_n\\-14s_n\end{matrix} & \begin{matrix}35+13c_n\\-7s_n\end{matrix} & \begin{matrix}42+12c_n\\0\end{matrix} & \begin{matrix}49+17c_n\\14s_n\end{matrix} & \\
\begin{matrix}28+28c_n\\-28s_n\end{matrix} & \begin{matrix}35+21c_n\\-21s_n\end{matrix} & \begin{matrix}42+19c_n\\-14s_n\end{matrix} & \begin{matrix}49+17c_n\\-7s_n\end{matrix} & \begin{matrix}56+20c_n\\0\end{matrix} &
\end{array}$$

We use Proposition 2.9 to calculate the partial derivatives in v-direction and obtain three quadratic-linear patches with the following coefficients:

$$\begin{array}{ccccc}
 & & \begin{matrix}7+7c_n\\7s_n\end{matrix} & \begin{matrix}7+7c_n\\7s_n\end{matrix} & \\
 & & \begin{matrix}7+7c_n\\7s_n\end{matrix} & \begin{matrix}7+7c_n\\7s_n\end{matrix} & \\
\begin{matrix}7-7c_n\\7s_n\end{matrix} & \begin{matrix}7+3c_n\\7s_n\end{matrix} & \begin{matrix}7+6c_n\\7s_n\end{matrix} & * \frac{1}{2}\\
\begin{matrix}7-7c_n\\7s_n\end{matrix} & \begin{matrix}7-c_n\\7s_n\end{matrix} & \begin{matrix}7+5c_n\\7s_n\end{matrix} & \\
\begin{matrix}7-7c_n\\7s_n\end{matrix} & \begin{matrix}7-2c_n\\7s_n\end{matrix} & \begin{matrix}7+3c_n\\7s_n\end{matrix} &
\end{array}$$

Since $s_n > 0$ and $c_n \geq -1/2$ for $n \geq 3$, both components of the control points are *positive*. By the convex hull property for Bézier surfaces, the first segment of the characteristic map is in the convex hull of its control points. Since all control points are positive, the components $\Psi^0_{1,v}$ and $\Psi^0_{2,v}$ of the partial derivative

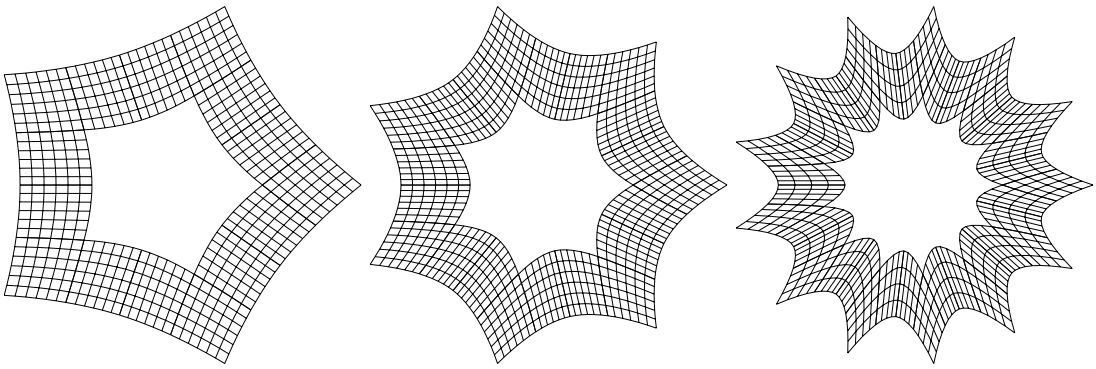


Figure 5.2: *Regular and injective characteristic map of the Doo-Sabin scheme for valencies* 5, 7, *and* 13 *(from left to right). Note the rotational symmetry of the segments.*

in v-direction are positive for all parameter values. Hence, by Corollary 4.2, the characteristic map will be regular and injective. Consequently, the limit surface of the algorithm is a C^1-manifold for almost all input data by Theorem 4.3. ■

Figure 5.2 depicts the characteristic map of the Doo-Sabin scheme with original weights for valencies 3, 6, and 12.

5.2.4 Degenerate weights

Having shown the correctness of the Doo-Sabin algorithm for its original weights, we now take a look at weights that have been intentionally chosen to generate highly irregular surfaces.

Theorem 5.2 (Degenerate version of the Doo-Sabin algorithm)**.** *Let*

$$\alpha_j := \left(1 - \delta_{0,j}\right) \frac{1}{n-1}$$

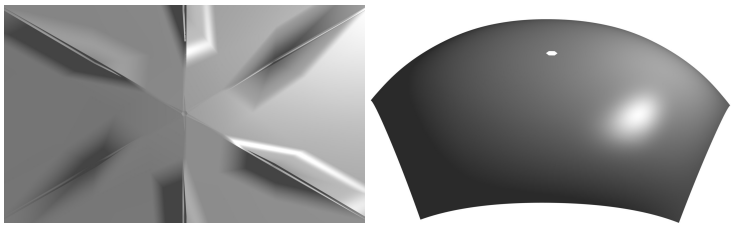
be the weights for the Doo-Sabin algorithm. Then the algorithm will not *produce regular surfaces.*

Proof. The weights are obviously symmetric and sum to 1. By Equation 5.3, the transformed weights that result from applying the DFT are given by

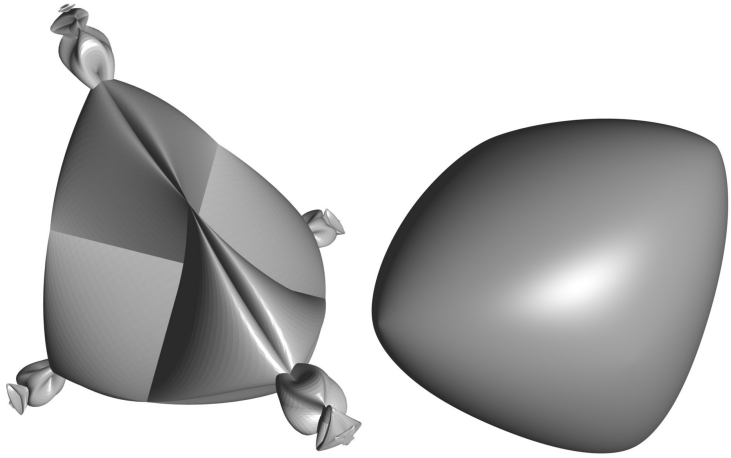
$$\begin{aligned}\widehat{\alpha}_k &= \sum_{j=0}^{n-1} \alpha_j w_n^{-jk} = \left(\sum_{j=0}^{n-1} w_n^{-jk} - w_n^0\right) \alpha_1 \\ &= -\alpha_1 = -\frac{1}{n-1}.\end{aligned}$$

Thus neither sign nor modulus nor multiplicity of the eigenvalues is correct. As a consequence of Theorem 4.6, the characteristic map will be non-injective. Hence the resulting surface will not be regular for almost every set of input data by Theorem 4.4. ■

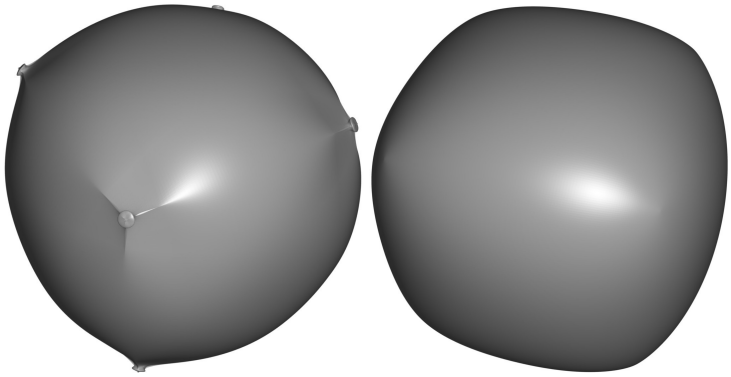
Figure 5.3 offers a visual proof of the preceding theorem. It compares the degenerate weights of this section to the original weights. Since the surfaces feature self-intersections, they *cannot* be regular.



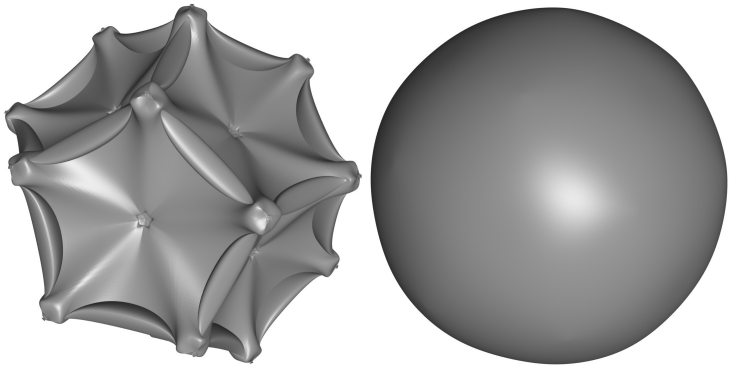
(a) 6-*sided hole*



(b) *Tetrahedron*



(c) *Hexahedron*



(d) *Icosahedron*

Figure 5.3: *Results of the Doo-Sabin algorithm using degenerate weights (left) and original weights (right) on common meshes. The same number of subdivision steps was applied for both weight schemes. For rendering the meshes, self-shadowing has been disabled because all meshes contain self-intersections. The degenerate variant of the* 6*-sided hole has been magnified in order to show the visual artefacts around the origin—they are the reason why the hole appears to be* closed.

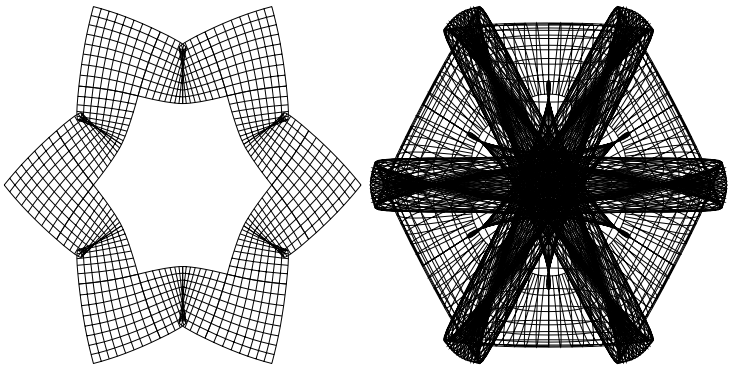


Figure 5.4: *Non-injective characteristic maps for the Doo-Sabin scheme. In both cases, $n = 6$. The darker parts indicate self-intersections. In the left example, the subdominant eigenvalue does not satisfy Theorem 4.6, i.e. its* Fourier index *is incorrect. In the right example, the subdominant eigenvalue is* negative.

5.2.5 Permissible weights

For general weights, the analysis of the Doo-Sabin algorithm is not straightforward and rather technical. We cite the following theorem of Peters and Reif [PR08]:

Theorem 5.3. *Let $\alpha_0, \alpha_1, \ldots, \alpha_{n-1}$ be symmetric and affine invariant weights for the Doo-Sabin subdivision algorithm. If and only if the subdominant eigenvalue $\lambda := \widehat{\alpha}_1 = \widehat{\alpha}_{n-1}$ satisfies*

$$1 > \lambda > \max\left\{\frac{1}{4}, \left|\widehat{\alpha}_2\right|, \ldots, \left|\widehat{\alpha}_{n/2}\right|\right\} \tag{5.11}$$

and

$$128\lambda^2(1-\lambda) - 7\lambda - 2 + 9\lambda\cos\left(\frac{2\pi}{n}\right) > 0, \tag{5.12}$$

then the limit surface will be a C^1-manifold for almost every set of initial control points.

Proof. See Peters and Reif [PR08], pp. 116–119, or [PR98], pp. 740–742. ∎

Figure 5.4 depicts two different weight sets that yield non-injective characteristic maps. Only by the previous theorem can these weight sets be distinguished from "good" weight sets.

5.3 Catmull-Clark subdivision scheme

We now consider the Catmull-Clark subdivision scheme and apply the methods from the previous chapter in order to derive conditions for regular surfaces. In Chapter 3, we have already seen that the Catmull-Clark scheme only generates quadrilaterals after the first subdivision step. Therefore we can assume that the mesh consists exclusively of quadrilaterals, and the *limit* considerations will certainly not be changed by this assumption.

Taking a look at the weighted stencil for extraordinary vertices as depicted in Figure 3.12, we see that the numbering of the control points is irrelevant for the stencil because only the topological situation is used for the calculation of refined control points. Furthermore, the stencils for the regular case, being

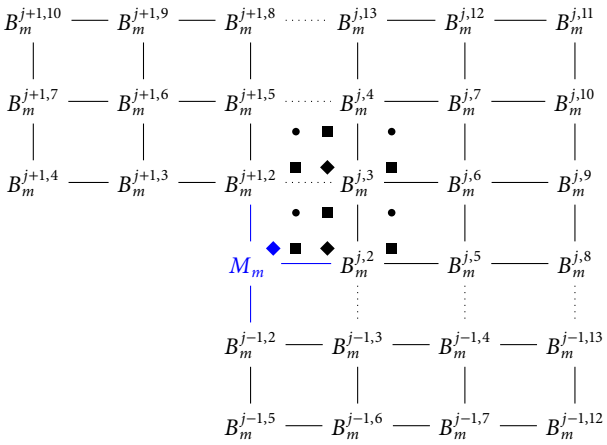


Figure 5.5: *The* local *part of a mesh based on bicubic B-spline patches. The mesh contains a single vertex of valency n and is regular everywhere else. Dotted lines between patches indicate faces that belong to two adjacent patches. The single irregular vertex and its connectivity information are highlighted in blue. When applied to this mesh, the Catmull-Clark algorithm generates new control points that replace the old control points adjacent to the irregular vertex. The new control points corresponding to the jth patch are drawn as small symbols: A circle indicates a new* face point*, a rectangle indicates a new* edge point*, and a diamond indicates a new* vertex point *(see Figure 3.10 and Figure 3.12 for their definitions). Note that patches $j-1$ and $j+1$ are not completely shown due to size constraints.*

defined by bicubic B-spline subdivision, also satisfy this condition. Thus, the Catmull-Clark scheme is a symmetric subdivision algorithm according to Definition 4.19.

As a preparation for setting up the subdivision matrix, we take a look at the neighbourhood around a vertex of valency n. Following Section 5.1, we have 16 control points per patch. Some of the control points of adjacent patches overlap; hence we index 13 control points with the current patch number and add 3 control points from the adjacent patch. Figure 5.5 shows the local part of the mesh along with the neighbourhood relations, including *type* and *position* of the refined control points. Again, the mesh is assumed to contain only one irregular vertex. Consequently, we can use the normal subdivision stencils as depicted in Figure 3.10 for all control points save the first one.

For the extraordinary vertex, a trick is required: Since this vertex is shared by *all* patches, we create n identical copies of the vertex. Hence we define $M_m := B_m^{0,1} = \cdots = B_m^{n-1,1}$, which we may also write as

$$M_m = \frac{1}{n} \sum_{j=0}^{n-1} B_m^j. \tag{5.13}$$

The identity described by this equation yields a block-circulant structure of the matrix (see Peters and Reif [PR08], pp. 97–98, for more details). As a consequence, if a weight w is applied to $B_m^{j,1}$ during the subdivision algorithm, we write down the value w/n in all those columns of the matrices $A_{j,1}$ that affect $B_m^{j,1}$.

5.3.1 Calculating the subdivision matrix

We now calculate the subdivision matrix of the Catmull-Clark scheme. For this purpose, we need to apply the stencils to the mesh and analyse the new neighbourhood relations. Applying the subdivision

stencils according to Figure 5.5 gives us the following set of matrices, namely,

$$A_0^{1:13,1:7} = \begin{pmatrix} \frac{\alpha}{n} & \frac{\beta}{n} & \frac{\gamma}{n} & 0 & 0 & 0 & 0 \\ \frac{3}{8n} & \frac{3}{8} & \frac{1}{16} & 0 & 0 & 0 & 0 \\ \frac{1}{4n} & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 & 0 \\ \frac{1}{16n} & \frac{1}{16} & \frac{3}{8} & \frac{1}{16} & 0 & 0 & 0 \\ \frac{3}{32n} & \frac{9}{16} & \frac{3}{32} & 0 & \frac{3}{32} & \frac{1}{64} & 0 \\ \frac{1}{16n} & \frac{3}{8} & \frac{3}{8} & 0 & \frac{1}{16} & 0 & 0 \\ \frac{1}{64n} & \frac{3}{32} & \frac{9}{16} & \frac{3}{32} & \frac{1}{64} & \frac{3}{32} & \frac{1}{64} \\ 0 & \frac{3}{8} & \frac{1}{16} & 0 & \frac{3}{8} & \frac{1}{16} & 0 \\ 0 & \frac{1}{4} & \frac{1}{4} & 0 & \frac{1}{4} & \frac{1}{4} & 0 \\ 0 & \frac{1}{16} & \frac{3}{8} & \frac{1}{16} & \frac{1}{16} & \frac{3}{8} & \frac{1}{16} \\ 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & \frac{1}{4} & \frac{1}{4} \\ 0 & 0 & \frac{3}{8} & \frac{3}{8} & 0 & \frac{1}{16} & \frac{1}{16} \\ 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & 0 & 0 \end{pmatrix}, \tag{5.14}$$

which contains the weights for the *central* patch, and

$$A_1^{1:13,1:5} = \begin{pmatrix} \frac{\alpha}{n} & \frac{\beta}{n} & \frac{\gamma}{n} & 0 & 0 \\ \frac{3}{8n} & \frac{1}{16} & 0 & 0 & 0 \\ \frac{1}{4n} & \frac{1}{4} & 0 & 0 & 0 \\ \frac{1}{16n} & \frac{3}{8} & 0 & 0 & \frac{1}{16} \\ \frac{3}{32n} & \frac{1}{64} & 0 & 0 & 0 \\ \frac{1}{16n} & \frac{1}{16} & 0 & 0 & 0 \\ \frac{1}{64n} & \frac{3}{32} & 0 & 0 & \frac{1}{64} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{16} & 0 & 0 & \frac{1}{16} \\ 0 & \frac{1}{4} & 0 & 0 & \frac{1}{4} \end{pmatrix}, \quad A_{n-1}^{1:8,1:4} = \begin{pmatrix} \frac{\alpha}{n} & \frac{\beta}{n} & \frac{\gamma}{n} & 0 \\ \frac{3}{8n} & \frac{1}{16} & \frac{1}{16} & 0 \\ \frac{1}{4n} & 0 & 0 & 0 \\ \frac{1}{16n} & 0 & 0 & 0 \\ \frac{3}{32n} & \frac{1}{64} & \frac{3}{32} & \frac{1}{64} \\ \frac{1}{16n} & 0 & 0 & 0 \\ \frac{1}{64n} & 0 & 0 & 0 \\ 0 & 0 & \frac{1}{16} & \frac{1}{16} \end{pmatrix}, \tag{5.15}$$

where A_1 contains the weights for the *right* patch and A_{n-1} contains the weights for the *left* patch. By Figure 5.5, no control points from other patches are required for calculating the positions of the new control points. Thus, as a consequence of Equation 5.13, the remaining matrices only contain the weights

for the central control point. Hence, we have

$$A_j^{1:8,1:4} = \begin{pmatrix} \frac{\alpha}{n} & \frac{\beta}{n} & \frac{\gamma}{n} & 0 \\ \frac{3}{8n} & 0 & 0 & 0 \\ \frac{1}{4n} & 0 & 0 & 0 \\ \frac{1}{16n} & 0 & 0 & 0 \\ \frac{3}{32n} & 0 & 0 & 0 \\ \frac{1}{16n} & 0 & 0 & 0 \\ \frac{1}{64n} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}, \tag{5.16}$$

with $j = 2, \ldots, n-2$.

We now apply the DFT to this set of matrices. Let w_k be the kth root of unity, i.e. $w_k := \exp(2\pi ik/n) = c_{n,k} + is_{n,k}$. For $k = 1$, we will write c_n and s_n instead of $c_{n,1}$ and $s_{n,1}$.

Since the sum of all roots of unity is zero, only the transformed matrix $\widehat{A}_0 := \sum_{j=0}^{n-1} A_j$ will contain nonzero entries in the first column and the first row. We denote this by the usual Kronecker delta notation and obtain a closed expression for the kth transformed matrix, namely,

$$\widehat{A}_k = \begin{pmatrix} \widehat{A}_k^{0,0} & 0 & 0 \\ \widehat{A}_k^{1,0} & \widehat{A}_k^{1,1} & 0 \\ \widehat{A}_k^{2,0} & \widehat{A}_k^{2,1} & 0 \end{pmatrix} = \begin{pmatrix} \widehat{A}_k^{1:13,1:7} & 0 \end{pmatrix}, \tag{5.17}$$

where the block matrices are given by

$$\widehat{A}_k^{1:13,1:7} = \left(\begin{array}{cccc|cccc} \alpha\delta_{k,0} & \beta\delta_{k,0} & \gamma\delta_{k,0} & 0 & 0 & 0 & 0 \\ \frac{3}{8}\delta_{k,0} & \frac{3}{8} + \frac{1}{16}\cdot 2c_{n,k} & \frac{1}{16}(1+\overline{w}_k) & 0 & 0 & 0 & 0 \\ \frac{1}{4}\delta_{j,0} & \frac{1}{4}(1+w_k) & \frac{1}{4} & 0 & 0 & 0 & 0 \\ \hline \frac{1}{16}\delta_{k,0} & \frac{1}{16} + \frac{3}{8}w_k & \frac{3}{8} & \frac{1}{16} & \frac{1}{16}w_k & 0 & 0 \\ \frac{3}{32}\delta_{k,0} & \frac{9}{16} + \frac{1}{64}\cdot 2c_{n,k} & \frac{3}{32}(1+\overline{w}_k) & \frac{1}{64}\overline{w}_k & \frac{3}{32} & \frac{1}{64} & 0 \\ \frac{1}{16}\delta_{k,0} & \frac{3}{8} + \frac{1}{16}w_k & \frac{3}{8} & 0 & \frac{1}{16} & \frac{1}{16} & 0 \\ \frac{1}{64}\delta_{k,0} & \frac{3}{32}(1+w_k) & \frac{9}{16} & \frac{3}{32} & \frac{1}{64}(1+w_k) & \frac{3}{32} & \frac{1}{64} \\ \hline 0 & \frac{3}{8} & \frac{1}{16}(1+\overline{w}_k) & \frac{1}{16}\overline{w}_k & \frac{3}{8} & \frac{1}{16} & 0 \\ 0 & \frac{1}{4} & \frac{1}{4} & 0 & \frac{1}{4} & \frac{1}{4} & 0 \\ 0 & \frac{1}{16} & \frac{3}{8} & \frac{1}{16} & \frac{1}{16} & \frac{3}{8} & \frac{1}{16} \\ 0 & 0 & \frac{1}{4} & \frac{1}{4} & 0 & \frac{1}{4} & \frac{1}{4} \\ 0 & \frac{1}{16}w_k & \frac{3}{8} & \frac{3}{8} & \frac{1}{16}w_k & \frac{1}{16} & \frac{1}{16} \\ 0 & \frac{1}{4}w_k & \frac{1}{4} & \frac{1}{4} & \frac{1}{4}w_k & 0 & 0 \end{array}\right). \tag{5.18}$$

5.3.2 Eigenvalues of the subdivision matrix

The partition of the matrix as described by Equation 5.17 facilitates the calculation of eigenvalues of $\widehat{A}_k$: Let $\mathbb{1}$ denote the identity matrix. By definition, we need to solve $\det\left(\widehat{A}_k - \lambda \cdot \mathbb{1}\right) = 0$, and in a slight

abuse of notation, we will not care about the dimension of the identity matrices. This yields

$$\det\left(\widehat{A}_k - \lambda\mathbb{1}\right) = \det\begin{pmatrix} \widehat{A}_k^{0,0} - \lambda\cdot\mathbb{1} & 0 & 0 \\ \widehat{A}_k^{1,0} & \widehat{A}_k^{1,1} - \lambda\cdot\mathbb{1} & 0 \\ \widehat{A}_k^{2,0} & \widehat{A}_k^{2,1} & -\lambda\cdot\mathbb{1} \end{pmatrix}$$
$$= \det\left(\widehat{A}_k^{0,0} - \lambda\cdot\mathbb{1}\right)\det\left(\widehat{A}_k^{1,1} - \lambda\cdot\mathbb{1}\right)\det\left(-\lambda\cdot\mathbb{1}\right).$$

Therefore, all nonzero eigenvalues of $\widehat{A}_k$ are either eigenvalues of $\widehat{A}_k^{0,0}$ or of $\widehat{A}_k^{1,1}$. From $\widehat{A}_k^{1,1}$, we get the "usual" eigenvalues 1/8, 1/16, 1/32, and 1/64. Their multiplicities are too high, so we do not consider them any further.

For $k = 0$, however, we obtain the eigenvalue $\lambda_0 = 1$ from $\widehat{A}_0^{0,0}$. Using that $\gamma := 1 - \alpha - \beta$ yields a pair of eigenvalues,

$$\lambda_{1,2}^0 := \frac{1}{8}\left(4\alpha - 1 \pm \sqrt{(4\alpha-1)^2 + 8\beta - 4}\right), \tag{5.19}$$

where λ_1^0 refers to the expression with the plus sign and λ_2^0 refers to the expression with the minus sign. Depending on the sign of the discriminant, $\lambda_{1,2}^0$ are either both real or complex conjugates.

For $k \neq 0$, the transformed matrix $\widehat{A}_k^{0,0}$ has two nonzero eigenvalues, namely

$$\lambda_{1,2}^k := \frac{1}{16}\left(c_{n,k} + 5 \pm \sqrt{(c_{n,k}+9)(c_{n,k}+1)}\right), \tag{5.20}$$

which are real for every k. By Theorem 4.6, the subdominant eigenvalue must come from $\widehat{A}_1$ and $\widehat{A}_{n-1}$. Since the multiplicities of the other eigenvalues are too high, the only candidate for the subdominant eigenvalue is

$$\lambda := \lambda_1^1 = \lambda_1^{n-1} = \frac{1}{16}\left(c_n + 5 + \sqrt{(c_n+9)(c_n+1)}\right). \tag{5.21}$$

It can be shown (see Chapter A, Section A.2) that

$$1 > \lambda \geq \frac{1}{4} > \lambda_2^k > \frac{1}{16} \qquad \text{for } k = 1, \ldots, n-1 \tag{5.22}$$

and

$$1 > \lambda > \lambda_1^k \geq \frac{1}{4} \qquad \text{for } k = 2, \ldots, n-2. \tag{5.23}$$

Hence, λ is always larger than the eigenvalues of all other block matrices except $\widehat{A}_0$. As a consequence, λ is the *subdominant eigenvalue* if and only if α, β, and γ are chosen such that

$$\lambda > \max\left(\left|\lambda_1^0\right|, \left|\lambda_2^0\right|\right). \tag{5.24}$$

Having obtained a condition for λ, we are now ready to calculate the characteristic map. In the following sections, we will use the condition of this section and several auxiliary results concerning regularity and injectivity of the characteristic map in order to derive a theorem about the smoothness of the Catmull-Clark subdivision scheme.

5.3.3 Calculating the characteristic map

Henceforth, we assume that λ is the subdominant eigenvalue. Since λ is independent of the weights α, β, and γ, we may compute the characteristic map in all generality. To this end, we need to find an eigenvector $\widehat{\psi}$ of the transformed subdivision matrix $\widehat{A}_1$. For this purpose, we use the structure of the matrix as in Equation 5.17 and partition the eigenvector $\widehat{\psi}$ into three blocks such that $\widehat{\psi} = (\widehat{\psi}_0, \widehat{\psi}_1, \widehat{\psi}_2)^T$. The eigenvector condition $\widehat{A}_1\widehat{\psi} = \lambda\widehat{\psi}$ then yields a system of equations:

$$\begin{aligned}\left(\widehat{A}_1^{0,0} - \lambda\mathbb{1}\right)\widehat{\psi}_0 &= 0\\ \left(\widehat{A}_1^{1,1} - \lambda\mathbb{1}\right)\widehat{\psi}_1 &= -\widehat{A}_1^{1,0}\widehat{\psi}_0\\ \widehat{\psi}_2 &= \left(\widehat{A}_1^{2,0}\widehat{\psi}_0 + \widehat{A}_1^{2,1}\widehat{\psi}_1\right)/\lambda\end{aligned}$$

The first equation is solved by

$$\widehat{\psi}_0 := \left(0, 1 + \overline{w}_n, 16\lambda - 2c_n - 6\right)^T. \tag{5.25}$$

Using this vector, we now solve the remaining eigenvector equations and get an eigenvector $\widetilde{\psi}$. Due to the occurrences of c_n and λ, the eigenvector $\widetilde{\psi}$ is rather unwieldy. Hence, we simplify it by substitution and scaling in the subsequent paragraphs.

Expressing c_n in terms of λ

By transforming the definition of λ from Equation 5.21, we obtain

$$c_n = \frac{16\lambda^2 - 10\lambda + 1}{2\lambda} \tag{5.26}$$

for $\lambda \in \Lambda := \left[\left(9 + \sqrt{17}\right)/32, \left(3 + \sqrt{5}\right)/8\right)$, where Λ is the range of values for λ (the range can be obtained by setting $c_n = -1/2$ and $c_n = +1$, which are the maximum values of c_n for $n \geq 3$; since $c_n = +1$ is never attained, the interval is half-open). Thus, we can substitute Equation 5.26 for c_n within the components of $\widetilde{\psi}$. As a result, $\widetilde{\psi}$ is written exclusively in terms of λ.

This transformation will allow us to derive a characteristic map that depends solely on λ, which greatly simplifies calculations because the expressions do *not* contain the valency n anymore.

Scaling $\widetilde{\psi}$

Factorizing the eigenvalue $\widetilde{\psi}$ still results in fractional expressions. As a consequence, in order to achieve a manageable representation, we scale $\widetilde{\psi}$ by

$$\widehat{\psi} := \widetilde{\psi} \cdot 4\lambda \cdot \left(65536\lambda^5 - 7168\lambda^4 + 224\lambda^3 - 2\lambda^2\right), \tag{5.27}$$

from which we obtain the scaled eigenvector $\widehat{\psi}$. The eigenvector $\widehat{\psi}$ can be written as

$$\widehat{\psi} = (4\lambda - 1)\,\widehat{\psi}_{\mathrm{re}} + i2s_n\lambda\,(64\lambda - 1)\,\widehat{\psi}_{\mathrm{im}}, \tag{5.28}$$

where

$$\widehat{\psi}_{\mathrm{re}} = \begin{pmatrix} 0 \\ 4\lambda^2(4\lambda-1)(16\lambda-1)(32\lambda-1)(64\lambda-1) \\ 8\lambda^2(16\lambda-1)(32\lambda-1)(64\lambda-1) \\ 4\lambda^2(64\lambda-1)(928\lambda^2+228\lambda-31) \\ 8\lambda^2(4\lambda-1)(4\lambda+13)(16\lambda-1)(64\lambda-1) \\ 4\lambda^2(64\lambda-1)(928\lambda^2+228\lambda-31) \\ 80\lambda^2(1280\lambda^3+2128\lambda^2-56\lambda-13) \\ (4\lambda-1)(16\lambda-1)(64\lambda-1)(100\lambda^2+42\lambda-1) \\ 4\lambda(64\lambda-1)(640\lambda^3+688\lambda^2-82\lambda-1) \\ 20\lambda(2048\lambda^4+11040\lambda^3+812\lambda^2-165\lambda-1) \\ 40\lambda(5248\lambda^3+1568\lambda^2-133\lambda-5) \\ 20\lambda(2048\lambda^4+11040\lambda^3+812\lambda^2-165\lambda-1) \\ 4\lambda(64\lambda-1)(640\lambda^3+688\lambda^2-82\lambda-1)) \end{pmatrix} \tag{5.29}$$

and

$$\widehat{\psi}_{\mathrm{im}} = \begin{pmatrix} 0 \\ -4\lambda^2(16\lambda-1)(32\lambda-1) \\ 0 \\ 84\lambda^2(8\lambda+1) \\ -8\lambda^2(4\lambda+13)(16\lambda-1) \\ -84\lambda^2(8\lambda+1) \\ 0 \\ -(16\lambda-1)(100\lambda^2+42\lambda-1) \\ -4\lambda(160\lambda^2+132\lambda-1) \\ -20\lambda(8\lambda^2+15\lambda+1) \\ 0 \\ 20\lambda(8\lambda^2+15\lambda+1) \\ 4\lambda(160\lambda^2+132\lambda-1) \end{pmatrix}. \tag{5.30}$$

Using this representation, we now check whether the patch is *normalized*—we have already seen that a normalized characteristic map is necessary for injectivity. Hence, we calculate $\Psi^0(2,2)$, which is one *corner point* of the bicubic B-spline surface patch defined by $\widehat{\psi}$. We may obtain the corner point by evaluating the central B-spline patch of Ψ^0. For this purpose, we use the matrix representation of a B-spline patch as described by Theorem 2.3. We then solve the equation for $u = 1$ and $v = 1$. Written more compactly, this results in

$$\Psi^0(2,2) = \begin{pmatrix} 1/6 \\ 2/3 \\ 1/6 \end{pmatrix} \begin{pmatrix} \widehat{\psi}_3 & \widehat{\psi}_4 & \widehat{\psi}_{13} \\ \widehat{\psi}_6 & \widehat{\psi}_7 & \widehat{\psi}_{12} \\ \widehat{\psi}_9 & \widehat{\psi}_{10} & \widehat{\psi}_{11} \end{pmatrix} \begin{pmatrix} 1/6 \\ 2/3 \\ 1/6 \end{pmatrix}^T.$$

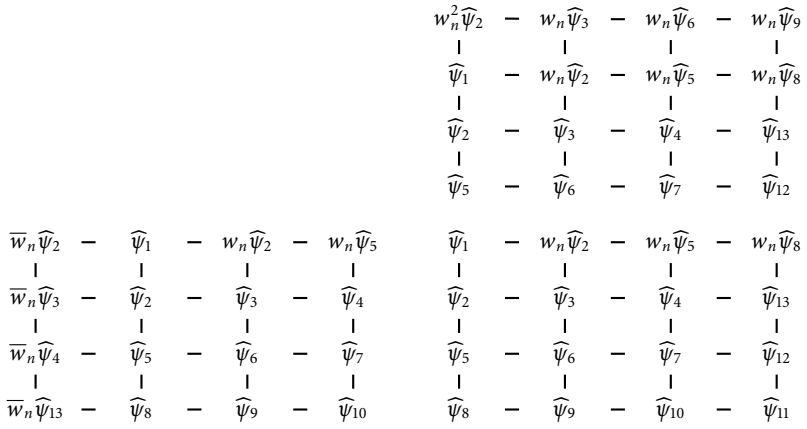


Figure 5.6: *B-spline control points for the three patches that form Ψ^0, the first segment of the characteristic map of the Catmull-Clark subdivision algorithm. The points are given in the order in which they appear in the matrices that define the corresponding B-spline patch. Note that the control points overlap. This ensures that the patches fit smoothly.*

Using Equations 5.28, 5.29, and 5.30, we obtain

$$\Psi^0(2,2) = \frac{8}{9}\lambda\,(4\lambda-1)\left(139264\lambda^4 + 170496\lambda^3 + 112\lambda^2 - 1476\lambda - 11\right).$$

The Sturm sequences (see Chapter A, Section A.3, for an introduction) of this polynomial show that it has no roots within the interval Λ. Since evaluation at $\lambda = 1/2$ yields $\Psi_*(2,2) = 13020$, the characteristic map is normalized.

In order to verify the regularity of the limit surface, we shall use Corollary 4.2. Starting from the patches as displayed in Figure 5.6, we convert the B-spline control points to Bézier-spline control points according to Section 2.5.3. By calculating the differences $P'_{i,j} = P_{i,j+1} - P_{i,j}$, we obtain the control points of Ψ^0_v, the derivative in v-direction of the first segment of the characteristic map. Since Ψ^0 is given by three bicubic patches, derivation in v-direction yields three patches with 4×3 control points each. Thus, we get 36 control points, which we enumerate by K_j with $j = 1,\dots,36$. It turns out that all K_j are polynomials in λ of the form

$$K_j = P_j(\lambda) + is_nQ_j(\lambda),$$

where P_j and Q_j are polynomials with rational coefficients whose degrees are less than or equal to 8:

$$\begin{aligned}
P_1 &= \frac{1}{9}\left(-8388608\lambda^8 + 11403264\lambda^7 - 2748416\lambda^6 - 447232\lambda^5 + 197056\lambda^4 - 16528\lambda^3 + 468\lambda^2 - 4\lambda\right)\\
Q_1 &= \frac{1}{9}\left(-1048576\lambda^7 + 901120\lambda^6 + 172544\lambda^5 - 25952\lambda^4 + 872\lambda^3 - 8\lambda^2\right)\\
&\vdots\\
P_{36} &= \frac{1}{36}\left(-4194304\lambda^7 - 3211264\lambda^6 + 6868992\lambda^5 - 1421952\lambda^4 - 54048\lambda^3 + 12120\lambda^2 - 110\lambda + 1\right)\\
Q_{36} &= \frac{1}{36}\left(524288\lambda^6 + 1073152\lambda^5 - 24064\lambda^4 - 9872\lambda^3 + 284\lambda^2 - 2\lambda\right)
\end{aligned}$$

For the sake of clarity, not all polynomials are shown. Instead, Figure 5.7 depicts both P_j and Q_j and their progression over Λ.

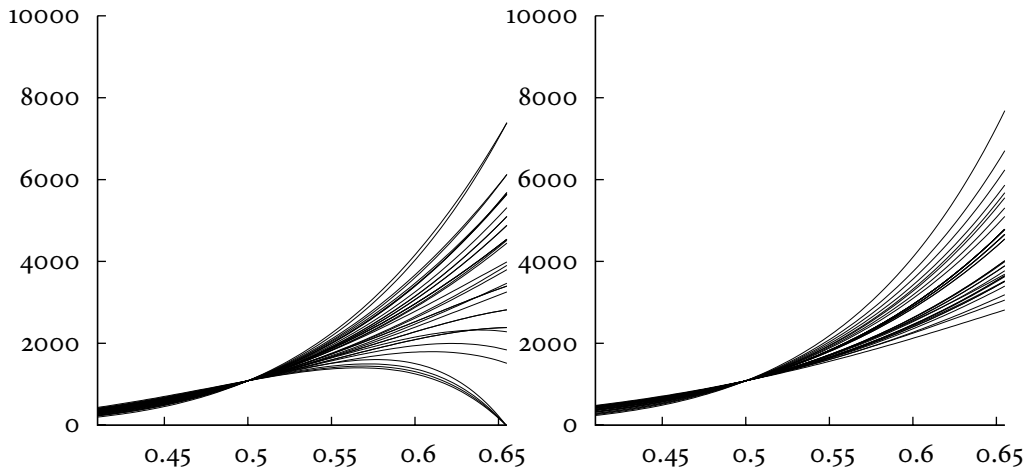


Figure 5.7: *Plots of P_j (left) and Q_j (right) over $\Lambda \approx [0.410, 0.655)$. Note that all polynomials have a common point at $\lambda = 1/2$.*

We now use Sturm sequences for the interval $[0.41, 0.66] \supset \Lambda$ (the rational interval facilitates computation of Sturm sequences) and find either zero roots or one root. In the case of one root, evaluation of the polynomials shows that the root is at $(3 + \sqrt{5})/8$. Since this value is outside Λ, no polynomial has roots within Λ. Evaluation at $\lambda = 1/2$ yields

$$P_j(1/2) = Q_j(1/2) = 1085$$

for $j = 1, \ldots, 36$. Since $s_n > 0$ for $n \geq 3$, both real and imaginary parts of K_j are *positive*. Hence, we can apply Corollary 4.2 and get the following theorem:

Theorem 5.4 (Smoothness of Catmull-Clark scheme)**.** *Let $n \geq 3$ and $c_n := \cos(2\pi/n)$. Furthermore, let*

$$\lambda := \frac{1}{16}\left(c_n + 5 + \sqrt{(c_n + 9)(c_n + 1)}\right) \tag{5.31}$$

and

$$\lambda^0_{1,2} := \frac{1}{8}\left(4\alpha - 1 \pm \sqrt{(4\alpha - 1)^2 + 8\beta - 4}\right). \tag{5.32}$$

Then the limit surface of the Catmull-Clark algorithm with weights α, β, and $\gamma := 1 - \alpha - \beta$ is a regular C^1-manifold for almost all input data if and only if

$$\lambda > \max\left(\left|\lambda^0_1\right|, \left|\lambda^0_2\right|\right).$$

Proof. This follows directly from the arguments of the preceding paragraphs: We have shown that we can represent the coefficients of Ψ^0_v as polynomials that are positive on their domain. By the convex hull property of Bézier surfaces, the polynomials (and thus the control points of Ψ^0_v) will attain *positive values* only. Hence, Corollary 4.2 is applicable and proves that the characteristic map is *regular* and *injective*. By Theorem 4.3, the limit surface is a regular C^1-manifold for almost all input data. ∎

5.3.4 Original weights

In this section, we perform the detailed analysis for the original weights of Catmull and Clark [CC78]. We show that λ is the subdominant eigenvalue and consequently, by Theorem 5.4 the Catmull-Clark subdivision algorithm will produce regular surfaces.

Proposition 5.2 (Original weights of the Catmull-Clark subdivision scheme)**.** *Let the weights of the Catmull-Clark scheme be*

$$\alpha = 1 - \frac{7}{4n}, \quad \beta = \frac{3}{2n}, \quad \gamma = \frac{1}{4n}, \tag{5.33}$$

which are the weights originally proposed by Catmull and Clark [CC78]. Then the eigenvalue λ as defined above is the subdominant eigenvalue.

Proof. We need to show that Equation 5.24 holds for λ. For the Catmull-Clark weights, the eigenvalues $\lambda_{1,2}^0$ are both positive. Starting with λ_2^0, we have

$$\begin{aligned} 16(\lambda - \lambda_2^0) &= 5 + c_n + \sqrt{(c_n+9)(c_n+1)} - 6 + \frac{14}{n} + 2\sqrt{5 + \frac{49}{n^2} - \frac{30}{n}} \\ &> -\frac{3}{2} + \frac{1}{2}\sqrt{19} + 2\sqrt{5} > 5. \end{aligned}$$

Thus, $\lambda > \lambda_2^0$.

For λ_1^0, the proof is more technical. Calculating the values for $n = 3$ shows that $\lambda > \lambda_1^0$, as claimed. For $n > 3$, we have an estimate of the cosine, namely

$$c_n := \cos\left(\frac{2\pi}{n}\right) > 1 - \frac{2\pi^2}{n^2},$$

which follows from the power series of the cosine. Using this estimate, we obtain a lower estimate on $\lambda - \lambda_1^0$. If we take $\lambda - \lambda_1^0$ as a polynomial in n, we find that it has no real roots. Consequently, the function does not change its sign. Evaluation at one point in order to determine the sign yields a value greater zero. Therefore, $\lambda > \lambda_1^0$. ∎

Having shown that λ is the subdominant eigenvalue for the original weights, the Catmull-Clark scheme in its original form produces regular limit surfaces as an application of Theorem 5.4.

5.3.5 Degenerate weights

Following the discussion of the original weights, we now turn to *degenerate weights* that will yield irregular limit surfaces.

Theorem 5.5 (Degenerate version of the Catmull-Clark algorithm)**.** *Let $\alpha = 1$, $\beta = 0$, $\gamma = 0$. Then the Catmull-Clark algorithm will* not *produce regular surfaces.*

Proof. Calculating the eigenvalues λ_1^0 and λ_2^0 yields

$$\lambda_{1,2}^0 = \frac{1}{8}\left(3 \pm \sqrt{5}\right). \tag{5.34}$$

The maximum value $\lambda_{\max}$ of λ, however, is given by

$$\lambda_{\max} < \frac{1}{8}\left(3+\sqrt{5}\right) \tag{5.35}$$

because $c_n = 1$ is *never* attained. Consequently, $\lambda_1^0 > \lambda$ and $\left|\lambda_1^0 - \lambda\right| \to 0$ as $n \to \infty$. Thus, Equation 5.24 is violated and by Theorem 5.4, the algorithm will *not* produce regular surfaces. ■

5.3.6 Permissible weights

We have seen that the Catmull-Clark subdivision algorithm generates smooth surfaces as long as λ is the subdominant eigenvalue. Therefore, we conclude this chapter with an examination of permissible weights α, β, and γ such that λ is the subdominant eigenvalue.

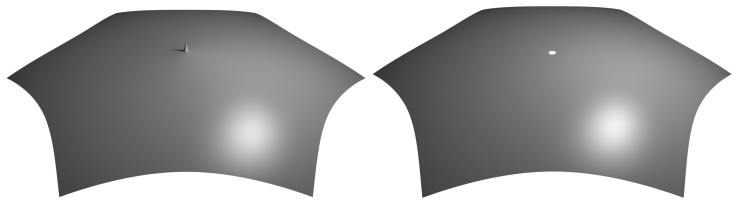
Using $\alpha, \beta \geq 0$ and $1-\alpha-\beta \geq 0$, we get the constraint $\alpha+\beta \leq 1$, which limits the region of all permissible weights. Solving $(4\alpha-1)^2 + 8\beta - 4 \geq 0$, which is the discriminant of the expression in Equation 5.19, yields $-2\alpha^2 + \alpha + 3/8 \leq \beta$. Hence, the parabola $\beta = -2\alpha^2 + \alpha + 3/8$ is the boundary of the set of weights for which λ_1^0 and λ_2^0 are complex numbers. We now want to derive constraints for α and β such that Equation 5.24 holds. The minimum of λ is attained for $c_n = -1/2$ and evaluates to $\lambda_{\min} = (\sqrt{17}+9)/32$. Thus, we need to solve $\lambda_{\min} - \left|\lambda_{1,2}^0\right| = 0$ in order to find the region of weights for which Equation 5.24 *cannot* hold. It turns out that there are two solutions, namely $\beta = 1/64\left(144\alpha + 45 + 5\sqrt{17} + 16\alpha\sqrt{17}\right)$, which is outside the region, and $\beta = 1/64\left(-144\alpha + 117 - 16\alpha\sqrt{17} + 13\sqrt{17}\right)$, which is inside the region of permissible weights. Having obtained all required equations, we plot the region of feasible weights. The result is depicted in Figure 5.9. Thus, if the weights are contained inside the shaded region *and* do not lie outside the smaller region that starts at $\alpha \approx 0.8$, then λ is the subdominant eigenvalue.

However, this condition is not *necessary*. The original weights from Catmull and Clark, for example, lie outside the region for almost every valency. Yet, we have been able to prove that λ is the subdominant eigenvalue. Figure 5.10 depicts this fact by showing the asymptotic behaviour of λ and $\lambda_{1,2}^0$ in addition to the original weights for different valencies.

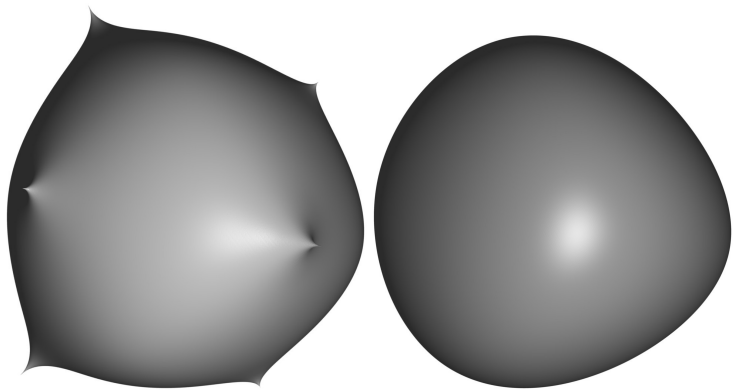
5.4 Summary

In this chapter, we analysed smoothness properties of the Doo-Sabin and the Catmull-Clark subdivision algorithms. We started with general considerations for both schemes, examined their eigenstructure, and formulated necessary conditions for their eigenvalues. Afterwards, we proved that the limit surfaces produced by the *original variants* of both algorithms are regular C^1-manifolds for almost all input data. We then examined *degenerate weights* for both schemes and demonstrated their effects on several common meshes. At last, we gave an overview of *permissible weights* and, for the Catmull-Clark scheme, characterized them graphically.

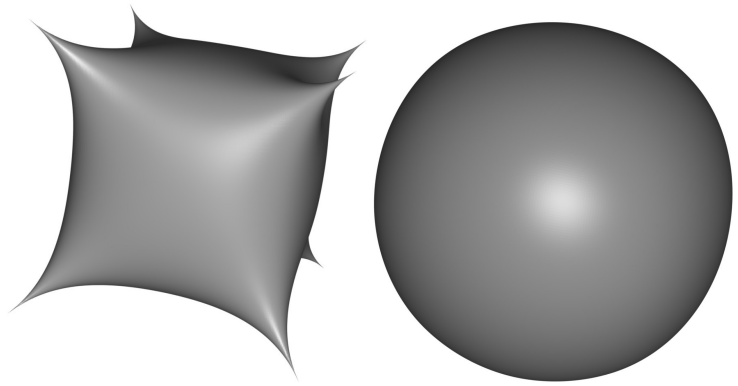
Retrospectively, it was rather unexpected that analysing the Doo-Sabin scheme required *more* work than the Catmull-Clark scheme—after all, having only one subdivision stencil, the Doo-Sabin algorithm is much simpler in comparison. Even the characterization of feasible weights for the Doo-Sabin scheme is more involved than the straightforward eigenvalue criterion of the Catmull-Clark scheme.



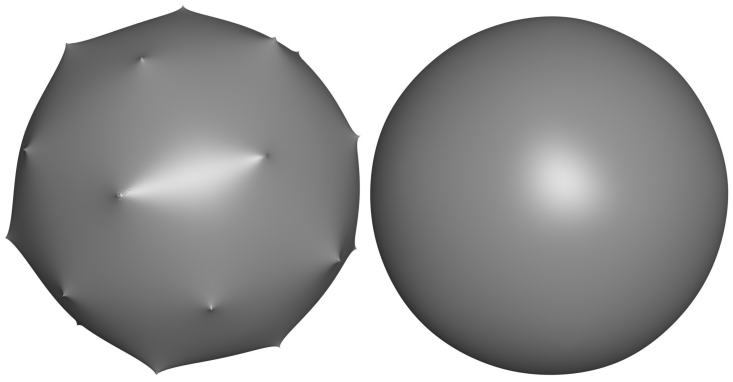
(a) 6-*sided hole*



(b) *Tetrahedron*



(c) *Hexahedron*



(d) *Icosahedron*

Figure 5.8: *Results of the Catmull-Clark algorithm using degenerate weights (left) and original weights (right) on common meshes. The same number of subdivision steps was applied for both weight schemes. The degenerate variant of the* 6*-sided hole has been scaled so that the single "spike" in the centre of the surface, which is the limit point of the extraordinary vertex, is shown.*

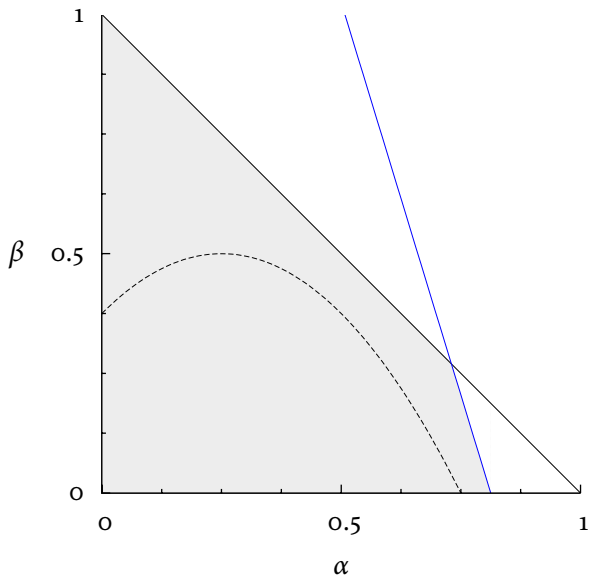


Figure 5.9: *A plot of the region of permissible weights for the Catmull-Clark scheme. The dotted parabola designates the boundary of the values for α and β such that λ_1^0 and λ_2^0 are complex conjugates. If the weights are inside the grey region bounded by the blue line and the black line, then the eigenvalue λ is guaranteed to be the subdominant eigenvalue of the Catmull-Clark subdivision algorithm. Note that this is a* sufficient *condition, but not a* necessary *one.*

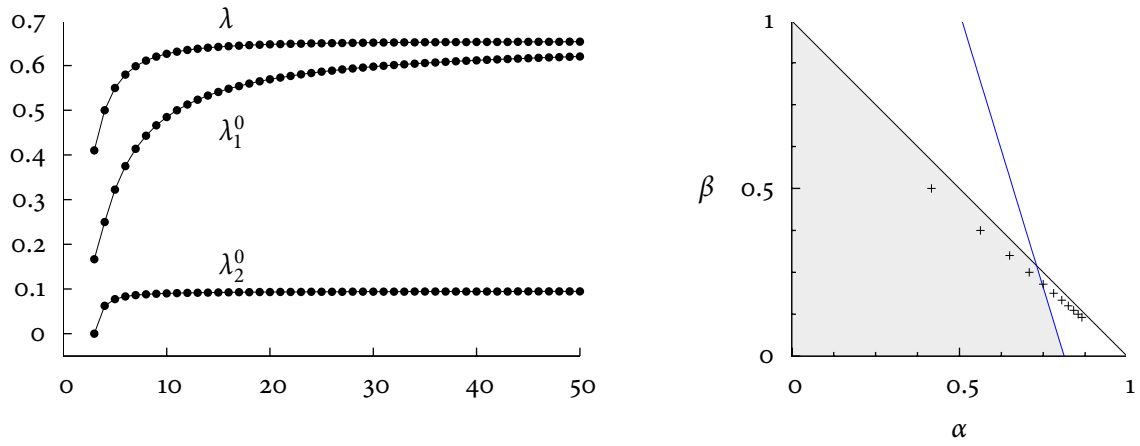


Figure 5.10: *The left figure shows the asymptotic behaviour of the eigenvalues λ, λ_1^0, and λ_2^0 for the original weights of the Catmull-Clark subdivision algorithm. In Section 5.3.4, we have proved that λ is the subdominant eigenvalue for these weights. The right figure, however, shows the behaviour of the weights for increasing valencies—since they approach the point $(1, 0)$, almost all of them are outside the permissible region. This demonstrates that the condition derived in this section is indeed only* sufficient.

A Additional mathematical background

This appendix contains some proofs that would distract too much from the main matter. Furthermore, it provides some background concerning *Euler's formula* and *Sturm sequences*.

A.1 Euler's formula for convex polyhedrons

For any convex polyhedron with V vertices, E edges, and F faces,

$$V - E + F = 2. \tag{A.1}$$

This result is known as Euler's formula for convex polyhedrons. It may be used to show that extraordinary vertices inevitably occur when trying to model more complicated shapes. Furthermore, it is employed by Doo [Doo78] in order to prove that the new vertices created by the Doo-Sabin subdivision algorithm have valency $k = 4$.

For a proof of this formula, see Fulton [Ful95], p. 244. We do not require this formula but cite it for reference purposes only.

A.2 Proofs for eigenvalue estimates

Lemma A.1. *For $k = 1, \ldots, n-1$, we have $1/4 \geq \lambda_2^k > 1/16$, where*

$$\lambda_2^k := \frac{1}{16}\left(c_{n,k} + 5 - \sqrt{(c_{n,k}+9)\,(c_{n,k}+1)}\right).$$

Proof. Since $c_{n,k} \in [-1, 1]$ and λ_2^k is *decreasing* for increasing values of $c_{n,k}$, it is sufficient to check these conditions at the boundaries. For $c_{n,k} = -1$, we have $\lambda_2^k = 1/4$. Likewise, for $c_{n,k} = +1$, we have $\lambda_2^k = \left(6 - \sqrt{20}\right)/16 > 1/16$. ∎

Lemma A.2. *For $k = 1, \ldots, n-1$, we have $1 > \lambda > 1/4$, where*

$$\lambda := \frac{1}{16}\left(c_n + 5 + \sqrt{(c_n+9)\,(c_n+1)}\right).$$

Proof. Again, it is sufficient to check the condition at the boundaries. For $n \geq 3$, we have $c_n > -1$. Furthermore, the eigenvalue λ is *increasing* for increasing values of c_n. Hence, $\lambda > 1/4$ as $c_n \to -1$ and $\lambda = (6 + \sqrt{20})/16 < 1$ for $c_n = +1$. ∎

Corollary A.1. *Since $c_{n,k} \in [-1,1]$, the claim from Lemma A.2 can be modified, thereby obtaining the estimate $1 > \lambda_1^k \geq 1/4$, where $k = 2,\ldots,n-2$ and*

$$\lambda_1^k := \frac{1}{16}\left(c_{n,k} + 5 + \sqrt{(c_{n,k}+9)\,(c_{n,k}+1)}\right).$$

Lemma A.3. *For $k = 2,\ldots,n-2$, we have $\lambda > \lambda_1^k$.*

Proof. For $n = 3$, nothing is to be shown. For $n > 3$, it is sufficient to prove $c_n > c_{n,k}$. To this end, we expand c_n and $c_{n,k}$ and apply the sum-to-product formula. This yields

$$\cos(2\pi/n) - \cos(2\pi k/n) = -2\sin\left(\frac{\pi(k+1)}{n}\right)\sin\left(\frac{\pi(1-k)}{n}\right).$$

Since $k = 2,\ldots,n-2$ and $n > 3$, we have $\sin(\pi(k+1)/n) > 0$ and $\sin(\pi(1-k)/n) < 0$. Therefore,

$$-2\sin\left(\frac{\pi(k+1)}{n}\right)\sin\left(\frac{\pi(1-k)}{n}\right) > 0.$$

■

A.3 Sturm sequences

This section quotes the most important results concerning Sturm sequences. In the main matter, we require these sequences in order to prove that certain polynomials do not have any roots within certain intervals. The following results are quoted from Stoer and Bulirsch [SB96].

Definition A.1 (Sturm sequence)**.** A sequence $p(x) = p_0(x), p_1(x), \ldots, p_m(x)$ of real polynomials is a *Sturm sequence* for the polynomial $p(x)$ if the following conditions hold:

1. All real roots of $p_0(x)$ are simple.
2. $\operatorname{sign} p_1(\xi) = -\operatorname{sign} p_0'(\xi)$ if ξ is a *real* root of $p_0(x)$.
3. If ξ is a *real* root of $p_i(x)$, then $p_{i+1}(\xi)p_{i-1}(\xi) < 0$ for $i = 1,2,\ldots,m-1$.
4. The last polynomial $p_m(x)$ has *no* real roots.

Sturm sequences yield an important theorem about the number of real roots. We will use this theorem to prove that the characteristic map of the Catmull-Clark scheme is regular.

Theorem A.1. *The number of* real *roots of $p(x) = p_0(x)$ in the interval $a \leq x < b$ is equal to $w(b) - w(a)$, where $w(x)$ is the number of changes of sign of a Sturm sequence $p_0(x),\ldots,p_m(x)$ at location x.*

Proof. See Stoer and Bulirsch [SB96], p. 298–299. ■

Sturm sequences can be constructed for any polynomial $p(x)$. In [SB96], Stoer and Bulirsch provide an algorithm for polynomials with *simple* real roots only. However, this algorithm can be extended to polynomials with multiple roots by determining the greatest common divisor $q(x)$ of a polynomial $p(x)$

and its derivative $p'(x)$. Since $p(x)/q(x)$ has the same roots as $p(x)$ but no multiple roots anymore, a Sturm sequence may be determined for this polynomial by using the aforementioned algorithm.

In this thesis, we will only use the *result* of this theorem. Since the calculations are rather complex, the help of computer algebra systems is employed.

B Visualization of subdivision algorithms

In order to visualize the algorithms analysed in this thesis, the author implemented Doo-Sabin, Catmull-Clark, and Loop subdivision (the latter was included for reference purposes). The implementations were combined in `psalm`, which is short for "**p**retty **s**ubdivision **al**gorithms on **m**eshes". This chapter gives a short overview of `psalm` and provides algorithms for preserving the orientation of meshes. It concludes with some example images created by `psalm` and `Blender`.

B.1 Description of `psalm`

`psalm` is a C++ command-line interface program. Its task is to *process* mesh input data by applying subdivision algorithms. The result is to be stored in a file for further processing. The following paragraphs briefly describe `psalm`'s features.

Mesh compiler `psalm` is a mesh compiler and does *not* contain rendering capabilities on its own. Instead, it produces output as `PLY` files, `Wavefront OBJ` files, or `Geomview OFF` files (see Figure B.1 for a quick comparison between the three formats). The user may then use any rendering software that is able to process one of these file formats. The author, for example, decided to use `Blender` for this purpose. `Blender` is an open-source 3D content creation suite that offers professional rendering algorithms for many purposes. In this thesis, all images depicting subdivision algorithms have been rendered by `Blender`.

Tuning of subdivision algorithms `psalm` allows the user to *fine-tune* subdivision algorithms. Users may change the weights for k-sided faces or vertices with valency k (without changing the code), or (with small code modifications) even implement their own weight schemes for a subdivision algorithm. This feature has been exploited in Section 5.2.4, for example, where a degenerate weight function has been added in order to perturb the Doo-Sabin algorithm.

Pruning of meshes `psalm` provides rudimentary pruning functions for meshes. It can delete vertices of certain valencies or faces with a certain number of sides. Pruning is very helpful for the removal of irregular parts of the mesh.

Orientation preservation `psalm` takes great care in order to preserve the orientation of input meshes. Given a consistently oriented input mesh, refined parts of the mesh will be oriented according to the initial orientation. Thus, the refined mesh maintains its initial orientation at every subdivision step. We will discuss the importance of preserving the orientation in the next section.

```
ply
format ascii 1.0
element vertex 8
property float x
property float y
property float z
element face 6
property list uchar int index
end_header
-0.5 -0.5 0.5
0.5 -0.5 0.5
-0.5 0.5 0.5
0.5 0.5 0.5
-0.5 0.5 -0.5
0.5 0.5 -0.5
-0.5 -0.5 -0.5
0.5 -0.5 -0.5
4 0 1 3 2
4 2 3 5 4
4 4 5 7 6
4 6 7 1 0
4 1 7 5 3
4 6 0 2 4
```

(a) *PLY data*

```
OFF
8 6 0
-0.5 -0.5 0.5
0.5 -0.5 0.5
-0.5 0.5 0.5
0.5 0.5 0.5
-0.5 0.5 -0.5
0.5 0.5 -0.5
-0.5 -0.5 -0.5
0.5 -0.5 -0.5
4 0 1 3 2
4 2 3 5 4
4 4 5 7 6
4 6 7 1 0
4 1 7 5 3
4 6 0 2 4
```

(b) *OFF data*

```
v -0.5 -0.5 0.5
v 0.5 -0.5 0.5
v -0.5 0.5 0.5
v 0.5 0.5 0.5
v -0.5 0.5 -0.5
v 0.5 0.5 -0.5
v -0.5 -0.5 -0.5
v 0.5 -0.5 -0.5
f 1 2 4 3
f 3 4 6 5
f 5 6 8 7
f 7 8 2 1
f 2 8 6 4
f 7 1 3 5
```

(c) *OBJ data*

Figure B.1: *Input data for a regular hexahedron. Note how all file formats represent the* same *object quite differently. When parsing the OBJ format, for example, the number of vertices and faces is unknown beforehand.*

B.2 Implementation details

This section expands on the most important facts of `psalm`'s implementation of subdivision algorithms. The code itself is deliberately *not* cited as it would be outside the scope of this thesis. Since the source code is provided under a simplified BSD licence, the interested reader is referred to `http://bastian.rieck.ru/research/diploma/psalm` for more details.

B.2.1 Orientation of meshes

The file formats processed by `psalm` describe meshes by maintaining a list of *vertices*, which are points in $\mathbb{R}^3$, and a list of *faces*, which are k-tuples of indices corresponding to vertices. The *edges* of the mesh are implicitly described by the k-tuples: Traversing the components of a tuple, each two adjacent vertices form an edge. The last vertex is supposed to be connected with the first one, thereby "closing" the face. A mesh that is correctly oriented will contain each edge twice (once for both possible directions of the edge). Thus, the *orientation* of faces is well-defined.

In computer graphics, faces are assumed to be oriented in *counter-clockwise* order because this orientation allows application of the *right-hand rule*. The orientation of polygonal faces is used for lighting and shading calculations—consequently, incorrectly oriented faces may appear unlit, thereby giving a larger object a "faceted" appearance. Figure B.2 depicts this problem.

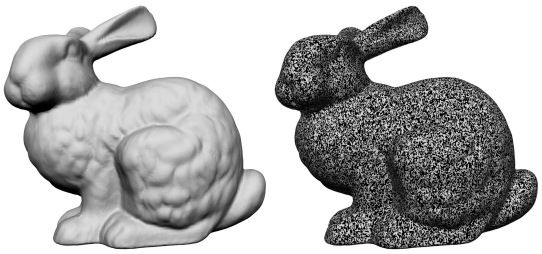


Figure B.2: *The left picture shows the "Stanford Bunny" mesh with correctly oriented normals. In the right picture, the normals have been flipped randomly. The result is a faceted appearance when rendering the mesh.*

B.2.2 Storing edges

For subdivision algorithms, care must be taken *not* to store the same edge twice. The edge point stencil for the Catmull-Clark subdivision scheme, for example, requires that an interior edge is part of exactly two faces. Thus, an implementation of this scheme needs to contain a method for quickly determining the two faces an interior edge is part of. Storing an edge twice (with only one adjacent face) would complicate this—in the worst case, all edges would have to be searched for the corresponding second edge.

To avoid duplicate edges, `psalm` maintains an internal hash map that uses edge IDs to access the list of edges. An edge ID of an edge (u, v) is calculated by sorting the vertex IDs such that $u \leq v$, and then taking the ID to be the pair (u, v). Upon loading a mesh, `psalm` uses this ID to check whether an edge has already been added to the hash map. If this is the case, `psalm` signals that the edge direction must be *reversed*. As a consequence, when processing the edges of a face, `psalm` uses pointers to mark the first and (for non-boundary edges) second face an edge is part of.

Assuming that meshes are oriented counter-clockwise, we may refer to the first face as the "left" face of an edge (since the face lies to the left of the edge's direction) and likewise, to the second face as the "right" face of an edge. If the mesh is oriented clockwise, the meaning of "left" and "right" is swapped, but nonetheless, all algorithms presented in this chapter still work.

B.3 Preserving the orientation of a mesh

We have seen that an inconsistent orientation of a mesh may lead to erroneous lighting calculations. Thus, every implementation of a subdivision scheme is required to *preserve* the orientation of the input mesh—assuming that it has been oriented consistently. The following sections describe the methods `psalm` uses to maintain the initial orientation of the mesh when different subdivision schemes are applied.

B.3.1 Doo-Sabin subdivision scheme

Recalling the description of the algorithm from Section 3.4.1, three kinds of faces appear during the Doo-Sabin algorithm. The strategies for preserving the orientation vary accordingly.

F-faces New vertices of every face can be connected in the order of the old vertices of the face. If the initial mesh is oriented consistently, this procedure works—see Algorithm B.1 for details.

Algorithm B.1 Preserving the orientation of F-faces

1: **for all** Face f **do**
2: **for all** Vertices v of f **do**
3: Connect the new vertices v' in the order of the old vertices.
4: **end for**
5: **end for**

E-faces When loading a mesh, `psalm` stores the face F that is encountered when traversing the edge (u, v). For an E-face, we require that the edge is adjacent to a second face, which we denote by G. This is the face that is encountered when traversing the edge (v, u), which is called the "inverted edge" in `psalm`'s terminology. The procedure is described by Algorithm B.2. Figure B.3 depicts a visual explanation.

Algorithm B.2 Preserving the orientation of E-faces

1: **for all** Interior edges (u, v) in the mesh **do**
2: Find first adjacent face F.
3: Find second adjacent face G.
4: Form a face by connecting vertices u_F, u_G, v_G, v_F, and u_F.
5: **end for**

V-faces Interior vertices are part of several faces. These faces need to be sorted in counter-clockwise order around the vertex. Then the new vertices that correspond to the face and the vertex can be connected accordingly. The sorting process is described by Algorithm B.3.

B.3.2 Catmull-Clark subdivision scheme

In the Catmull-Clark algorithm, new faces are formed by connecting vertex points, edge points, and face points. Thus, we can determine the correct order by using the given orientation of the mesh. To this end, edges and faces around a given vertex need to be enumerated in a consistent direction. The enumeration process is accomplished by Algorithm B.4.

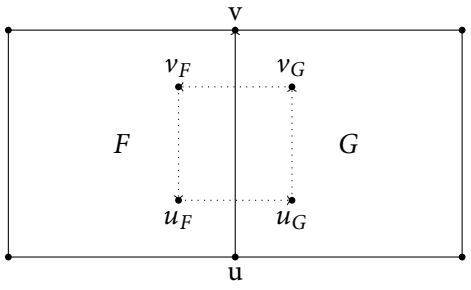


Figure B.3: *Illustration of Algorithm B.2*

Algorithm B.3 Preserving the orientation of V-faces

for all Vertices v **do**
  Enumerate all faces:
  Store first face of first incident edge.
  for all Incident edges e of v, starting from the second edge **do**
    if First face of e is equal to one of the faces of the previous edge **then**
      Store second face of edge e.
    else
      Store first face of edge e.
    end if
  end for
  Check orientation of enumerated faces:
  if v is *start* vertex of first edge **then**
    Orientation is wrong if the second face is to the right of the first edge.
  else
    Orientation is wrong if the second face is to the left of the first edge.
  end if
  If the orientation was found to be wrong, reverse the order of the list of stored faces.
end for

Algorithm B.4 Preserving orientation of faces in the Catmull-Clark algorithm

for all Vertices v **do**
  for all Faces f that v is a part of **do**
    Find the two incident edges e_1 and e_2 belonging to f.
    if e_1 does not have an edge point **or** e_2 does not have an edge point **then**
      Continue with the next face.
    end if
    if (v is start vertex of e_1 **and** f is to the right of e_1) **or** (v is end vertex of e_1 **and** f is to the left of v) **then**
      Swap e_1 and e_2.
    else if (v is start vertex of e_2 **and** f is to the left of e_2) **or** (v is end vertex of e_2 **and** f is to the right of v) **then**
      Swap e_1 and e_2.
    end if
    Form a new face by connecting the vertex point of v, the edge point of e_1, the face point of f, and the edge point of e_2.
  end for
end for

Input mesh	No. of steps	Initial size	Catmull-Clark
Tetrahedron	8	4 vertices 6 edges 4 faces	196610 vertices 393216 edges 196608 faces
Hexahedron	8	8 18 12	589826 1179648 589824
Icosahedron	8	12 30 20	983042 1966080 983040
Klein Bottle I	3	2500 7500 5000	240000 480000 240000
Klein Bottle II	3	2500 5000 2500	160000 320000 160000
Dragon	4	1697 5088 3388	646821 1293267 646501
6-sided hole	8	36 36 13	101395 200466 99846
12-sided hole	8	36 54 19	202771 400914 199686

Table B.1: *Sizes of common meshes after performing a number of subdivision steps. The three numbers in columns "Initial size" and "Catmull-Clark" indicate the number of vertices, edges, and faces of the mesh* before *and* after *subdivision. Note the difference in the meshes "Klein Bottle I" and "Klein Bottle II". The first mesh is a triangulated version of the second mesh that consists solely of quadrangles.*

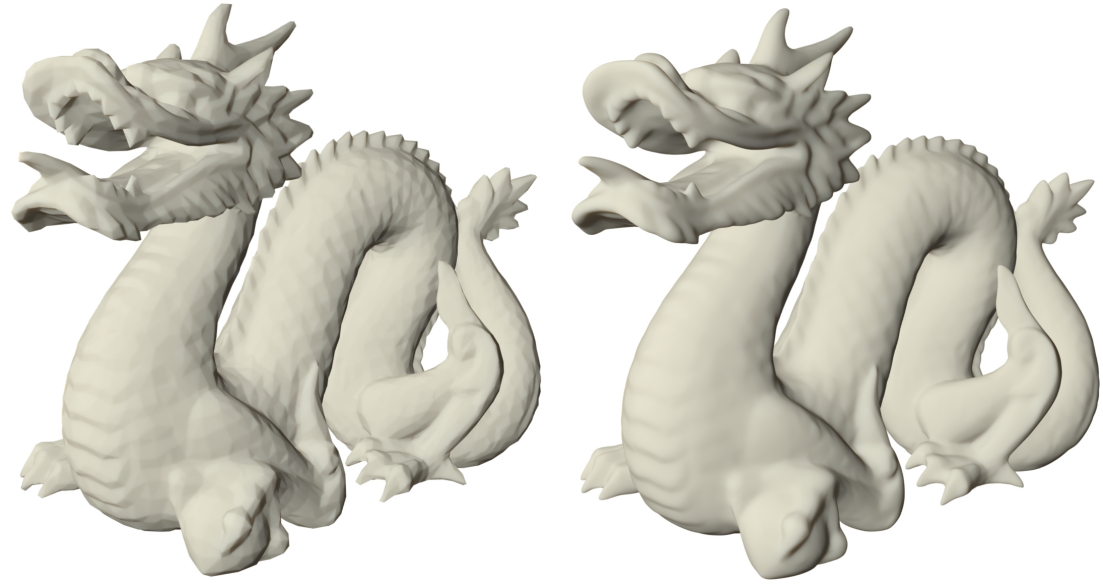
B.4 Performance and examples

This section contains examples of `psalm`'s performance. Since subdivision schemes require access to the complete topology of a mesh, they are rather complex algorithms. Table B.1 shows the increase in vertices, edges, and faces for some typical meshes. The time required for several subdivision steps applied to these meshes is shown in Table B.2.

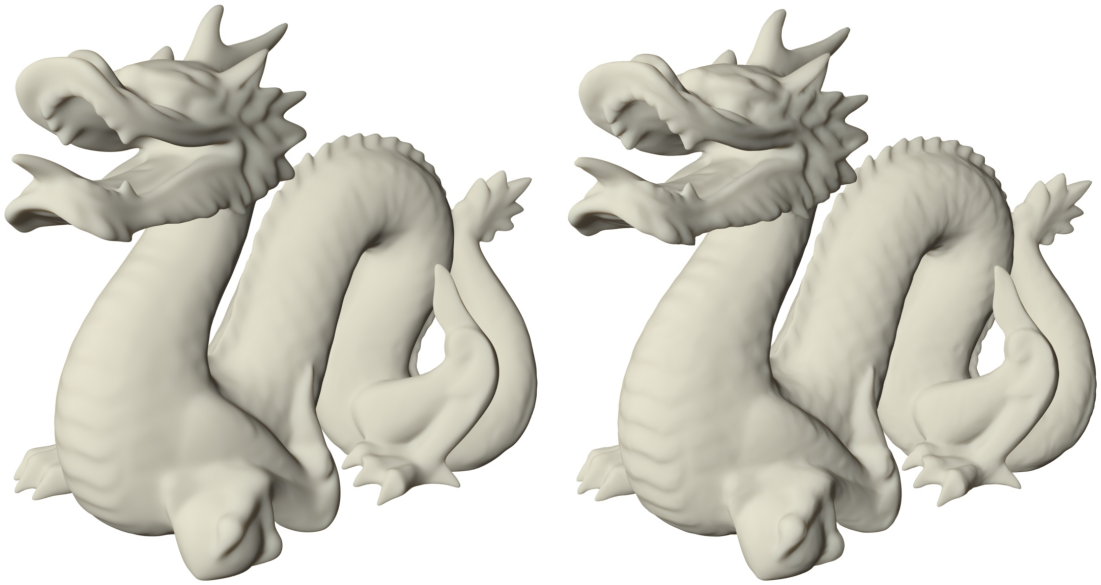
We conclude this chapter with an example of `psalm`'s capabilities. Figure B.4 depicts a simplified version of the "Stanford Dragon" mesh provided by Stanford Computer Graphics Laboratory.

Input mesh	No. of steps	Doo-Sabin	Catmull-Clark	Loop
Tetrahedron	4	0.0078125	0.0078125	0.0078125
	6	0.15625	0.140625	0.140625
	8	2.64062	2.50781	2.34375
Hexahedron	4	0.0234375	0.0234375	0.0234375
	6	0.46875	0.4375	0.429688
	8	8.13281	7.75781	7.20312
Icosahedron	4	0.046875	0.0390625	0.0390625
	6	0.804688	0.75	0.726562
	8	13.6484	13.0781	12.1328
Klein bottle I	1	0.132812	0.125	0.125
	2	0.695312	0.648438	0.65625
	3	3.10938	2.92969	2.85156
Klein bottle II	1	0.0859375	0.09375	
	2	0.460938	0.4375	
	3	2.0625	1.94531	
Dragon	2	0.46875	0.453125	0.484375
	3	2.07812	1.99219	2.0
	4	8.70312	8.52344	8.1875
6-sided hole	6	0.304688	0.0703125	
	7	1.26562	0.304688	
	8	5.17969	1.26562	
12-sided hole	6	0.539062	0.15625	
	7	2.23438	0.632812	
	8	9.16406	2.57812	

Table B.2: *Performance of* `psalm` *when operating on several example meshes. The unit of the measurements is CPU time in seconds. Performance was measured on an Intel Celeron M* 1.4 *GHz processor. The table only contains data for* subdividing *the mesh. The overhead from* loading *the initial mesh and* storing *the resulting mesh has not been included because it would skew the results (different mesh formats can be processed at different speeds). Blank lines in the Loop column indicate that the mesh does contain non-triangular faces, which makes the Loop scheme inapplicable. As a result, we see that the Catmull-Clark is slightly faster than the Doo-Sabin algorithm, especially when being used to fill n-sided holes. This is due to the fact that the new topology of meshes with n-sided holes may be determined much more quickly for the Catmull-Clark algorithm, which generates quadrangles only, than for the Doo-Sabin algorithm, which generates non-quadrangular faces.*



(a) *Original* **(b)** *Catmull-Clark*



(c) *Loop* **(d)** *Doo-Sabin*

Figure B.4: *A simplified version of the "Stanford Dragon" mesh provided by Stanford Computer Graphics Laboratory. In clockwise order, starting from the upper left image, three steps of the Catmull-Clark, the Doo-Sabin, and the Loop subdivision scheme are depicted. Note how the different schemes smooth details of the mesh. The Loop scheme, for example, does* not *preserve the eyes and scales of the dragon very well when compared to the Catmull-Clark scheme.*

Bibliography

[Böh81] Wolfgang Böhm, *Generating the Bézier points of B-spline curves and surfaces*, Computer-Aided Design **13** (1981), no. 6, 365–366.

[BS88] Alan A. Ball and David J. T. Storry, *Conditions for tangent plane continuity over recursively generated B-spline surfaces*, ACM Transactions on Graphics **7** (1988), no. 2, 83–102.

[Cat74] Edwin Catmull, *A subdivision algorithm for computer display of curved surfaces*, Ph.D. thesis, University of Utah, December 1974.

[CC78] Edwin Catmull and James Clark, *Recursively generated B-spline surfaces on arbitrary topological meshes*, Computer-Aided Design **10** (1978), no. 6, 350–355.

[Cha74] George Merrill Chaikin, *An algorithm for high-speed curve generation*, Computer Graphics and Image Processing **3** (1974), no. 4, 346–349.

[Dav94] Philip J. Davis, *Circulant matrices*, 2nd ed., Chelsea Publishing, 1994.

[dB72] Carl de Boor, *On calculating with B-splines*, Journal of Approximation Theory **6** (1972), no. 1, 50–62.

[DLG90] Nira Dyn, David Levin, and John A. Gregory, *A butterfly subdivision scheme for surface interpolation with tension control*, ACM Transactions on Graphics **9** (1990), no. 2, 160–169.

[Doo78] Daniel Doo, *A subdivision algorithm for smoothing down irregular shaped polyhedrons*, Proceedings on Interactive Techniques in Computer-Aided Design, 1978, pp. 157–165.

[DS78] Daniel Doo and Malcom Sabin, *Behaviour of recursive division surfaces near extraordinary points*, Computer-Aided Design **10** (1978), no. 6, 356–360.

[EB88] Leonhard Euler and John D. Blanton, *Introduction to analysis of the infinite: Book I*, Springer-Verlag, 1988.

[Far96] Gerald Farin, *Curves and surfaces for computer-aided geometric design: A practial guide*, 4th ed., Academic Press, 1996.

[FHK02] Gerald Farin, Josef Hoschek, and Myung-Soo Kim (eds.), *Handbook of computer aided geometric design*, North Holland, 2002.

[Ful95] William Fulton, *Algebraic topology*, Graduate Texts in Mathematics, Springer-Verlag, 1995.

[Kob00] Leif Kobbelt, $\sqrt{3}$-*subdivision*, SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques, ACM Press/Addison-Wesley Publishing Co., 2000, pp. 103–112.

[KPR04] Kęstutis Karčiauskas, Jörg Peters, and Ulrich Reif, *Shape characterization of subdivision surfaces—case studies*, Computer Aided Geometric Design **21** (2004), no. 6, 601–614.

[Lip81] John D. Lipson, *Elements of algebra and algebraic computing*, The Benjamin/Cummings Publishing Company, Inc., 1981.

[Loo87] Charles Loop, *Smooth subdivision surfaces based on triangles*, Master's thesis, University of Utah, August 1987.

[Por10] Julia Portl, *Subdivision curve and surface fitting for mesh compression*, Wissenschaftliche Arbeit zum 1. Staatsexamen, Ruprecht-Karls-Universität Heidelberg, October 2010.

[PR97] Jörg Peters and Ulrich Reif, *The simplest subdivision scheme for smoothing polyhedra*, ACM Transactions on Graphics **16** (1997), no. 4, 420–431.

[PR98] _____, *Analysis of algorithms generalizing B-spline subdivision*, SIAM Journal on Numerical Analysis **35** (1998), no. 2, 728–748.

[PR04] _____, *Shape characterization of subdivision surfaces—basic principles*, Computer Aided Geometric Design **21** (2004), no. 6, 585–599.

[PR08] _____, *Subdivision surfaces*, Geometry and Computing, Springer-Verlag, 2008.

[PT96] Les Piegl and Wayne Tiller, *The NURBS book*, 2nd ed., Monographs in Visual Communications, Springer-Verlag, 1996.

[Rei95] Ulrich Reif, *A unified approach to subdivision algorithms near extraordinary vertices*, Computer Aided Design **12** (1995), no. 2, 153–174.

[Rei98] _____, *Analyse und Konstruktion von Subdivisionsalgorithmen für Freiformflächen beliebiger Topologie*, Habilitation thesis, Universität Stuttgart, December 1998.

[Rie75] Richard F. Riesenfeld, *On Chaikin's algorithm*, Computer Graphics and Image Processing **4** (1975), no. 3, 304–310.

[Sal05] David Salomon, *Curves and surfaces for computer graphics*, Springer-Verlag, 2005.

[SB96] Josef Stoer and Roland Bulirsch, *Introduction to numerical analysis*, 2nd ed., Texts in Applied Mathematics, Springer-Verlag, 1996.

[Sch96] Jean E. Schweitzer, *Analysis and application of subdivision surfaces*, Ph.D. thesis, University of Washington, 1996.

[Uml00] Georg Umlauf, *Analyzing the characteristic map of triangular subdivision schemes*, Constructive Approximation **16** (2000), no. 1, 145–155.

[Yam88] Fujio Yamaguchi, *Curves and surfaces in computer aided geometric design*, Springer-Verlag, 1988.

[Zor97] Denis Zorin, *Stationary subdivision and multiresolution surface representations*, Ph.D. thesis, California Institute of Technology, September 1997.

[Zor98] _____, *A method for analysis of C^1-continuity of subdivision surfaces*, Tech. Report CSL-TR-98-764, Stanford University, May 1998.

[Zor00] _____, *A method for analysis of C^1-continuity of subdivision surfaces*, SIAM Journal on Numerical Analysis **37** (2000), no. 5, 1677–1708.

Colophon

This thesis has been typeset using LaTeX with the `scrbook` documentclass. The body text is set in Adobe Minion Pro 11pt, a typeface designed by Robert Slimbach in 1990 for Adobe Systems. Furthermore, Donald Knuth's typewriter font was used. The design of plots has been inspired by Edward Tufte.

Figures have been created by `TikZ`, a package developed by Till Tantau, and `gnuplot`. Meshes have been processed by `psalm` and rendered with `Blender`. All symbolical calculations have been assisted by the `Maxima` computer algebra system.