Andreas Riffnaller-Schiefer, Dipl.-Ing.

A SUBDIVISION APPROACH TO ISOGEOMETRIC ANALYSIS

ANALYSIS, DESIGN AND SIMULATION

Doctoral Thesis

to achieve the university degree of

Doktor der technischen Wissenschaften

submitted to

Graz University of Technology

Supervisor

Univ.-Prof. Dipl.-Ing. Dr.techn. Dieter W. Fellner

Co-Supervisor

Ass.Prof. M.Sc. PhD Ursula Augsdörfer

Institute of Computer Graphics and Knowledge Visualisation

Graz, April 2019

AFFIDAVIT

ABSTRACT

Physical products for everyday life are usually designed using computer-aided design (CAD) software. To validate physical properties like structural stability for those digital designs, computer-aided engineering (CAE) software is used to perform different types of analysis and simulations on the virtual objects. Due to their historic development, the tools and methods for design and engineering are based on different geometrical representations to describe the shape of the designed objects. Analysis, therefore, requires a time consuming conversion of the designed geometry.

To overcome this problem, the concept of *isogeometric analysis* was proposed, where the same geometry representation is used for both CAD and CAE. This eliminates the need to convert between different geometry representations for design and analysis.

This thesis explores the use of isogeometric analysis for typical engineering tasks as well as for the design process itself. To closely link design and analysis, the isogeometric concept is extended to a subdivision based surface representation that combines the properties required for reliable engineering design as well as for precise freeform surface design.

The unified analysis platform developed in this thesis provides seamless design and analysis of freeform surfaces, accessible to a wide range of applications and devices. It is used for structural analysis and optimizations of designed shapes, and it is also employed to create new modeling tools for designers and to enable interactive deformation within virtual worlds.

ACKNOWLEDGMENTS

First, I would like to thank my supervisor Dieter W. Fellner for the opportunity to work on this topic. The research for this thesis was greatly influenced by all of my colleagues here at the Institute of Computer Graphics and Knowledge Visualization at Graz University of Technology and the Visual Computing group at Fraunhofer Austria. In particular, I would like to thank my co-supervisor Ursula Augsdörfer. She layed the foundation for this thesis by introducing me to the topic of isogeometric analysis. Her continued support and her valuable feedback and ideas throughout the years significantly influenced the research presented in this thesis.

Finally, I am deeply grateful for my family. I would like to thank my wife Maria for her continued support and for everything she does, for me and our family, and my kids Cassandra and Benjamin for their inspiring curiosity.

Research for parts of this thesis was also supported by the following funded projects:

SIGMACADE:
: Marie Curie Intra-European Fellowships (MC-IEF): project ID 254944

SIGMA CAD:
: Austrian Research Promotion Agency (FFG): project number 843637

Creation of a base mesh from captured data:
: Austrian Science Fund (FWF): P 24481

CONTENTS

Part I

FOUNDATION

"If I have seen further it is by standing on ye sholders of Giants."

Sir Isaac Newton

1 INTRODUCTION

Today, almost all manufactured objects are designed with the support of *computer-aided design (CAD)* software. This simplifies the design process and enables a designer to quickly experiment with different design ideas. However, the physical behavior of the design is often not fully known until a prototype is built and tested. To avoid encountering problems in physical behavior, designers often make overly conservative design decisions.

Independent of the CAD field, mathematical methods and software systems to simulate and analyse real world phenomenons on a model of some domain were developed in the field of engineering. Today, the software tools implementing these methods are commonly called *computer-aided engineering (CAE)* applications. Using CAE tools, the physical behavior of a virtual design can be analysed even before a prototype is build. A common method used in engineering to solve a wide range of these problems is known as the *finite element method (FEM)*, whose use is commonly referred to as *finite element analysis (FEA)*. FEA is used to analyse many different physical behaviors, including, for example, effects of forces on physical structures, transfer of heat or fluid flow.

The basic idea of FEA is to partition the problem domain and the solution space into a set of elements defined by nodes and their associated basis functions. For FEA, the basis functions are typically linear or low degree Lagrange basis functions local to each element. All elements together form the finite element mesh, often also called the simulation mesh, which approximates the original problem domain given by a smooth manifold. A solution for the problem is formulated for each element in terms of its nodal degrees of freedom (DOF). The element solutions are then assembled to define the global solution.

To analyse the physical behavior of a CAD design using CAE software, the CAD representation needs to be meshed into a finite element mesh. This is a time consuming process which, in some cases, is estimated to account for about 80% of the overall analysis time [48]. Further, the finite element mesh is only an approximation of the original smooth CAD design, leading to errors already in the geometry de-

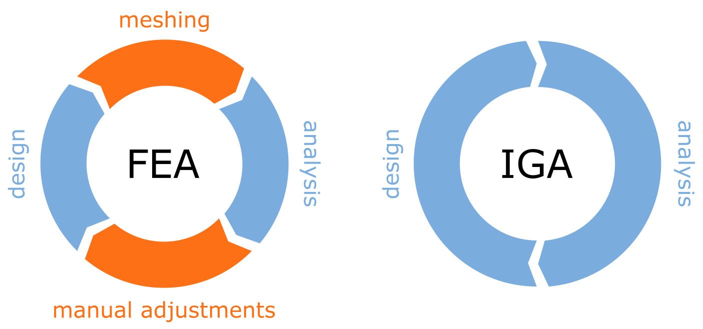


Figure 1.1: For traditional finite element analysis (FEA), the design-analysis-cycle is interrupted by manual or semi-automatic steps (orange) to convert between the different geometry representations used in CAD and FEA. Using isogeometric analysis (IGA), the same geometry representation is used for both design and analysis, enabling a seamless integration.

scription. Another problem is that the finite element mesh is defined by completely different DOF than the CAD design. This complicates the use of the analysis results to improve the design, e.g. displacements or stresses computed on nodes of the finite element mesh cannot simply be transferred onto the CAD design. For the same reason, an automatic optimization of the CAD design is not easily possible because the optimization, using FEA, operates on different DOF.

This gap between design and analysis hinders a fast product development cycle because it requires repeated conversion of the geometry representation, as illustrated in Figure 1.1. Ideally, analysis is integrated into the design process so that designed objects can be tested and analysed already during the early stages of design. In this way, analysis can guide the design to a result satisfying both aesthetics and functionality. This is difficult to achieve with traditional methods, where the required conversion between each design-analysis iteration is time consuming and often requires manual user interaction.

1.1 ISOGEOMETRIC ANALYSIS

To bridge this gap, Hughes *et al.* [48] proposed the concept of *isogeometric analysis (IGA)*. The observation is that, similar to finite element meshes, CAD geometry is defined by elements (faces/patches) consisting of nodes (control points) with associated basis functions. For CAD geometry the basis functions are typically B-spline basis functions or generalizations thereof. The key idea of IGA is to use these basis functions and the DOF of the CAD geometry also for analysis, i.e. to define the problem domain and the solution space. This provides a number of advantages:

- no meshing is required to define the simulation mesh, so
- the exact surface is used for analysis, and
- the degrees of freedom used to solve the problem correspond to the degrees of freedom for the CAD design.

The idea of using the same basis functions for design and analysis is not entirely new, and has already been implemented before the concept of IGA has been proposed by Hughes *et al.* [48], e.g. by Sabin [75] using B-splines or Cirak *et al.* [25] using Loop subdivision surfaces based on generalized box splines. However, the work by Hughes *et al.* [48] sparked further interest in this technique and lead to many applications of IGA for different methods in various fields of research.

One of the main differences to classical FEM is that the basis functions of CAD geometry, i.e. typically B-splines, are not local to each face of the control mesh, i.e. they have support outside of the element. This is in contrast to the linear and Lagrange functions used in FEM, which are defined locally for each element. A consequence of this difference is that elements in IGA typically join in a continuous way, depending on the basis functions, while adjacent FEM elements only share the position of their boundary. This is an advantage of IGA for use cases where higher continuity between elements is required for analysis.

A challenge for IGA is that the DOF and basis functions of the CAD geometry now also need to meet the requirements for analysis. They not only define the input geometry, but also need to be suitable to represent any unknown analysis result with the desired accuracy.

In [48], *Non-Uniform Rational B-Spline (NURBS)* surfaces were used to define the concept of IGA. But NURBS can only represent rectangular domains and require manual layout and alignment of multiple NURBS patches to define more complex geometries. For analysis, this requires maintaining the required continuity between patches and in particular to prevent tearing during deformations. Subdivision surfaces, as used in [25], overcome this limitation and enable the design of an arbitrary topology using a single surface. However, standard subdivision algorithms typically do not provide the precise control over the shape of the surface as with NURBS. Therefore, state-of-the-art IGA requires a compromise when choosing the underlying surface representation: either a continuous freeform surface or multiple patches with precise control. Patches may rip open leading to problems in analysis while small imperfections in the shape may also influence analysis results.

Independent of the surface representation used, IGA may also have additional use cases beyond analysis, because it operates directly on the CAD geometry, using the same DOF. For example, in the context of design, IGA could be seen as a new tool which operates on the DOF of the CAD design to change its shape, similar to existing tools common in design software like a scale tool or a smooth tool.

Open questions in IGA concerning the requirements on the surface representation as well as its potential beyond analysis are the main motivation for the research described in this thesis.

1.2 RESEARCH QUESTIONS

As discussed in the previous section, the choice of geometry representation for the CAD design influences the possible use cases for IGA. So the first question I want to address in this thesis is:

RESEARCH QUESTION 1: *Is it possible to unify design and analysis and at the same time provide exact control over the surface and the flexibility of freeform surfaces?*

In other words, is there a single surface representation that can be used for IGA which provides the flexibility of NURBS while also allowing to easily design freeform surfaces? Having a surface representation combining the advantages of NURBS and subdivision would be a significant advantage for the unified design and analysis of surfaces.

The second question addressed in this thesis, which has already been briefly motivated in the previous section, is:

RESEARCH QUESTION 2: *With a unified design and analysis platform, can these analysis methods, usually used in engineering, also be used for the design process itself?*

By using the same DOF for design and analysis, an IGA computation could also be used as a design tool, i.e. to change the shape of a design. But there are some open questions in this regard, e.g.:

- What analysis methods could be useful for design?
- What are possible use cases for analysis-based tools?
- How can the designer interact with these tools?

These questions need to be answered in order to use IGA also for design, not only for analysis. But if IGA becomes interesting also for other types of applications, besides engineering analysis, there needs to be a way to access IGA features from other software, so the next question is:

RESEARCH QUESTION 3: *How can isogeometric simulation methods be easily integrated into other applications?*

This question addresses both the technical questions of integrating IGA tools into other software as well as requirements for user interfaces to use these tools.

Finally, if it is indeed possible to easily integrate IGA tools into other software, the application of IGA could be extended to different domains, other than CAD and engineering. For example, because virtual worlds become increasingly important, IGA may be used not only to analyse how virtual objects behave in the real world, but also to simulate real world behavior of objects in virtual worlds:

RESEARCH QUESTION 4: *Can isogeometric methods also be used to improve realism in interactive applications?*

The main challenge for this is whether IGA can be adapted for interactive use cases. FEA and IGA are computationally expensive and were traditionally only used in long running offline computations. However, for interactive applications, the simulation must complete within a fraction of a second.

In the remainder of this thesis, these research questions will be successively addressed in more detail.

1.3 THESIS OVERVIEW

Following this introduction, Chapter 2 will present the necessary background on CAD curve and surface representations as well as the foundations of thin shell analysis. These are then used to discuss the existing work on subdivision-based IGA. This concludes the first part of this thesis.

My work, subject of the second part, builds on the foundations layed out in the first part to create an integrated design and analysis platform using subdivision-based IGA. This platform will be used to answer the research questions stated in the previous section. Chapter 3 extends state-of-the-art subdivision-based IGA to a geometry representation combining the advantages of previous approaches. This implementation, used as the basis for the IGA platform, is then verified with several benchmark problems. The combination of design and analysis is then discussed in Chapter 4, where the IGA implementation is integrated into several design applications and where services are presented to provide access to the IGA platform for a wide range of applications and devices. Chapter 5 further explores the integration of IGA into other software by defining a client-server architecture for using the IGA implementation in interactive applications.

The following third part completes this thesis with final conclusions in Chapter 6.

2 BACKGROUND

Parts of this chapter have also been published in the following peer-reviewed article:

> C. Schinko, A. Riffnaller-Schiefer, U. Krispel, E. Eggeling, and T. Ullrich, "State-of-the-art overview on 3D model representations and transformations in the context of computer-aided design," *International Journal on Advances in Software*, vol. 10, p. 446, 2017, ISSN: 1942-2628

This chapter provides an overview of foundations and previous work on which the rest of this thesis is based. First, the foundations to describe geometry, i.e. curves and surfaces, in a CAD context are discussed. Section 2.1 introduces several representations of curves and Section 2.2 extends these concepts to surfaces. Afterwards, required concepts of engineering analysis are presented. Section 2.3 gives some background on the Kirchhoff-Love theory of thin shells and Section 2.4 discusses the discretization of the resulting partial differential equations with subdivision surfaces for isogeometric analysis.

2.1 CURVES

In the 1960s, companies designing airplanes and cars first saw a need to design and process exact curves on computers. However, to work with curves in a computational context, a representation of the continuous nature of a curve as a set of discrete values, usable by a computer, is required. This marked the beginning of the field *Computer Aided Geometric Design* (CAGD). Since then, many representations of curves have been developed in the context of CAGD. This section summarizes the most important concepts for defining curves as can be found in text books like e.g. [38] or [79], on which this section is based.

2.1.1 *Mathematical Representations*

Mathematically, curves may be defined either in an explicit, implicit or parametric form.

The explicit definition of a two-dimensional curve can be written as

$$y = f(x)\,. \tag{2.1}$$

This explicitly defines the y location of a curve based on the x coordinate. For example, a line can be defined with $y = mx + b$. Another example is a parabolic curve, defined by $y = ax^2 + bx + c$. One drawback of this representation is that it does not allow the definition of a vertical line.

The implicit form is written as an equation

$$f(x, y) = 0\,. \tag{2.2}$$

Here, the curve is defined implicitly by all points satisfying the equation. A straight line can be implicitly defined by $ax + by + c = 0$. Another common example is the implicit definition of a circle with radius r around the origin, defined by $x^2 + y^2 - r^2 = 0$. A disadvantage of the implicit form is that it is hard to evaluate a specific point on the curve, as the points on the curve are generally not known.

A drawback of both the explicit and implicit form is that they depend on a particular coordinate system in which the curve is defined. This makes them impractical for design, where complex shapes are often created by combining multiple curves, or segments thereof, each with different transformations applied to it.

Therefore, the most common type of curves in CAGD are parametric curves. A parametric definition of a curve depends on a curve parameter t. This parameter is independent of a particular coordinate system and therefore provides more flexibility for defining the curve. A parametric two-dimensional curve can therefore be defined as

$$\begin{aligned} x &= f(t) \quad \text{and} \\ y &= g(t) \end{aligned} \tag{2.3}$$

using two functions f and g to define the x and y coordinates, respectively, based on parameter t. Using the parametric form, a vertical line may be defined for example with $f(t) = x_0$ and $g(t) = t$ for $t \in [-\infty, \infty]$. A parametric circle with radius r can be defined with $f(t) = r\cos(t)$ and $g(t) = r\sin(t)$ for $t \in [0, 2\pi]$. Using a parametric definition, curve segments can be defined by limiting the evaluated range of parameter t.

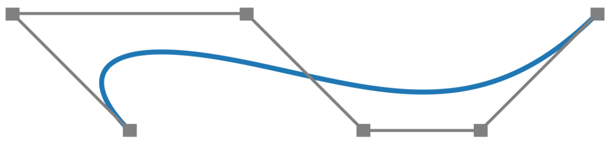


Figure 2.1: Bézier curve of degree 5 and its control polygon.

2.1.2 *Bézier curves*

One of the first, and still one of the most important, representations of curves for CAGD was developed in the 1960s independently by P. de Casteljau at Citroën and P. Bézier at Renault. The work of de Casteljau was initially not published which is why today these curves are known as *Bézier curves*.

The idea of Bézier curves is to represent a parametric curve by a sequence of control points, called the control polygon. The control points can be easily manipulated by designers to change the shape of the curve. Figure 2.1 shows an example of such a curve together with the control points defining its control polygon.

The actual curve is defined as a linear combination of these control points weighted by a set of basis functions evaluated at the curve parameter $t \in [0, 1]$:

$$c(t) = \sum_{i=0}^{n} B_i^n(t) p_i \,. \tag{2.4}$$

For Bézier curves, the basis functions B_i^n are the *Bernstein polynomials* of degree n. Therefore, a Bézier curve of degree n requires a control polygon with $n+1$ control points p_i. The Bernstein polynomials for $t \in [0, 1]$ are explicitly defined by

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i} \,. \tag{2.5}$$

The resulting basis functions are shown in Figure 2.2 for a cubic (degree 3) Bézier curve. These basis functions lead to several noteworthy properties of Bézier curves. First, the Bernstein polynomials are non-negative within the parameter domain on which they are defined, in this case the interval $[0, 1]$. Additionally, at each parameter value the sum of all basis functions is always 1.0. This implies that a Bézier

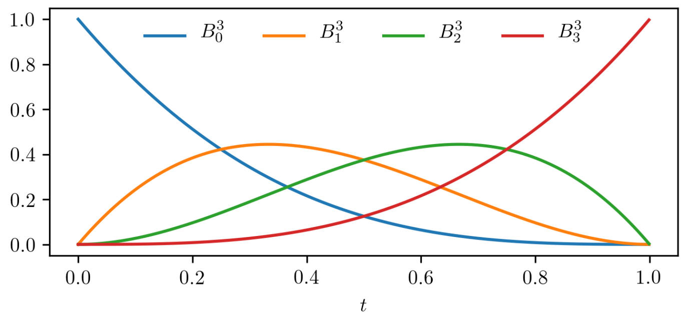


Figure 2.2: Bernstein basis functions of a degree 3 Bézier curve.

curve is always fully contained within the convex hull of its control points. It can also be observed that the first and last basis function have the value 1.0 at the start and at the end of the curve respectively. Therefore, the start and end control points of the control polygon are interpolated by the curve, while all other control points are approximated.

The definition of the Bernstein polynomials can also be written in an recursive form as

$$B_i^n(t) = (1-t)B_i^{n-1}(t) + tB_{i-1}^{n-1}(t) \tag{2.6}$$

with $B_0^0(t) = 1$ and $B_j^n(t) = 0$ for $j \notin \{0, \dots, n\}$.

Similar to the recursive definition of the Bernstein basis functions, also the Bézier curve itself can be evaluated with a recursive algorithm. This is the approach taken by de Casteljau, who computed a point on the curve by linearly interpolating the control points recursively. Given a control polygon with control points p_i and a curve parameter $t \in [0, 1]$, *de Casteljau's algorithm* recursively creates linear combinations b_i^r of these points. Let $b_i^0(t) = p_i$, the recursive definition is given by

$$b_i^r(t) = (1-t)b_i^{r-1}(t) + tb_{i+1}^{r-1}(t) \tag{2.7}$$

for $r = 1, \dots, n$ and $i = 0, \dots, n-r$. Finally, $b_0^n(t)$ defines the point on the Bézier curve at parameter value t.

Figure 2.3 shows a cubic Bézier curve which is evaluated using de Casteljau's algorithm at $t = 0.4$. In addition to the initial control polygon, also the intermediate points computed from de Casteljau's algorithm are shown. The intermediate points $b_i^r(t)$ obtained through

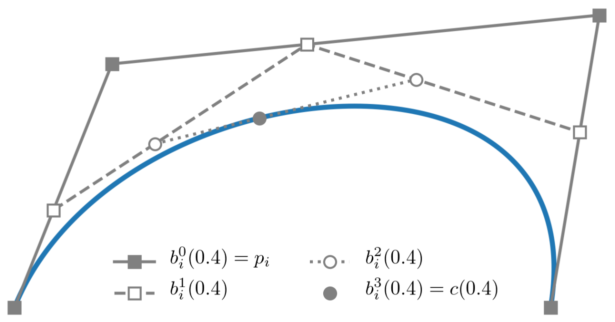


Figure 2.3: Cubic Bézier curve evaluated at parameter value $t = 0.4$ using de Casteljau's algorithm.

linear interpolation can also be represented in a triangular scheme:

$$\begin{array}{llll} b_0^0 \\ b_1^0 & b_0^1 \\ b_2^0 & b_1^1 & b_0^2 \\ b_3^0 & b_2^1 & b_1^2 & b_0^3 . \end{array} \tag{2.8}$$

The leftmost column defines the input control polygon. Each point in the following columns is computed as a linear combination of two points from the column to its left. The final value on the right is the point on the curve. The diagonal values $b_0^0, b_0^1, b_0^2, b_0^3$ and the bottom row $b_3^0, b_2^1, b_1^2, b_0^3$ define two new control polygons for Bézier curves, which split the original curve at parameter t. The curve represented by the diagonal elements is the same as the original curve evaluated on the interval $[0, t]$ and the elements of the bottom row define the control points for a new curve on the original interval $[t, 1]$. This can be used for example to subdivide or clip the curve at a given parameter value.

While Bézier curves enable designers to intuitively define the shape of a curve by manipulating its control polygon, they also have some drawbacks. To represent complex shapes many control points may be needed. However, for Bézier curves the number of control points directly determines the degree of the curve. Therefore, Bézier curves with many control points are of high degree. This limits their usefulness for design as each control point influences every part of the curve, but the strength of its influence reduces with increasing degree. Therefore, designers do not have local control over the curve. Changing one control point changes the shape of the whole curve.

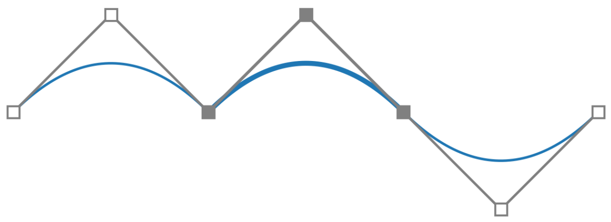


Figure 2.4: A composite Bézier curve of degree 2 consisting of three segments. The second segment is highlighted with bold lines. The placement of control points determines the smoothness of the curve at joins.

One solution to these problems is to model complex curves by combining multiple low-degree Bézier curves into a single *composite Bézier curve*. This keeps the advantage of having a single control polygon, but enables local control of the low-degree piecewise Bézier curve. Each control point only influences one or two of the low-degree Bézier segments, all the other parts of the composite Bézier curve are unaffected. Composite Bézier curves of degree 2 and 3 are commonly used in graphic design and are, for example, part of the *scalable vector graphics* (SVG) standard [33].

To parameterize a composite Bézier curve with L segments, a sequence of knot values $u_0 < \cdots < u_L$ is defined, where each parameter interval $[u_i, u_{u+1}]$ defines a curve segment. This sequence of knot values is called the knot vector. Each global parameter value u between the minimum and maximum knot value defines a point $s(u)$ on curve s. The individual Bézier curve segments s_i can be parameterized by a local parameter t. For $u \in [u_i, u_{i+1}]$ the local parameter t for the Bézier curve defining segment s_i of the curve is

$$t = \frac{u - u_i}{u_{i+1} - u_i}, \tag{2.9}$$

which is always between 0 and 1. Using this parameterization, a point on the curve is defined as $s(u) = s_i(t)$.

Figure 2.4 shows a composite Bézier curve which consists of three segments of degree two. The second segment in the center of the curve is highlighted with bold lines. A single control polygon is used to define a complex curve using multiple low degree segments. The knot vector defining the parameterization of the curve has the form

$[u_0, u_1, u_2, u_3]$ to define three intervals for the respective curve segments.

However, as shown, the resulting curve is not necessarily smooth. While the segments are always connected at their end points, providing C^0 continuity, higher continuities depend on the parameterization of the curve, i.e. the knot vector, and the placement of the control points in the control polygon. To get at least C^1 tangent continuity between two segments the first derivatives of the curve segments need to be equal at their common point.

The derivative of a Bézier curve can be computed based on the derivative of its basis functions, the Bernstein polynomials. Following [38], the derivative of a Bernstein polynomial B_i^n is

$$\begin{aligned} \frac{d}{dt}B_i^n(t) &= \frac{d}{dt}\binom{n}{i}t^i(1-t)^{n-i} \\ &= n\left(B_{i-1}^{n-1}(t) - B_i^{n-1}(t)\right) . \end{aligned} \tag{2.10}$$

Inserting this definition into Equation 2.4 results in the first derivative of a Bézier curve c^n of degree n:

$$\frac{d}{dt}c^n(t) = n\sum_{i=0}^{n}\left(B_{i-1}^{n-1}(t) - B_i^{n-1}(t)\right)p_i\,. \tag{2.11}$$

Transforming indices, this can also be written as

$$\frac{d}{dt}c^n(t) = n\sum_{i=0}^{n-1}(p_{i+1} - p_i)B_i^{n-1}(t)\,. \tag{2.12}$$

The derivative at the start and end of a curve, important for the continuity of composite curves, can be computed by evaluating Equation 2.12 with $t = 0$ and $t = 1$ respectively. In both cases, only one of the basis functions has support, with a value of 1.0, and all others are 0.0. Therefore, for these cases, Equation 2.12 simplifies to:

$$\begin{aligned} \frac{d}{dt}c^n(0) &= n(p_1 - p_0)B_0^{n-1}(0) \\ &= n(p_1 - p_0) \end{aligned} \tag{2.13}$$

and

$$\begin{aligned} \frac{d}{dt}c^n(1) &= n(p_n - p_{n-1})B_{n-1}^{n-1}(1) \\ &= n(p_n - p_{n-1})\,. \end{aligned} \tag{2.14}$$

Therefore, the first derivative and the tangent vector at the start and at the end of a Bézier curve depend only on the degree of the curve and two control points. The tangent at the start of the curve depends

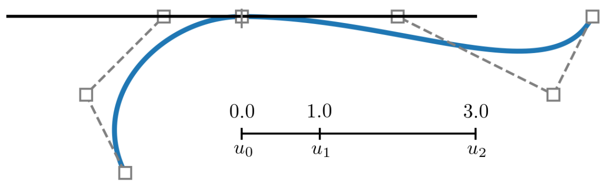


Figure 2.5: Tangent vectors, one of them inverted for clarity, at the join of two segments of a smooth composite Bézier curve of degree 3. The length of the tangents not only depends on the control points but also on the parameterization, i.e. the relative distances in the knot vector shown at the bottom.

on the first two control points p_0 and p_1 in the control polygon. The tangent at the end of the curve is influenced only by the last two control points p_{n-1} and p_n in the control polygon.

However, in a composite Bézier curve, each curve segments can have a different parameterization. Therefore, for a composite curve, the derivatives must take the parameter mapping from Equation 2.9 into account. For some global parameter $u \in [u_i, u_{i+1}]$ within the curve segment s_i, the derivative of a composite Bézier curve s at u then becomes

$$\begin{aligned} \frac{ds(u)}{du} &= \frac{ds_i(t)}{dt}\frac{dt}{du} \\ &= \frac{1}{u_{i+1}-u_i}\frac{ds_i(t)}{dt} \end{aligned} \tag{2.15}$$

where $\frac{ds_i(t)}{dt}$ is the derivative of the curve segment s_i with respect to its local parameter t, as defined in Equation 2.12.

Figure 2.5 shows a composite Bézier curve of degree 3 with two segments. Additionally, the tangent vectors for the two curves at the common point are visualized. One of the tangents is inverted to point in the opposite direction to make it visible in the visualization. For a composite Bézier curve the first derivatives, defining the tangent vectors, do not only depend on the placement of control points, but also on the global parameterization of the composite curve as defined in Equation 2.15. In this case, the control points and the parameterization are chosen to achieve C^1 continuity. To get C^1 continuity, the control points around the join must be colinear and satisfy the same relation as the knot intervals. As the knot intervals $u_1 - u_0$ and $u_2 - u_1$ in

Figure 2.5 have a ratio of 1 : 2, the magnitude of the vectors $p_i - p_{i-1}$ and $p_{i+1} - p_i$ around the common control point p_i must satisfy the same ratio of 1 : 2 for the first derivatives to be equal. Generally, continuity higher than C^0 is not guaranteed and must be manually ensured by the designer by placing control points and choosing a correct parameterization.

2.1.3 *B-Splines*

To solve the continuity problems of composite Bézier curves but still allow the design of complex shapes using low degree curves and a single control polygon, spline curves have been developed. In the context of CAGD, the most common type of spline curves are B-splines.

B-splines, short for *basis spline*, are named after wooden splines used to design and draw curves before the days of CAGD. Curves drawn with wooden splines traced the shape of the splines as they bent between weights, called *ducks*, which held them in place. Similar to how the bending moment varies continuously across a wooden spline, following the derivation by Saxena and Sahay [79], a spline curve of degree n is a curve which is C^{n-1} continuous within the domain it is defined in.

The idea of spline curves is to find basis functions with similar properties as the Bernstein polynomials, i.e. they should be non-negative and sum up to 1.0, but be restricted to a limited part of the curve. Therefore, unlike Bernstein polynomials, which are defined globally on the domain of the curve, spline basis functions should be defined locally within some parameter interval $[u_i, u_j]$ and should be zero outside of this interval.

Similar to composite Bézier curves from Section 2.1.2, a sequence of non-decreasing knot values u_i, called the knot vector, is required to define the parameterization of the basis functions for B-splines.

For B-splines, the basis functions N_i^n of degree n can be defined recursively from lower degree basis functions. The base case for $n = 0$ is

$$N_i^0(u) = \begin{cases} 1 & \text{if } u \in [u_{i-1}, u_i) \\ 0 & \text{otherwise}\,. \end{cases} \tag{2.16}$$

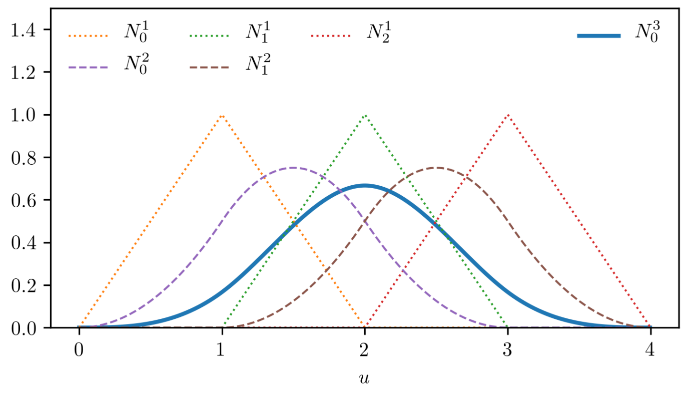


Figure 2.6: A uniform B-spline basis function of degree 3 and all lower degree B-spline basis functions required to recursively evaluate it.

Higher degree basis functions are defined by the recursion

$$N_i^n(u) = \frac{u - u_{i-1}}{u_{i+n-1} - u_{i-1}} N_i^{n-1}(u) + \frac{u_{i+n} - u}{u_{i+n} - u_i} N_{i+1}^{n-1}(u)\,. \quad (2.17)$$

Each B-spline basis is a linear combination of two lower degree B-spline basis functions. This formula was independently developed by Cox [31] and de Boor [15] and is therefore known as the Cox-de Boor recursion formula.

For example, to evaluate a single cubic B-spline basis N_0^3, two quadratic basis functions N_0^2 and N_1^2 are required, which in turn depend on three linear B-spline basis functions N_0^1, N_1^1 and N_2^1 as visualized in Figure 2.6. The linear B-spline basis functions just interpolate between 0 and 1 over the range of a knot interval, as defined by Equation 2.16.

A B-spline curve $s(u)$ of degree n is defined over $[u_{n-1}, u_{L+n-1}]$ where L is the number of curve segments defined by the knot vector. With a control polygon of $L+n$ control points p_i, the curve can now be defined in terms of its basis functions N_i, similar to Equation 2.4 for Bézier curves:

$$s(u) = \sum_{i=0}^{L+n-1} N_i^n(u) p_i\,. \quad (2.18)$$

To evaluate the basis functions N_i required for Equation 2.18, a B-spline curve of degree n with $L+n$ control points in its control polygon requires a knot vector with $L+2n-1$ knot values.

It should be noted that for a naive implementation of Equation 2.17 two additional knot values u_{-1} and u_{L+2n-1} would be required.

However, they do not influence the basis functions within the interval $[u_{n-1}, u_{L+n-1}]$, where the curve is defined, so they can be chosen arbitrarily. For example, a simple strategy is to just duplicate the first and last knot values. In some B-spline literature and implementations these additional knots are explicitly specified, leading to a knot vector with $L + 2n + 1$ values for a B-spline curve of degree n with $L + n$ control points. One notable example where this notation is used is the *IGES* [49] file format commonly used to exchange CAD data. In the remainder of this thesis, these additional knots are not specified explicitly, as they do not influence the resulting curve.

If all knot values are equally spaced, the knot vector is called *uniform* and all basis functions are just shifted copies of each other. In Figure 2.6 this can be observed for the linear and quadratic basis functions. A knot vector may also be *non-uniform*, having unevenly spaced knot values. If a knot vector contains the same value u_i more than once, u_i is called a *multiple knot* and its multiplicity is defined as the number of times u_i is contained in the knot vector. At this point, it is worth noting that there are alternative definitions of uniform knot vectors which also allow multiple knots, as long as all unique knot values are equally spaced, independent of their multiplicity. This thesis uses the definition given above, where all knot values must be equally spaced for a knot vector to be uniform.

Increasing the multiplicity of a knot has the effect of decreasing the continuity of the resulting B-spline curve at that parameter value. For example, while a B-spline curve of degree n without multiple knots is C^{n-1} continuous everywhere, a double knot u_i reduces the continuity to C^{n-2} at parameter value u_i. If the multiplicity of a knot u_i is greater or equal than the degree of the curve, the continuity reduces to C^0. In this case, only the basis function N_i^n will be non-zero at parameter value u_i, as the surrounding knot intervals vanish, and therefore $N_i^n(u_i) = 1.0$. As a result, the defined B-spline curve will interpolate control point p_i at parameter value u_i. An example curve with non-uniform knot vector, which interpolates one of its control points, is shown in Figure 2.7, together with the corresponding basis functions.

By using a non-uniform knot vector with multiple knots, B-spline curves can therefore represent the interpolating end points of Bézier curves. Further, a B-spline curve of degree n with $n + 1$ control points p_i and a knot vector where each knot has a multiplicity of n is equivalent to the Bézier curve defined by the control points p_i. Similarly, a B-spline curve with more than $n + 1$ control points where all knots have multiplicity n is equivalent to the composite Bézier curve de-

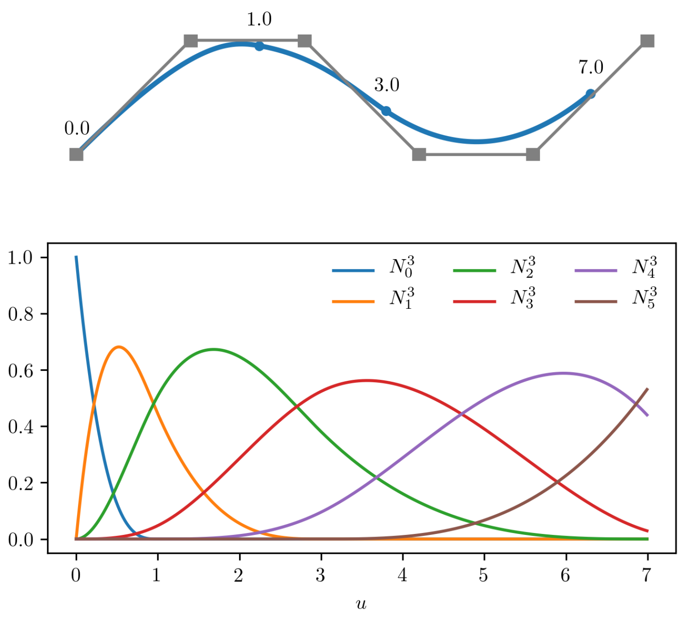


Figure 2.7: A non-uniform knot vector provides more control over a B-spline curve. The degree 3 curve shown on top is defined with the knot vector $[0, 0, 0, 1, 3, 7, 8, 9]$. Below, the corresponding basis functions are shown. The first knot value $u_0 = 0$ has a multiplicity equal to the degree. Therefore, $N_0^3(0) = 1.0$ and the curve interpolates the corresponding control point p_0. Different intervals between knot values change the influence of control points on the curve, as indicated by the basis functions.

fined by the control points. Therefore, B-spline curves are a superset of Bézier curves and composite Bézier curves.

One limitation of B-spline curves, as defined above, is, that they cannot represent conic sections exactly. But conic sections like circles or circle segments are often required in geometric design. To overcome this limitation, a rational representation of B-spline curves can be defined. A rational B-spline curve with control points p_i in three-dimensional Euclidean space $\mathbb{E}^3$ is the projection of a B-spline curve in $\mathbb{E}^4$ to the hyperplane $w = 1$. The coordinates $[x, y, z, w]^T$ of the four-dimensional control points are defined as $[w_i p_i, w_i]^T$, where w_i are the *weights* defined for each control point. A rational B-spline curve of degree n with $L + n$ control points $p_i \in \mathbb{E}^3$ is therefore defined as

$$s(u) = \frac{\sum_{i=0}^{L+n-1} w_i p_i N_i^n(u)}{\sum_{i=0}^{L+n-1} w_i N_i^n(u)}. \qquad (2.19)$$

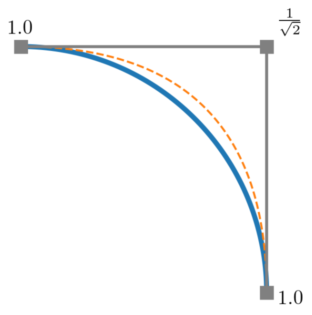


Figure 2.8: An exact circle segment can only be defined using rational B-splines, where control points have different weights, indicated by numbers next to them. The blue curve shows the exact circle segment defined by the rational quadratic B-spline curve, while the orange dashed curve visualizes the non-rational curve defined by the same control polygon without weights.

If all weights are equal, this definition simplifies to the non-rational case from Equation 2.18.

Using Equation 2.19, a quarter circle segment can be defined for example by a quadratic rational B-spline curve with interpolating end points and a center vertex with weight $1/\sqrt{2}$, as shown in Figure 2.8.

However, while all circle segments, including the full circle with separate start and end points, can be represented exactly by rational B-spline curves, there is no periodical definition of a closed circle for B-splines.

2.1.4 *Chaikin's Algorithm*

A different approach to define smooth curves from a coarse control polygon was developed by Chaikin [23]. He presented a geometric approach to refine a control polygon consisting of four points, which converges to a smooth curve in the limit. A variant of his stack-based algorithm is shown in Listing 2.1.

This algorithm creates a smooth curve based on repeated refinement of the control polygon. The control polygon is refined until two neighboring points on the curve are closer than a defined `epsilon`. The curve segment between these points is then approximated by a line.

Listing 2.1: Chaikin's algorithm to draw smooth curves.

```
def chaikin(p1, p2, p3, p4):
    while True:
        stack.push((p3, p4))
        p4 = (p2 + p3) / 2
        while norm(p1 - p4) <= epsilon:
            draw(p1, p4)
            p1 = p4
            if len(stack) == 0:
                return
            p2, p4 = stack.pop()
        p3 = (p2 + p4) / 2
        p2 = (p2 + p1) / 2
```

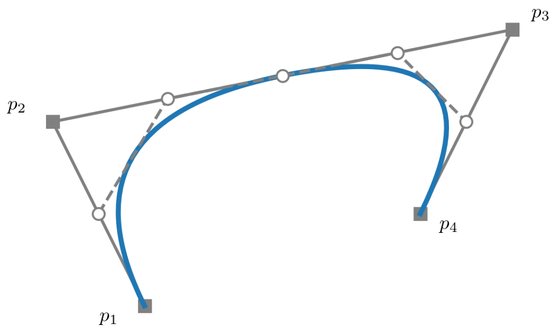


Figure 2.9: Chaikin's algorithm subdivides a control polygon of four points repeatedly into two smaller control polygons until it arrives at a smooth curve. The initial control points have square markers, while the new control points for the first refinement, leading to two subdivided control polygons meeting at the midpoint between p_1 and p_2, are indicated by circles.

Figure 2.9 shows a curve generated by this algorithm, and the intermediate control polygons of the first refinement. As the algorithm geometrically cuts off the corners of the control polygon, it is also known as *Chaikin's corner cutting algorithm*.

Chaikin also extended this algorithm to control polygons with more than four points, e.g. by pre-pushing the remaining control points on the stack in reverse order. However, the generated curve segments only join smoothly if the second control point of the new segment lies on the line defined by the last two control points of the previous segment.

Later, Riesenfeld [70] generalized this idea to arbitrary open or closed control polygons, always creating a smooth curve. Given a control

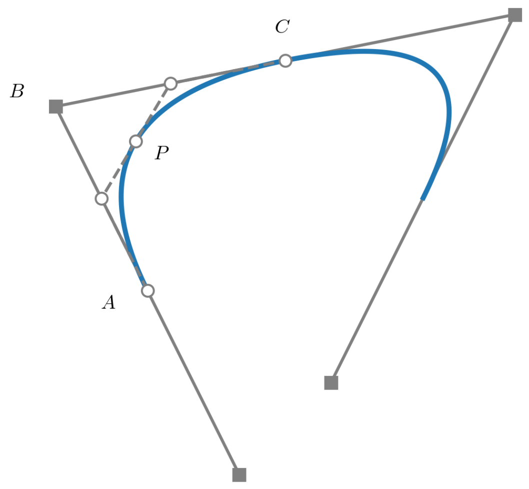


Figure 2.10: The generalization of Chaikin's algorithm by Riesenfeld works for arbitrary control polygons. From each control point B and the adjacent edge midpoints A and C a point P on the curve can be computed. The computation can be continued recursively on the generated subdivided control polygon to evaluate more points on the curve.

polygon with control points p_i, let B be some control point p_i and A and C be the midpoints of the two lines $p_{i-1}p_i$ and p_ip_{i+1} meeting at p_i. From these points, a point P on the limit curve is defined as

$$P = (A + 2B + C)/4 \tag{2.20}$$

Point P lies on the line connecting $(A+B)/2$ and $(B+C)/2$. These two points are then used as new control points for a subdivided control polygon for which A, B and C are defined as

$$\begin{aligned} A &= A \\ B &= (A+B)/2 \\ C &= P \end{aligned}$$

and

$$\begin{aligned} A &= P \\ B &= (B+C)/2 \\ C &= C \end{aligned}$$

allowing the recursive evaluation of more points P on the limit curve.

Figure 2.10 shows the application of this algorithm to create the same smooth curve as in Figure 2.9. Note that the approach of Riesenfeld uses a different control polygon to arrive at the same curve as Chaikin's original algorithm. The first and last control point of

Chaikin's algorithm are the midpoints of the first and last edge in the control polygon for the generalization by Riesenfeld. Therefore, point p_1 from Figure 2.9 corresponds to point A in Figure 2.10.

In [70], Riesenfeld also proved that the curves generated by this approach, and by Chaikin's original algorithm, are in fact quadratic B-spline curves. This demonstrated that B-splines can also be constructed by a geometric refinement operation and later led to the development of subdivision surfaces, as will be discussed in Section 2.2.3.

2.2 SURFACE REPRESENTATIONS

The curve representations discussed in the previous section provide the foundation for several representations of surfaces used in CAGD. Especially, B-splines are the basis for almost all surface representations commonly used in design.

While there are many other techniques to represent surfaces, like polygonal meshes or implicit surfaces, this section focuses only on surface representations commonly used for high quality surface design, such as B-splines/NURBS, T-splines and subdivision surfaces. These surface representations can be found in standard CAGD software which can therefore be readily used to create an integrated design and analysis platform.

There are also more recent developments in surfaces, like e.g PHT-splines [35, 51] or THB-splines [41, 42]. They build on standard surface representations like B-splines or T-splines and add hierarchical features, which are also important for certain isogeometric analysis applications. However, they are currently not available in standard design software and are therefore not widely used in practice. This limits their use for an integrated design and analysis platform, which is why they are not discussed in more detail here.

2.2.1 *B-Splines/NURBS*

As mentioned earlier, one of the foundational representations for curves and surfaces in CAGD are B-splines. Many other curve and surface representations are based either directly on B-spline curves or make use of the B-spline basis functions.

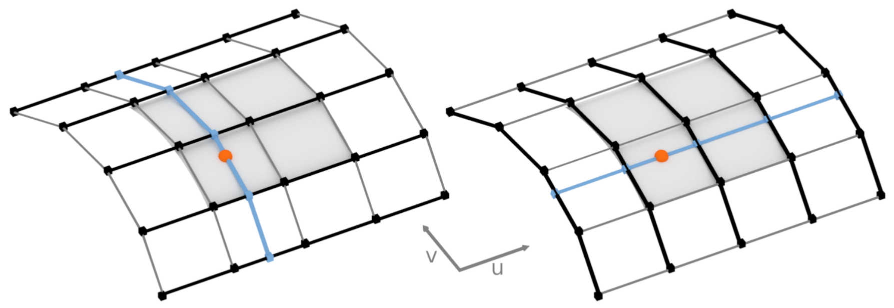


Figure 2.11: Tensor product surfaces can be evaluated by either first evaluating the curves defined by the rows (left, black) or the columns (right, black) of the control grid at a common parameter value u or v respectively. The evaluated points on these curves define the control polygon (blue) of a curve on the surface defined by the control grid. Evaluating this curve at the complementary parameter value v or u yields a point on the surface (orange).

The definition of a B-spline curve from Equation 2.18 can be extended to surfaces using the concept of a tensor product surface. Instead of a sequence of control points, a two-dimensional grid of control points is defined. Each row and column of this grid defines one B-spline curve. To define a parametric surface $x(u, v)$, first the B-spline curves corresponding to each row of control points can be evaluated at the horizontal position at u. This leads to a column of points, one for each row. These points can in turn be used as the control points for a B-spline curve in the vertical direction of the grid. Evaluating this curve at v defines a point on the surface $x(u, v)$. The order of evaluation does not matter, so evaluating the curves corresponding to the columns first, and then evaluating the resulting horizontal curve leads to the same result, as demonstrated in Figure 2.11.

Using the B-spline basis functions, such a tensor product surface is defined as

$$x(u, v) = \sum_j N_j^m(v) \sum_i N_i^n(u) p_{ij} \tag{2.21}$$

for a grid of control points p_{ij}. For each parameter direction u or v a single knot vector is defined, which is used across the surface. In u direction, the resulting surface is of degree n while it is of degree m in v direction. This definition can be rearranged to the more common notation as follows:

$$x(u, v) = \sum_j \sum_i N_i^n(u) N_j^m(v) p_{ij} \,. \tag{2.22}$$

Similar to the B-spline curve case, an extension to a rational surface is required to be able to represent common surfaces based on conic sections, like cylinders or spheres. Again, a weight w_{ij} is specified for each control point p_{ij} and the rational B-spline surface is defined as the projection of a four-dimensional tensor product surface:

$$x(u,v) = \frac{\sum_j \sum_i N_i^n(u) N_j^m(v) p_{ij} w_{ij}}{\sum_j \sum_i N_i^n(u) N_j^m(v) w_{ij}} \,. \tag{2.23}$$

This type of surface is commonly called NURBS, for *Non-uniform rational B-spline*, and is the de facto standard surface representation in the CAD industry.

However, due to the tensor product nature, NURBS can only represent surfaces with a rectangular parameter domain. One approach to overcome this limitation is to join multiple rectangular surfaces, called patches, to form a more complex shape. However, this requires manually aligning control points so that these patches line up and have the required continuity across the join, similar to joining multiple Bézier curves as described in Section 2.1.2.

Another approach, called *trimming*, is to cut out parts of the rectangular domain to define more complex shapes. To do so, trimming curves are defined on the parameter domain to define regions to exclude. During evaluation, parameter values within these regions are not considered as part of the surface. While this approach works reasonably well for rendering the geometry, it poses difficulties for analyzing the surface as comprehensively discussed in [63]. Because trimming of NURBS surfaces is standard practice in CAD, such designs cannot easily be used for analysis purposes without preprocessing the geometry.

2.2.2 *T-Splines*

A surface representation to overcome the strict grid structure of tensor product surfaces are T-splines [86, 87]. T-splines are currently available as a plugin for the CAD software *Rhino* and are the native surface representation of the modeling tool *Autodesk Fusion 360*.

While tensor product surfaces like NURBS require a full grid of control points, T-splines allow T-junctions in the control mesh, which is often referred to as the T-mesh. T-junctions enable rows or columns that start or end at a control point in the interior of the control mesh, not at its boundary. This enables techniques like local refinement, where specific regions of the control mesh can be refined to use more DOF, independent of the rest of the surface, as demonstrated in Figure 2.12.

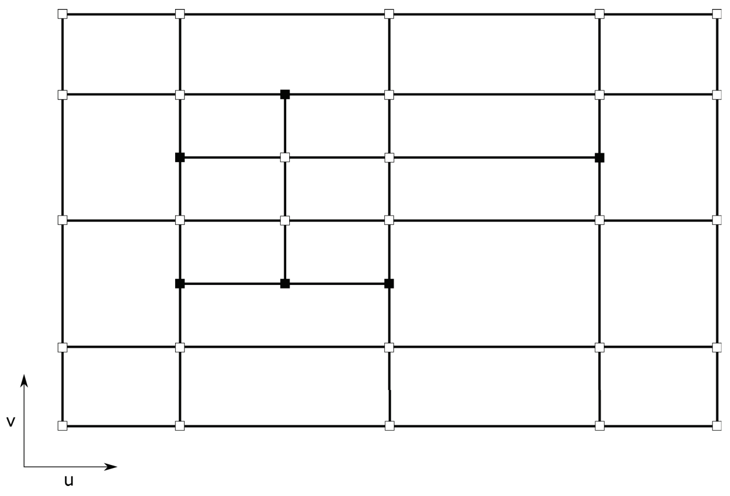


Figure 2.12: The pre-image of a T-mesh in parameter space. T-junctions, highlighted with filled squares, enable local refinement, which reduces the required DOF compared to tensor product surfaces.

To map from the control points of the T-mesh to a smooth surface, T-splines also use the B-spline basis functions. Initially, T-splines were only defined for cubic B-spline basis functions, but later have also been extended to arbitrary degree B-splines [9]. To support surfaces based on conic sections, each control point is assigned a rational weight, similar to NURBS. The main difference to NURBS is that the B-spline basis function associated with each control point p_i in the T-mesh is defined with a *local* knot vector, derived from the neighborhood of the control point in the pre-image of the T-mesh. The definition of T-spline surfaces of degree n defined by c control points is therefore similar to that of NURBS:

$$t(u,v) = \frac{\sum_{i=1}^{c} B_i^n(u,v) p_i w_i}{\sum_{i=1}^{c} B_i^n(u,v) w_i}\,. \tag{2.24}$$

Here, $B_i^n(u,v)$ is the i-th two-dimensional B-spline basis of bi-degree n, defined as the product of two B-spline basis functions: $N_j^n(u)N_k^n(v)$. For a full grid of control points Equation 2.24 is equivalent to Equation 2.23. T-splines are therefore fully compatible to NURBS.

For geometric modeling, constructing a T-spline surface usually begins with a coarse grid of control points which is then locally refined based on the T-spline rules. There are two different sets of rules available to create T-splines. While the initial rules proposed in [86] are simpler, they can lead to many additional unnecessary DOF, as the local refinement process may insert additional control points to create a valid T-spline surface. The improved, but more complex, T-spline rules from [87] offer more flexibility in adding new local control points and therefore reduce the number of additional control points inserted in most cases. Also, to guarantee linear independence of the basis functions for analysis, there is additional work to define a subset of T-splines called *analysis suitable T-splines* (AST) [10, 57].

While T-splines offer the advantages of local refinement to reduce the number of DOF compared to tensor product NURBS surfaces, the originally proposed variant of T-splines still only defines a rectangular parameter domain. To support the design of complex shapes without requiring the designer to manually align patches, as with NURBS, unstructured T-splines have been developed [100, 101]. They support unstructured quad meshes for the initial T-mesh, but require additional control points and T-junctions where the topology deviates from a regular grid to locally define regular patches. These patches define a single gap free T-spline surface of degree 3, but join only with reduced continuity in irregular regions.

T-splines have also been used as the basis for a subdivision scheme called T-NURCC [86]. Subdivision schemes enable designing smooth, arbitrary topology surfaces using a coarse control mesh, and will be discussed in detail in the next section.

2.2.3 *Subdivision Surfaces*

Unlike tensor product surfaces or T-splines, subdivision surfaces are not restricted to a rectangular parameter domain.

The main idea of subdivision surfaces to enable this is to define a smooth surface via geometric refinement of a coarse control mesh. This control mesh may contain vertices with a valence other than regular, so called extraordinary vertices (EVs), which enable the design of arbitrary topology surfaces. For quadrilateral control meshes, regular vertices have a valency of 4, that is these vertices are connected to 4 edges, defining a grid topology. Regular vertices in triangle meshes have a valence of 6. The idea of subdivision surfaces is similar to Chaikin's corner cutting approach for curves, discussed in Section 2.1.4, but extended to surfaces.

As Chaikin's corner cutting approach was found to be equivalent to quadratic B-spline curves, some important generalizations to subdivision surfaces are also based on B-splines in regular regions of the surface, e.g. Doo-Sabin [36, 37] and Catmull-Clark [22] subdivision generalize quadratic and cubic B-splines respectively. However, many different subdivision rules are possible, and today there exist a wide range of different subdivision schemes.

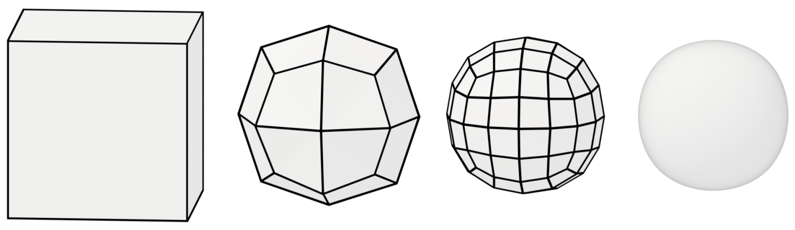


Figure 2.13: An initial cube control mesh (left) is refined using Catmull-Clark subdivision. In the limit, the subdivision process converges to a smooth surface (right).

The most commonly used subdivision scheme today is Catmull-Clark. It is widely used in the entertainment industry and is available in most modeling applications. In recent years, the CAD industry also started adopting Catmull-Clark, e.g. it is used in the CATIA Imagine and Shape software to support the design of freeform surfaces.

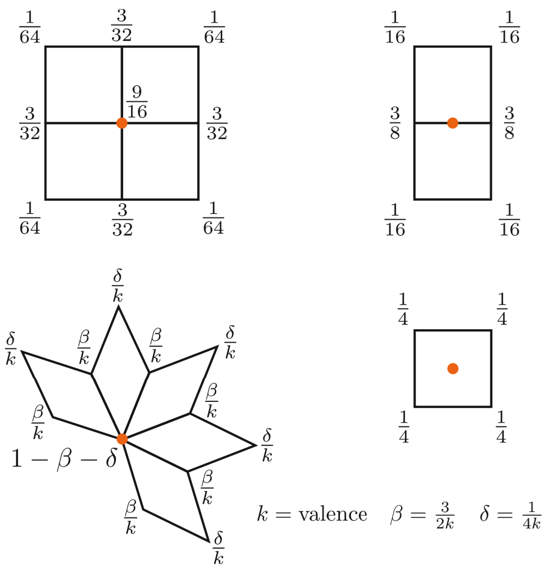
Catmull-Clark generalizes uniform B-splines of degree 3 to arbitrary topology surfaces. The surface is defined by a coarse control mesh, which is repeatedly refined to create a smooth surface in the limit, as demonstrated in Figure 2.13. The refinement operation defines new control points in the middle of all existing edges in the control mesh, a new control point in the center of each face and an updated control point at the position of each existing control point. To compute the positions of these new control points Catmull-Clark uses the stencils shown in Figure 2.14: a vertex stencil, an edge stencil and a face stencil. These stencils define the linear combination of existing control points in a quad mesh to define the new control point positions of the subdivided mesh.

These stencils are only valid for pure quad meshes. To support arbitrary control meshes, Catmull-Clark subdivision is therefore commonly described and implemented using generalized formulas [22]:

- New face points are defined as the average of all existing control points defining a face.
- New edge points are placed at the average position of the edge midpoint with the average of the two face points introduced in the previous step for faces sharing the edge.
- New vertex points are defined by

  $$\frac{Q + 2R + (n-3)S}{n}$$

  where Q is the average of all new face points of faces adjacent to the vertex, R is the average of the midpoints of all existing



$\frac{1}{64}$ $\frac{3}{32}$ $\frac{1}{64}$
$\frac{3}{32}$ $\frac{9}{16}$ $\frac{3}{32}$
$\frac{1}{64}$ $\frac{3}{32}$ $\frac{1}{64}$
$\frac{1}{16}$ $\frac{1}{16}$
$\frac{3}{8}$ $\frac{3}{8}$
$\frac{1}{16}$ $\frac{1}{16}$
$\frac{\delta}{k}$ $\frac{\beta}{k}$
$1-\beta-\delta$
$\frac{1}{4}$ $\frac{1}{4}$ $\frac{1}{4}$ $\frac{1}{4}$
$k =$ valence $\beta = \frac{3}{2k}$ $\delta = \frac{1}{4k}$

Figure 2.14: The linear combinations to define new control points for the Catmull-Clark scheme are described by three stencils. The vertex stencil (left) defines the updated position of existing control points. Two variants are shown: for regular vertices of valence 4 (top, left) and for EVs (bottom, left). The edge stencil (top, right) defines the position of a new control point inserted on each edge. And the face stencil (bottom, right) defines the position of a new control point for each face.

edges connected to the vertex, S is the current position of the vertex and n is the valence of the vertex.

The new face points are then connected to the new edge points to form new quad faces with the updated vertex points. All faces created during subdivision are quadrilaterals. Therefore, after one subdivision step, the number of EVs in the mesh is constant. Also, irregular regions around EVs shrink as only regular regions are created during subdivision.

In addition to the original rules for Catmull-Clark, many extensions have been developed to add additional features like creases and boundaries [13, 34]. These extensions are widely used in the entertainment industry. However, for usage in CAD, more control over the surface and compatibility with NURBS is desired.

Most commonly used subdivision schemes, like Doo-Sabin or Catmull-Clark, only support non-rational surfaces of low degree, i.e. quadratic or cubic, with an uniform parameterization. This limits their use in the context of CAD. For example, due to the missing rational representation, conic sections like cylinders cannot be represented exactly by such surfaces. Also, a non-uniform parameterization is often re-

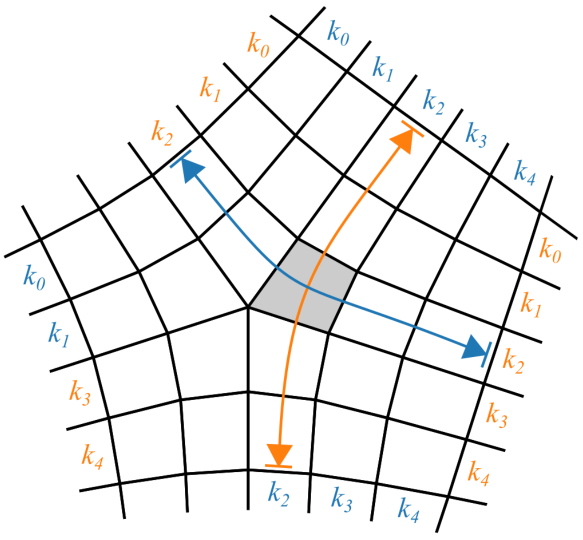


Figure 2.15: The NURBS compatible subdivision scheme uses two local knot vectors to define the parameterization of a face (shaded in gray). The knot vectors are derived from the knot spacings k associated with each edge of the control mesh. In this example, the orange and blue arrows indicate the knot vectors of a face to define a surface of degree 3.

quired, e.g. to define interpolating boundaries, as is commonly done with NURBS surfaces. And higher degree surfaces provide additional advantages like higher continuity for smooth surfaces or improved convergence in the context of analysis.

Ideally, a CAD surface representation provides the precise control of NURBS without the need for trimming and stitching. To achieve this, Cashman et al. [18, 19] extended the Catmull-Clark subdivision scheme to non-uniform, higher degree, and rational surfaces. This subdivision scheme is compatible with odd degree NURBS in regular regions, away from extraordinary vertices with a valence other than four, but generalizes to arbitrary topology control meshes. It therefore combines the advantages of NURBS and subdivision surfaces in a single surface representation.

Similar to rational Bézier surfaces or NURBS, each control point of a NURBS compatible subdivision surface gets an additional weight to provide a rational representation. To allow for a non-uniform parameterization, a knot spacing k is associated with each edge in the odd degree control mesh, defining the interval between two knot values in the knot vector. Knot spacings are required to be equal on opposite edges of a quadrilateral face. Therefore, a knot spacing is always defined for a strip of faces, similar to NURBS. This definition leads to each face of the control mesh having two associated local knot vectors u and v, visualized as two colored arrows in Figure 2.15.

Subdivision is then performed in two stages. During the initial refine stage, edges and faces are split according to the subdivision rules defined in [18], which depend on the valence of control points and the local knot vector. The refinement is followed by multiple smoothing steps, depending on the degree of the surface, in which all control points are moved to their new, updated position. This results in a smooth surface that is equivalent to the corresponding NURBS surface for regular regions of the control mesh and is at least C^1 continuous in all other regions.

NURBS compatible subdivision surfaces therefore maintain the compatibility with NURBS while at the same time they offer the flexibility of subdivision surfaces.

2.2.4 *Comparison of Surface Representations*

The different surface representations presented in the previous sections have different characteristics regarding certain surface properties.

In the following, some of the properties important for design and analysis are discussed and compared.

2.2.4.1 *Arbitrary Topology*

Different surface representations place different constraints on how the control points of a surface can be connected to define a valid surface.

B-spline and NURBS surfaces are based on a tensor product of curves. This requires that all control points are arranged in a regular rectangular grid. This limits the designer to only define rectangular surface patches, which need to be stitched together manually to create more complex shapes. Stiching patches manually is time consuming and error prone, making it hard to guarantee watertight surfaces with a certain continuity for analysis. Another drawback for analysis is that the manually aligned surface geometry needs to be maintained automatically during analysis or the surface may break up under deformation.

The main idea of T-splines is to loosen this strict requirement of a full rectangular grid of control points. With T-splines, the overall structure of the control points is still a grid, but rows and columns can be inserted locally into this grid. However, this still only allows to define rectangular surfaces. To use T-splines for analysis, the restricted AST subset is required to ensure linear independence of the basis functions.

To be able to define more complex shapes, subdivision surfaces can be used. They allow to create a smooth surface from an arbitrary topology control polygon, freeing the designer from splitting a surface into rectangular patches. This also applies to the T-NURCC scheme, generalizing T-splines to arbitrary topology. However, the T-NURCC scheme, being based on Catmull-Clark, is limited to surfaces of degree 3. Similarly, the extension to unstructured T-splines also supports arbitrary topology surfaces, but requires additional control points to do so. Unstructured T-splines are also limited to surfaces of degree 3.

The NURBS compatible subdivision scheme supports arbitrary topology quad meshes as control meshes and can define, in its current implementation, surfaces of any odd degree. This enables designers to create complex shapes with a single surface which can be directly used for analysis.

2.2.4.2 *Continuity*

Continuity is both an aesthetic requirement in design as well as a mathematical requirement for certain types of analysis.

There are two types of continuity, geometric continuity and parametric continuity. They generally describe how a curve looks and behaves, especially where multiple curves or surfaces are joined together.

Parametric continuity C^k always implies geometric continuity G^k of the same order k. However, this does not hold in the opposite direction.

The first case is that the geometry is not continuous at all (G^{-1}). In this case, the two parts of the geometry are disjoint and there is no common parameterization. In other words, there is a gap between different parts of a surface and thus such a surface is generally not suitable for analysis purposes.

Starting at continuity of order 0, the two parts of the geometry are connected, i.e. the end point of the first curve is exactly the start point of the second curve (C^0/G^0). This is also called *positional continuity*.

For G^1 continuity, the curves must also share the same tangent direction at the join point. Similarly, for C^1 continuity, the first derivative of the two curves at the join point must match. Therefore, in addition to the tangent direction, C^1 continuity also requires the magnitudes of the tangents to match. This is referred to as *tangential continuity*.

Continuity of order $k = 2$ is also commonly used in design. For G^2 continuity, both curves must share a common center of curvature at the point where they meet. Again, C^2 continuity is slightly stricter in that it requires that the first and second derivatives at the common point match, providing *curvature continuity*.

Higher orders of parametric continuity C^n require that all derivatives up to the n-th derivative are equal at the common join point.

Analysis methods have different continuity requirements, e.g. the thin shell analysis described in the next section requires at least C^1 continuity. The order of continuity ensured by a surface representation differs between the various types of surfaces.

Within a single NURBS or T-spline surface of degree d, C^{d-1} continuity is automatically ensured (unless the parameterization is changed to lower the continuity, see also Section 2.1.3 and Section 2.2.4.3).

However, due to the restriction to a rectangular parameter domain, multiple separate NURBS patches have to be joined together to model complex shapes. This requires the designer to manually ensure proper continuity at the joined patch boundaries. Further, *trimmed* NURBS surfaces, often used to model complex shapes, may not even provide C^0 continuity [88].

With unstructured T-splines, the patches do not need to be joined manually, but the resulting surface has only a continuity of C^0 and C^1 in irregular regions.

In contrast, with subdivision surfaces (and T-splines using the T-NURCC scheme), a single surface can be used to model geometry with an arbitrary topology. But for subdivision surfaces the achieved continuity depends on the subdivision scheme. Using Catmull-Clark, the surface is C^2 continuous in regular regions, which are equivalent to uniform B-splines of degree 3. For regions near EVs, Peters and

Reif [68] showed that Catmull-Clark surfaces are at least C^1 continuous.

The NURBS compatible subdivision scheme ensures C^{d-1} continuity within the regular regions of a degree d surface, like NURBS, and at least C^1 continuity near EVs, similar to Catmull-Clark, as shown in [19].

2.2.4.3 *Non-Uniform parameterization*

The parameterization defines the mapping of a parameter value u to a point on the curve. For B-spline curves, the parameterization of the curve is defined by its knot vector. The knot vector specifies the parameter intervals over which the basis functions are defined.

B-spline curves with a uniform parameterization have a knot vector where all knot values are equally spaced. Therefore, all basis functions are identical, except that they are shifted across the parametric range.

In contrast, non-uniform B-spline curves allow any monotonic increasing sequence of values as knot vector. The same knot value can even occur multiple times, where the number of times a value is in the knot vector is referred to as the multiplicity of a knot.

The multiplicity of a knot is directly related to the continuity of the curve. If all knot values have multiplicity one, a curve of degree d has C^{d-1} continuity everywhere, as explained in Section 2.2.4.2. If the multiplicity of a knot increases by one, the continuity at that parameter value decreases by one, down to C^0, if the multiplicity is greater or equal than the degree of the curve.

BOUNDARIES One important aspect of setting the multiplicity of a knot equal to the degree of the curve is, that in this case, the curve interpolates the corresponding control point of the control polygon.

This is often used to let the curve start/end exactly at a control point, and is often referred to as *Bézier end conditions* or as having a *clamped knot vector*.

For surfaces, setting the knot multiplicity equal to the degree along the boundary lets the surface boundary interpolate the curve defined by the boundary control points. A clamped knot vector allows designers to intuitively and exactly position the boundary using just the boundary control points.

However, as this requires a non-uniform parameterization, this is not possible using subdivision schemes like Catmull-Clark, which only provide a uniform parameterization.

For Catmull-Clark, special rules have been designed to emulate the interpolation offered by non-uniform B-spline surfaces [13, 34]. Using these rules, the boundary of a Catmull-Clark surface can also be defined with only the boundary control points. However, as these rules result in surfaces which differ from B-splines, it requires special treatment when evaluating positions or derivatives of the surface.

SYMMETRIES Many designs exhibit some form of symmetry which simplifies the design process. For example, often only half of an object is designed, and just mirrored to the other side.

Symmetries are also exploited in analysis, where they help to reduce the number of degrees of freedom, to improve performance.

To use symmetries with any surface representation, it is important to be able to exactly define the surface boundary, as described above. However, to achieve a certain continuity between symmetric parts of a design, also the tangents and maybe even the curvature at the boundary are important.

For B-spline-based surface representations with a non-uniform parameterization the Bézier end conditions provide a convenient way to not only control the surface boundary but also to control the tangents at the boundary.

For Bézier curves or non-uniform B-spline curves with Bézier end conditions, the tangent at the start and end point of a curve are given by the first and last line segment of the control polygon respectively. Therefore, the tangent can be exactly controlled with just the inner control point, next to the boundary control point.

Non-uniform boundary parameterization can be used with NURBS, T-splines or the NURBS compatible subdivision scheme, but not with traditional subdivision schemes like Catmull-Clark.

2.2.4.4 *Higher Degrees*

Often bi-quadratic (degree 2) or bi-cubic (degree 3) surfaces are used for design, but higher degree surfaces are required to achieve higher continuity, which commonly is an aesthetic requirement for high-end design.

For analysis, higher degree surfaces are known to provide better accuracy and faster convergence rates. Therefore, higher degree surfaces are often desirable for analysis, despite the increase in computation time.

B-splines can be generalized to arbitrary degree, so NURBS can be used with any degree. T-splines, being based on B-spline basis functions, have also been generalized to arbitrary degree in [10].

Most commonly used subdivision schemes are restricted to low degree surfaces, i.e. to degree 3 in the case of Catmull-Clark. However, there are subdivision schemes generalizing higher degree surfaces, e.g. by Stam [93] or Zorin and Schröder [107], but they do not provide a non-uniform parameterization or a rational representation. The current implementation of the NURBS compatible subdivision scheme by Cashman [19] generalises to arbitrary odd degree surfaces, allowing to represent all NURBS surfaces (even degree NURBS can be elevated to the next odd degree).

2.2.4.5 *Rational Representations*

Many common shapes in CAGD, like cylinders, circles or other conic sections, cannot be represented exactly by standard B-splines. This leads to aesthetic artifacts as well as to artifacts in analysis results.

In order to exactly define conic sections with B-spline-based surfaces, a rational representation is needed. While subdivision surfaces like Catmull-Clark do not provide a rational representation, NURBS, T-splines and the NURBS compatible subdivision scheme do support it, and therefore can represent conic sections exactly.

2.2.4.6 *Local Refinement*

During surface design it is common to add small surface features to an otherwise smooth surface. This requires adding new control points to have the necessary degrees of freedom to represent the surface feature.

A similar requirement is common during analysis, where more degrees of freedom are used to locally increase the quality of analysis results in regions of interest.

However, for tensor product surfaces like B-splines or NURBS, control points cannot be inserted locally because the control points must form a regular grid. This often leads to many unnecessary degrees of freedom.

With subdivision surfaces supporting arbitrary topology control polygons, the topology can be changed locally to insert the necessary degrees of freedom for the surface feature. However, this will introduce additional extraordinary vertices and can change the original surface in areas surrounding the surface feature. Both consequences might not be desirable in certain cases.

The main advantage of T-splines is support for local insertion of additional control points, therefore local surface changes can be easily made.

2.2.4.7 *Summary*

The most commonly used surface representations in CAGD have different strengths and weaknesses, summarized in Table 2.1. While NURBS are widely used and offer precise control over the surface, they are limited to a rectangular domain. T-splines and subdivision surfaces offer more flexibility via local refinement and arbitrary topology control meshes. However, T-splines and extensions thereof are limited to either a rectangular domain or lower degree surfaces. Similarly, the most commonly used subdivision scheme, Catmull-Clark, is also limited to surfaces of low degree without providing a rational representation.

A promising alternative surface representation, the NURBS compatible subdivision scheme by Cashman *et al.* [18], offers the advantages of both NURBS and subdivision. Its compatibility with NURBS and Catmull-Clark subdivision allows to use many existing designs while taking advantage of additional features.

Table 2.1: Features supported by common surface representations (✓: fully supported, ∼: partially supported, ✗: not supported).

Feature	NURBS	T-Splines	Unstructured T-Splines	T-NURCC	Catmull-Clark	NURBS compatible subdivision
Arbitrary topology	✗	✗	✓	✓	✓	✓
with at least C^1 continuity	✗	✗	∼	✓	✓	✓
Non-uniform parameterization	✓	✓	✓	✓	✗	✓
Higher degrees	✓	✓	✗	✗	✗	✓
Rational representation	✓	✓	✓	✓	✗	✓
Local refinement	✗	✓	✓	✓	∼	∼

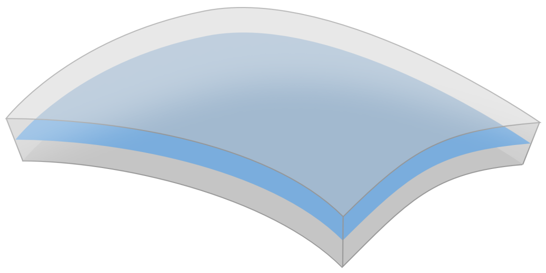


Figure 2.16: Thin shells are structures where the thickness of the material is small compared to the other dimensions. In both CAD and analysis, such structures are usually represented by the middle surface, highlighted in blue, rather than a volumetric representation. This common representation makes thin shells a good target for IGA.

2.3 KIRCHHOFF-LOVE THIN SHELLS

The surface representations from the previous chapter are used to design a wide range of different types of objects. For example, surfaces are often used in CAD to define a boundary representation (B-rep) of a solid object [95]. In this case, the geometry does not define the volume of the object directly, but only the border between solid and non-solid regions.

Another type of object represented by surfaces are thin objects like plates and shells, where the thickness is small compared to the other dimensions of the object. For these types of objects, the surface geometry usually defines the *middle surface* (or midsurface) of the objects volume, shown in Figure 2.16. The middle surface passes through the center of the volume, dividing the thickness into two equal halves [98].

Thin plates, thin flat structures without curvature, are objects where the *slenderness ratio*, the ratio between a common length on the surface a and the plate thickness t, fulfills $10 \leqslant \frac{a}{t} \leqslant 100$. Thin shells, thin curved structures, satisfy $\max(\frac{t}{R}) \leqslant \frac{1}{20}$, for any given radius of curvature R of the middle surface [98]. Such structures are common in many areas of engineering, biology or nature. For example, thin shells are widely used in the automotive and aerospace industries. In everyday life, they appear in the form of e.g. objects made of metal sheets or thin plastic materials. Generally, from tiny blood cells to large structures like concrete domes or the containment of nuclear power plants, thin shells describe a wide range of different objects. They are well suited for isogeometric analysis, because they are represented only by their middle surface, not by a volumetric representation. This middle surface directly corresponds to the surfaces designed in CAD for such thin walled objects, which therefore can be used directly. Also, analysis of thin shell structures is sensitive to geometric errors. Having an exact surface representation for analysis, by taking advantage of IGA, can lead to more accurate results. This thesis therefore focuses on the analysis of *thin shells*, thin structures where one dimension is small compared to its other dimensions, to answer the research questions stated in Section 1.2.

For analysis, thin shell structures are covered by the Kirchhoff-Love shell theory [54, 59]. The Kirchhoff-Love theory assumes that straight lines which are normal to the middle surface in the undeformed configuration

- remain straight during deformation,
- are always normal to the middle surface, and
- keep their initial length.

These assumptions allow to reduce the elasticity behavior of the three-dimensional material volume to that of the two-dimensional middle surface.

In the following, the shell material is always assumed to be homogeneous and isotropic. In other words, the elastic behavior of the shell is the same everywhere throughout its volume and is independent of direction. The used notation and the derivation of the thin shell formulation follows previous work on isogeometric analysis of thin shells based on subdivision [25].

2.3.1 *Geometry*

Assuming a shell with constant thickness t, any material point $\boldsymbol{r}$ of the shell can be defined by a parametric function of three curvilinear coordinates u, v and w as

$$\boldsymbol{r}(u,v,w) = \boldsymbol{x}(u,v) + w\boldsymbol{a}_3(u,v) \tag{2.25}$$

where u and v define a parameterization of the middle surface $\boldsymbol{x}(u,v)$ and w defines the position along the shell director $\boldsymbol{a}_3$ for $-\frac{t}{2} \leqslant w \leqslant \frac{t}{2}$.

Two tangent vectors $\boldsymbol{a}_\alpha$, for $\alpha \in \{1,2\}$, defining a tangent plane on the middle surface, are defined by the partial derivatives of $\boldsymbol{x}(u,v)$:

$$\boldsymbol{a}_\alpha = \boldsymbol{x}_{,\alpha} \tag{2.26}$$

where the comma in the subscript indicates the derivative of the parameter with the given index, i.e. $\boldsymbol{x}_{,1} = \frac{\partial x}{\partial u}$. Due to the Kirchhoff-Love assumptions, the shell director $\boldsymbol{a}_3(u, v)$ always corresponds to the unit normal of the middle surface at $\boldsymbol{x}(u, v)$ and can therefore also be defined by the tangents:

$$\boldsymbol{a}_3 = \frac{\boldsymbol{a}_1 \times \boldsymbol{a}_2}{|\boldsymbol{a}_1 \times \boldsymbol{a}_2|}\,. \tag{2.27}$$

The tangents define the covariant components $a_{\alpha\beta}$ of the surface metric tensor

$$a_{\alpha\beta} = \boldsymbol{a}_\alpha \cdot \boldsymbol{a}_\beta\,. \tag{2.28}$$

The contravariant components $a^{\alpha\beta}$ correspond to the inverse of the covariant metric tensor

$$\left[a^{\alpha\beta}\right] = \left[a_{\alpha\beta}\right]^{-1}\,. \tag{2.29}$$

Using the tangents and normals and Equation 2.25, covariant base vectors for a material point can be defined as

$$\begin{aligned}
\boldsymbol{g}_1 &= \frac{\partial \boldsymbol{r}}{\partial u} = \boldsymbol{a}_1 + w\boldsymbol{a}_{3,1} \\
\boldsymbol{g}_2 &= \frac{\partial \boldsymbol{r}}{\partial v} = \boldsymbol{a}_2 + w\boldsymbol{a}_{3,2} \\
\boldsymbol{g}_3 &= \frac{\partial \boldsymbol{r}}{\partial w} = \boldsymbol{a}_3\,.
\end{aligned} \tag{2.30}$$

Due to the Kirchhoff-Love assumptions, $\boldsymbol{g}_3$ is equal to the normal, as the length of the normals does not change. These covariant base vectors are used to define the components g_{ij}, for $i, j \in \{1, 2, 3\}$, of the shell metric tensor:

$$g_{ij} = \boldsymbol{g}_i \cdot \boldsymbol{g}_j\,. \tag{2.31}$$

The shell metric tensor can be used to measure e.g. lengths, volumes or angles in the volume of the shell material.

2.3.2 *Kinematics*

When the shell material undergoes deformation, a material point $\bar{\boldsymbol{r}}(u, v, 0)$ on the undeformed middle surface $\bar{\boldsymbol{x}}$ will be moved to a new position $\boldsymbol{r}(u, v, 0)$ on the deformed middle surface $\boldsymbol{x}$. Figure 2.17 shows the relationship between the common parameter space and the initial and deformed surface configurations.

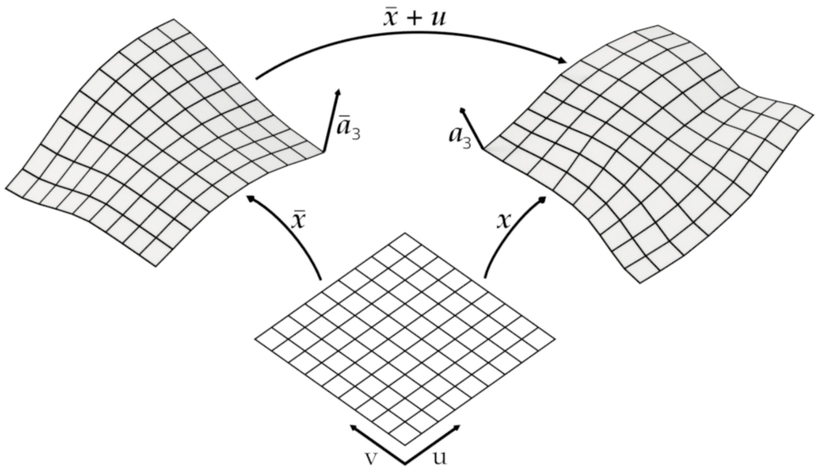


Figure 2.17: A thin shell can be defined by a parametric representation of its middle surface. The physical domains are defined by $\bar{\boldsymbol{x}}$ and $\boldsymbol{x}$, mapping from the 2D parameter space to the initial or deformed surface. The deformed surface $\boldsymbol{x}$ is defined by a displacement field $\boldsymbol{u}$ applied to the initial middle surface $\bar{\boldsymbol{x}}$. In the Kirchhoff-Love theory of shells, the shell director $\bar{\boldsymbol{a}}_3$ or $\boldsymbol{a}_3$ is equivalent to the unit normal of the middle surface.

Any material point of the shell away from the middle surface, i.e. at $\bar{\boldsymbol{x}}(u,v)+w\bar{\boldsymbol{a}}_3(u,v)$ following Equation 2.25, will be moved to $\boldsymbol{x}(u,v)+w\boldsymbol{a}_3(u,v)$. Therefore, the displacement of the middle surface completely defines the deformation of the shell. The deformed middle surface is then defined by a displacement field $\boldsymbol{u}(u,v)$ as

$$\boldsymbol{x}(u,v)=\bar{\boldsymbol{x}}(u,v)+\boldsymbol{u}(u,v)\,. \tag{2.32}$$

This displacement field will be the primary unknown in the analysis of thin shells for the rest of this thesis.

Deformation of the shell causes strain. As a measure of strain the Green-Lagrange strain tensor is used, which can be defined using the metric tensors on the initial and the deformed shell as

$$E_{ij}=\frac{1}{2}(g_{ij}-\bar{g}_{ij})\,. \tag{2.33}$$

Filling in Equation 2.31 and Equation 2.30, the strain components E_{ij} can be split into membrane strains α_{ij}, for the straining of the shell middle surface, and bending strains β_{ij}, describing the change in curvature:

$$E_{ij}=\alpha_{ij}+w\beta_{ij}\,. \tag{2.34}$$

For indices $\alpha,\beta\in\{1,2\}$, the membrane strains, which are independent of the thickness parameter w, are defined as

$$\alpha_{\alpha\beta}=\frac{1}{2}(\boldsymbol{a}_\alpha\cdot\boldsymbol{a}_\beta-\bar{\boldsymbol{a}}_\alpha\cdot\bar{\boldsymbol{a}}_\beta)\,. \tag{2.35}$$

Due to the Kirchhoff-Love assumptions, the shearing $\alpha_{\alpha 3}$ and stretching α_{33} of the normal are zero.

The bending strains contain quadratic terms in w. However, these can be neglected because $w^2 \approx 0$. The non-zero components of the linear approximation of the bending strains are then defined as

$$\beta_{\alpha\beta} = \boldsymbol{a}_\alpha \cdot \boldsymbol{a}_{3,\beta} - \bar{\boldsymbol{a}}_\alpha \cdot \bar{\boldsymbol{a}}_{3,\beta} \tag{2.36}$$

which, using the Kirchhoff-Love assumptions, have been algebraically simplified in [25] to

$$\beta_{\alpha\beta} = \bar{\boldsymbol{a}}_{\alpha,\beta} \cdot \bar{\boldsymbol{a}}_3 - \boldsymbol{a}_{\alpha,\beta} \cdot \boldsymbol{a}_3 \,. \tag{2.37}$$

Using Equation 2.32, the tangents of the deformed middle surface can be defined using $\boldsymbol{u}$ as

$$\boldsymbol{a}_\alpha = \bar{\boldsymbol{a}}_\alpha + \boldsymbol{u}_{,\alpha}$$

which leads to components for the membrane strain also defined in terms of the initial surface and $\boldsymbol{u}$:

$$\begin{aligned}\alpha_{\alpha\beta} &= \frac{1}{2}\left((\bar{\boldsymbol{a}}_\alpha + \boldsymbol{u}_{,\alpha}) \cdot (\bar{\boldsymbol{a}}_\beta + \boldsymbol{u}_{,\beta}) - \bar{\boldsymbol{a}}_\alpha \cdot \bar{\boldsymbol{a}}_\beta\right) \\ &= \frac{1}{2}(\bar{\boldsymbol{a}}_\alpha \cdot \boldsymbol{u}_{,\beta} + \bar{\boldsymbol{a}}_\beta \cdot \boldsymbol{u}_{,\alpha} + \boldsymbol{u}_{,\alpha} \cdot \boldsymbol{u}_{,\beta})\end{aligned} \tag{2.38}$$

In the linear theory of thin shells, which is used here, the non-linear last term of this definition is not considered. The linearized components for the membrane strain are therefore defined as

$$\alpha_{\alpha\beta} = \frac{1}{2}(\bar{\boldsymbol{a}}_\alpha \cdot \boldsymbol{u}_{,\beta} + \bar{\boldsymbol{a}}_\beta \cdot \boldsymbol{u}_{,\alpha}) \,. \tag{2.39}$$

Similarly, linearized components for the bending strain can be defined in terms of the displacement field $\boldsymbol{u}$ as

$$\begin{aligned}\beta_{\alpha\beta} = -\boldsymbol{u}_{,\alpha\beta} \cdot \bar{\boldsymbol{a}}_3 &+ \frac{1}{\sqrt{\bar{a}}}[\boldsymbol{u}_{,1} \cdot (\bar{\boldsymbol{a}}_{\alpha,\beta} \times \bar{\boldsymbol{a}}_2) + \boldsymbol{u}_{,2} \cdot (\bar{\boldsymbol{a}}_1 \times \bar{\boldsymbol{a}}_{\alpha,\beta})] \\ &+ \frac{\bar{\boldsymbol{a}}_3 \cdot \bar{\boldsymbol{a}}_{\alpha,\beta}}{\sqrt{\bar{a}}}[\boldsymbol{u}_{,1} \cdot (\bar{\boldsymbol{a}}_2 \times \bar{\boldsymbol{a}}_3) + \boldsymbol{u}_{,2} \cdot (\bar{\boldsymbol{a}}_3 \times \bar{\boldsymbol{a}}_1)]\end{aligned} \tag{2.40}$$

where $\sqrt{\bar{a}} = |\bar{\boldsymbol{a}}_1 \times \bar{\boldsymbol{a}}_2|$ is the Jacobian determinant of $\bar{\boldsymbol{x}}$.

2.3.3 *Equilibrium*

The deformed configuration of a shell is found as the configuration where internal and external forces are in equilibrium. In general, an exact solution satisfying this requirement at every point cannot be computed. The finite element approach simplifies this requirement so that it needs to be satisfied only for integrals over an area. Following the notation of [52] and applying the principle of virtual work, the sum of the internal and external virtual work done by the respective forces must be zero for an infinitesmale virtual displacement $\delta\boldsymbol{u}$ applied to the surface:

$$\delta W = \delta W_{\mathrm{int}} + \delta W_{\mathrm{ext}} = 0\,. \tag{2.41}$$

The internal virtual work δW_{int} over a domain Ω is defined by stresses and strains as

$$\delta W_{\mathrm{int}} = \int_{\Omega} n^{\alpha\beta}\delta\alpha_{\alpha\beta} + m^{\alpha\beta}\delta\beta_{\alpha\beta}\,d\Omega \tag{2.42}$$

where n and m are the membrane and bending stress tensors respectively. The external virtual work depends on the applied surface forces $\boldsymbol{q}$ per unit area of Ω and the forces $\boldsymbol{N}$ per unit length of the boundary Γ:

$$\delta W_{\mathrm{ext}} = -\int_{\Omega} \boldsymbol{q}\cdot\delta\boldsymbol{u}\,d\Omega - \int_{\Gamma} \boldsymbol{N}\cdot\delta\boldsymbol{u}\,d\Gamma\,. \tag{2.43}$$

The stresses required for Equation 2.42 can be derived from strain measures, as defined in Equation 2.39 and Equation 2.40, using the fourth order material tensor H and two material parameters describing an isotropic material: the Young's modulus E and Poissons's ratio ν. For indices $\alpha, \beta, \gamma, \delta \in \{1, 2\}$, the membrane and bending stress tensors are then given by

$$\begin{aligned} n^{\alpha\beta} &= \frac{Et}{1-\nu^2} H^{\alpha\beta\gamma\delta}\alpha_{\gamma\delta} \\ m^{\alpha\beta} &= \frac{Et^3}{12(1-\nu^2)} H^{\alpha\beta\gamma\delta}\beta_{\gamma\delta}\,. \end{aligned} \tag{2.44}$$

The material tensor H is also defined by the material parameters E and ν as

$$H^{\alpha\beta\gamma\delta} = \nu\bar{a}^{\alpha\beta}\bar{a}^{\gamma\delta} + \frac{1}{2}(1-\nu)(\bar{a}^{\alpha\gamma}\bar{a}^{\beta\delta} + \bar{a}^{\alpha\delta}\bar{a}^{\beta\gamma}) \tag{2.45}$$

where $\bar{a}^{\alpha\beta}$ are the contravariant components of the undeformed surface metric tensor defined in Equation 2.29.

In the next chapter, Equation 2.42 and Equation 2.43 will be used as the basis for the finite element approximation of the equilibrium.

2.4 SUBDIVISION-BASED ISOGEOMETRIC ANALYSIS

Cirak *et al.* [25] were the first to describe finite element analysis based on subdivision surfaces, even before the term *isogeometric analysis* was introduced by Hughes *et al.* [48]. In [25] Loop subdivision was used to describe the undeformed shell geometry as well as the smooth displacement fields of the analysis results.

However, the idea can be generalized to other subdivision schemes as well, as was demonstrated later by multiple implementations for Catmull-Clark subdivision surfaces, e.g. [7, 44, 103], and for Catmull-Clark solids [17].

The Kirchhoff-Love shell theory requires a C^1 continuous geometry description [106], which is hard to guarantee with classical finite elements. In classical FEA, basis functions are local to each element and therefore require careful alignment of the DOF to provide the required continuity. Using subdivision-based IGA, C^1 continuity is guaranteed everywhere. Compared to NURBS, this makes subdivision surfaces better suited for IGA of arbitrary topology surfaces. For NURBS-based IGA, multiple patches are used to represent complex shapes, leading to difficulties in satisfying the continuity requirements. Multi-patch surfaces also require additional treatment to maintain the continuity during analysis [53]. With subdivision, any surface can be modeled as a single, continuous geometry.

To define the subdivision-based finite element discretization, the approach from [25] is used. To simplify definitions and implementation, Voigt's notation is used to map the symmetric second-order strain and stress tensors into vectors:

$$\boldsymbol{n} = \begin{pmatrix} n^{11} \\ n^{22} \\ n^{12} \end{pmatrix} \quad \boldsymbol{m} = \begin{pmatrix} m^{11} \\ m^{22} \\ m^{12} \end{pmatrix} \quad \boldsymbol{\alpha} = \begin{pmatrix} \alpha_{11} \\ \alpha_{22} \\ 2\alpha_{12} \end{pmatrix} \quad \boldsymbol{\beta} = \begin{pmatrix} \beta_{11} \\ \beta_{22} \\ 2\beta_{12} \end{pmatrix}.$$

Similarly, the fourth-order material tensor H is mapped into a symmetric 3×3 matrix, which is defined as

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

Using these definitions, the stress-strain relationships from Equation 2.44 are defined as

$$\boldsymbol{n} = \frac{Et}{1-\nu^2}\boldsymbol{H}\boldsymbol{\alpha} \quad \text{and} \quad \boldsymbol{m} = \frac{Et^3}{12(1-\nu^2)}\boldsymbol{H}\boldsymbol{\beta} \tag{2.46}$$

and can be used to rewrite the definition of the internal virtual work from Equation 2.42 to use Voigt's notation:

$$\delta W_{int} = \int_\Omega \frac{Et}{1-\nu^2}\delta\boldsymbol{\alpha}^T\boldsymbol{H}\boldsymbol{\alpha} + \frac{Et^3}{12(1-\nu^2)}\delta\boldsymbol{\beta}^T\boldsymbol{H}\boldsymbol{\beta}\, d\Omega\,. \tag{2.47}$$

For the finite element approach, the domain Ω is split into m elements defined by nodes I and corresponding shape functions N^I. Each element K defines a distinct subdomain Ω_K of Ω. The finite element formulation $\boldsymbol{u}_h$ of the displacement field $\boldsymbol{u}(u,v)$ can then be expressed with the nodal displacements u_I and the shape functions as

$$\boldsymbol{u}_h(u,v) = \sum_{I=1}^{n} N^I(u,v)\boldsymbol{u}_I \tag{2.48}$$

where n is the number of nodes in the finite element mesh.

Similarly, Equation 2.35 and Equation 2.36 can be reformulated using nodal displacements and shape functions as

$$\begin{aligned} \boldsymbol{\alpha}_h(u,v) &= \sum_{I=1}^{n} \boldsymbol{M}^I(u,v)u_I \\ \boldsymbol{\beta}_h(u,v) &= \sum_{I=1}^{n} \boldsymbol{B}^I(u,v)u_I \end{aligned} \tag{2.49}$$

using matrices $\boldsymbol{M}^I$ and $\boldsymbol{B}^I$ to define membrane and bending strains. The exact definitions of these matrices, as given in [25] based on an orthonormal reference frame with basis vectors $(\boldsymbol{e}_1, \boldsymbol{e}_2, \boldsymbol{e}_3)$, are

$$\boldsymbol{M}^I = \begin{bmatrix} N^I_{,1}\boldsymbol{a}_1 \cdot \boldsymbol{e}_1 & N^I_{,1}\boldsymbol{a}_1 \cdot \boldsymbol{e}_2 & N^I_{,1}\boldsymbol{a}_1 \cdot \boldsymbol{e}_3 \\ N^I_{,2}\boldsymbol{a}_2 \cdot \boldsymbol{e}_1 & N^I_{,2}\boldsymbol{a}_2 \cdot \boldsymbol{e}_2 & N^I_{,2}\boldsymbol{a}_2 \cdot \boldsymbol{e}_3 \\ (N^I_{,2}\boldsymbol{a}_1 + N^I_{,1}\boldsymbol{a}_2) \cdot \boldsymbol{e}_1 & (N^I_{,2}\boldsymbol{a}_1 + N^I_{,1}\boldsymbol{a}_2) \cdot \boldsymbol{e}_2 & (N^I_{,2}\boldsymbol{a}_1 + N^I_{,1}\boldsymbol{a}_2) \cdot \boldsymbol{e}_3 \end{bmatrix}$$

and

$$\boldsymbol{B}^I = \begin{bmatrix} \boldsymbol{B}^I_1 \cdot \boldsymbol{e}_1 & \boldsymbol{B}^I_1 \cdot \boldsymbol{e}_2 & \boldsymbol{B}^I_1 \cdot \boldsymbol{e}_3 \\ \boldsymbol{B}^I_2 \cdot \boldsymbol{e}_1 & \boldsymbol{B}^I_2 \cdot \boldsymbol{e}_2 & \boldsymbol{B}^I_2 \cdot \boldsymbol{e}_3 \\ \boldsymbol{B}^I_3 \cdot \boldsymbol{e}_1 & \boldsymbol{B}^I_3 \cdot \boldsymbol{e}_2 & \boldsymbol{B}^I_3 \cdot \boldsymbol{e}_3 \end{bmatrix}$$

with

$$\begin{aligned} \boldsymbol{B}^I_1 &= -N^I_{,11}\boldsymbol{a}_3 + \frac{1}{\sqrt{a}}[N^I_{,1}\boldsymbol{a}_{1,1} \times \boldsymbol{a}_2 + N^I_{,2}\boldsymbol{a}_1 \times \boldsymbol{a}_{1,1} \\ &\qquad + \boldsymbol{a}_3 \cdot \boldsymbol{a}_{1,1}(N^I_{,1}\boldsymbol{a}_2 \times \boldsymbol{a}_3 + N^I_{,2}\boldsymbol{a}_3 \times \boldsymbol{a}_3)] \\ \boldsymbol{B}^I_2 &= -N^I_{,22}\boldsymbol{a}_3 + \frac{1}{\sqrt{a}}[N^I_{,1}\boldsymbol{a}_{2,2} \times \boldsymbol{a}_2 + N^I_{,2}\boldsymbol{a}_1 \times \boldsymbol{a}_{2,2} \\ &\qquad + \boldsymbol{a}_3 \cdot \boldsymbol{a}_{2,2}(N^I_{,1}\boldsymbol{a}_2 \times \boldsymbol{a}_3 + N^I_{,2}\boldsymbol{a}_3 \times \boldsymbol{a}_3)] \\ \boldsymbol{B}^I_3 &= -N^I_{,12}\boldsymbol{a}_3 + \frac{1}{\sqrt{a}}[N^I_{,1}\boldsymbol{a}_{1,2} \times \boldsymbol{a}_2 + N^I_{,2}\boldsymbol{a}_1 \times \boldsymbol{a}_{1,2} \\ &\qquad + \boldsymbol{a}_3 \cdot \boldsymbol{a}_{1,2}(N^I_{,1}\boldsymbol{a}_2 \times \boldsymbol{a}_3 + N^I_{,2}\boldsymbol{a}_3 \times \boldsymbol{a}_3)] \end{aligned} .$$

Now, applying Equations 2.48 and 2.49 to Equation 2.42 and Equation 2.47 results in the finite element formulation of the internal virtual work based on nodal displacements, the *stiffness matrix* $\boldsymbol{K}_h$:

$$\boldsymbol{K}_h^{IJ} = \sum_{K=1}^{m} \int_{\Omega_K} \left[\frac{Et}{1-\nu^2} (\boldsymbol{M}^I)^T \boldsymbol{H} \boldsymbol{M}^J + \frac{Et^3}{12(1-\nu^2)} (\boldsymbol{B}^I)^T \boldsymbol{H} \boldsymbol{B}^J \right] d\Omega \,. \tag{2.50}$$

Similarly, the finite element formulation of the external virtual work, the force vector $\boldsymbol{f}_h$, is defined based on Equation 2.43:

$$\boldsymbol{f}_h^I = \sum_{K=1}^{m} \left[\int_{\Omega_K} \boldsymbol{q} N^I d\Omega + \int_{\Gamma_K \cap \Gamma} \boldsymbol{N} N^I ds \right] . \tag{2.51}$$

Using these definitions and the equilibrium condition from Equation 2.41 results in the final system of equations

$$\boldsymbol{K}_h \boldsymbol{u}_h = \boldsymbol{f}_h \tag{2.52}$$

for the equilibrium. This enables the computation of the unknown displacement field $\boldsymbol{u}_h$, which corresponds to the vector of nodal displacements u_I here, for given external forces.

This computes a linear approximation to the solution of the partial differential equations describing the physical deformation. Therefore, given enough DOF to represent the solution, this results in highly accurate deformations as long as the overall deformations are small. Unfortunately, what exactly is considered a small deformation cannot be easily checked. The linear simulation assumes that the resulting stress and strain are proportional [90]. This implies that the resulting stress must be below the yield-strength of the simulated material. But to check this, one needs to perform a non-linear simulation, e.g. as described by Cirak and Ortiz [26]. The remainder of this thesis focuses on the linear simulation and its use cases.

To split the subdivision domain into a discrete number of elements, Cirak *et al.* [25] define each face of the subdivision control mesh to be one element in the finite element sense. An element not only depends on the vertices of the face, but also on several neighboring vertices, because the support of the basis functions spreads over multiple faces in the subdivision mesh. For the two most common subdivision schemes, Loop and Catmull-Clark, the vertices of the face and their one ring neighborhood are required. This is different from standard finite elements, where all support nodes are local to the element.

To evaluate positions and derivatives of the surface defined by an element, required for numerically integrating Equation 2.50 and Equation 2.51, two cases need to be considered: regular and irregular elements. Each element will be parameterized by two local coordinates $u, v \in [0, 1]$.

In the regular setting, where all vertices required for the evaluation of an element have a valence of six and four for Loop and Catmull-Clark, respectively, the surface patch can be directly evaluated using its basis functions. Loop subdivision generalizes quartic box splines and Catmull-Clark subdivision is based on uniform cubic B-splines. A regular region of a Catmull-Clark surface can therefore be evaluated as a tensor product B-spline, i.e. as defined in Equation 2.22.

Irregular regions, containing vertices with other than regular valency, cannot be directly evaluated, as the basis functions are different around these extraordinary vertices. A method to evaluate Catmull-Clark and Loop subdivision surfaces near EVs has been presented by Stam [91, 92]. He built on the observation that subdividing the control mesh introduces only regular faces, the irregular regions shrink with each subdivision step. The basic idea is that, after a sufficient number of subdivision steps, all points away from an EV are eventually located within a regular face, and can be evaluated directly. To avoid the time and memory consuming subdivision process, Stam showed how this can be formulated in terms of eigenanalysis of the subdivision matrix, leading to a fast algorithm to evaluate arbitrary parameter locations of a subdivision surface. This technique has been used by Cirak *et al.* [25] for their work on IGA based on Loop subdivision and later by several others for IGA based on Catmull-Clark, e.g. [7, 44, 103].

In both cases, the surface geometry of an element is defined by a number of control points p_I and their associated basis functions N^I as

$$\boldsymbol{x}(u, v) = \sum_I N^I(u, v)\boldsymbol{p}_I \,. \tag{2.53}$$

This parameterization is used to describe both the thin shell middle surface as well as the displacement field $\boldsymbol{u}(u, v)$, as used in Equation 2.32.

The described isogeometric subdivision formulation of the finite element method performed well in numerical benchmarks [25]. However, its application is limited by the subdivision schemes used, as discussed in Section 2.2.4. For example, Loop and Catmull-Clark subdivision cannot accurately represent common CAD geometries like cylinders. Also, advantages of higher degree surfaces for analysis cannot be utilized by these low degree surface representations. Finally, interpolated boundaries require special rules for these subdivision schemes.

Therefore, to fully exploit the potential of subdivision-based IGA, a formulation based on a different subdivision scheme is required which overcomes these short comings.

2.5 SUMMARY

The standard CAD representation NURBS does not support arbitrary topology surfaces, which is a major drawback for the design and analysis of complex objects. During the design, the designer has to manually align multiple NURBS patches to get C^1 continuity. To maintain the required continuity during Kirchhoff-Love thin shell analysis, special techniques, e.g. as presented by Kiendl *et al.* [53], are necessary.

Subdivision surfaces support arbitrary topology surfaces and provide C^1 continuity by design. But state-of-the-art subdivision-based IGA is limited to Loop and Catmull-Clark, both of which do not support boundary interpolation, higher degree surfaces and do not provide a rational representation, which are important properties to ensure a reliable and fast analysis. A comprehensive discussion on this will follow in Section 3.2.

While T-splines do support higher degree surfaces for analysis, e.g. [10, 20, 21], the T-spline-based subdivision scheme T-NURCC, based on Catmull-Clark, and other approaches to support arbitrary topology like [83], also only support surfaces of degree 3. For analysis, only the restricted AST subset can be used, to ensure linear independence of the basis functions.

NURBS compatible subdivision surfaces provide the advantages of both NURBS and subdivision surfaces:

- smooth arbitrary topology surfaces
- rational representation
- support for higher degree surfaces

NURBS compatible subdivision surfaces are well suited for isogeometric analysis of thin shells and are therefore used as the basis for this work, to combine the advantages offered by NURBS and subdivision surfaces for analysis.

In the following, the theory of subdivision based IGA is extended to NURBS compatible subdivision and applications of thin shell IGA for a combined design and analysis approach are explored.

Part II

ISOGEOMETRIC ANALYSIS, DESIGN AND SIMULATION

"[Design is] not just what it looks like and feels like. Design is how it works."

Steve Jobs

3 AN ISOGEOMETRIC ANALYSIS PLATFORM FOR ANALYSIS AND DESIGN

This chapter is based on research that has been published in the following peer-reviewed articles:

A. Riffnaller-Schiefer, U. H. Augsdörfer, and D. W. Fellner, "Isogeometric shell analysis with NURBS compatible subdivision surfaces," *Applied Mathematics and Computation*, vol. 272, Part 1, pp. 139 –147, 2016, Subdivision, Geometric and Algebraic Methods, Isogeometric Analysis and Refinability, ISSN: 0096-3003

A. Riffnaller-Schiefer, U. H. Augsdörfer, and D. W. Fellner, "Physics-based deformation of subdivision surfaces for shared virtual worlds," *Computers & Graphics*, vol. 71, pp. 66 –76, 2018, ISSN: 0097-8493

The previous part described the general idea of isogeometric analysis and the foundations of commonly used geometry representations. This leads to an isogeometric formulation of thin shells based on subdivision surfaces.

The isogeometric thin shell formulation, based on work from Cirak *et al.* [25], can be used with classical subdivision surfaces like Loop or Catmull-Clark to perform various analysis tasks. However, these surface representations miss several features commonly used in CAD, i.e. higher degree surfaces, a rational representation, and non-uniform parameterization. These features also hold advantages for analysis. To take advantage of these common CAD features but also keep the advantages of a subdivision based formulation, a different geometry representation is required.

Chapter 1 also indicated that isogeometric analysis might be useful not only for engineering analysis, but also for the design process itself. But this area of application is still largely unexplored.

The research questions stated in Section 1.2 define several open questions and challenges which address these issues. To answer them, this part of the thesis defines a unified isogeometric analysis platform based on NURBS compatible subdivision surfaces, combining the advantages of NURBS and subdivision surfaces for analysis and design.

This platform enables using IGA for traditional engineering analysis tasks, as well as for new tasks in design and interactive simulation.

To do so, first the state-of-the-art of subdivision-based IGA, described in the previous part, is advanced to NURBS compatible subdivision in Section 3.1. This provides users with the precise control over the surface, as with NURBS, but enables the design of freeform surfaces that are not limited to rectangular patches. The resulting implementation is then benchmarked with various example problems in Section 3.2 to demonstrate its accuracy and advantages over existing techniques. Finally, a number of challenges for using a design directly for analysis and some possible artifacts in the analysis results are discussed in Section 3.3.

Having the necessary foundations, the following chapters build on this subdivision-based IGA platform. Chapter 4 integrates IGA based on subdivision surfaces into existing design tools to enable common engineering analysis tasks as well as to expand the tools available to designers. In Chapter 5, the presented platform for simulating real-world physical deformations is then used to improve the realism of virtual worlds by enhancing them with interactively computed deformations. This is implemented as a service oriented architecture which makes IGA accessible to a wide range of different applications and devices.

3.1 NURBS COMPATIBLE SUBDIVISION FOR ANALYSIS

The NURBS compatible subdivision scheme developed by Cashman *et al.* [18] is a superset of Catmull-Clark subdivision. Therefore, all Catmull-Clark surfaces can also be represented by the NURBS compatible subdivision scheme. However, not all research on IGA using Catmull-Clark is directly applicable to NURBS compatible subdivision: the definition of an element differs, evaluation of irregular regions needs to be extended to higher degrees, and some areas, like boundaries, are represented differently. Other features of the NURBS compatible subdivision scheme, like higher degrees, are not available in Catmull-Clark, but are important for analysis.

This section extends the previous work on subdivision-based IGA, as summarized in Section 2.4, to the NURBS compatible subdivision scheme, providing the first fully NURBS compatible isogeometric analysis method based on subdivision surfaces. First, Section 3.1.1 discusses how an *element*, in terms of the finite element method, is defined using the NURBS compatible subdivision scheme. Then, Section 3.1.2 compares different treatments of surface boundaries. Afterwards, Section 3.1.3 discusses evaluation near extraordinary vertices. Section 3.1.4 covers techniques and improvements for numerical integration of surface quantities and Section 3.1.5 describes how forces and constraints are applied to the resulting system of equations.

3.1.1 *NURBS compatible Subdivision-based Element*

The definition of a subdivision-based element, in terms of FE, as explained in Section 2.4 is not applicable to the NURBS compatible subdivision scheme. Therefore, a different definition equivalent to NURBS-based IGA is chosen.

For uniform subdivision schemes like Loop or Catmull-Clark each face of the subdivision control mesh is considered an element for analysis. This is appropriate because each face maps to some surface area of the resulting subdivision limit surface. But the NURBS compatible subdivision scheme also supports non-uniform knot vectors. Using non-uniform knot vectors with multiple knots, some faces of the control mesh may not correspond to an area of the limit surface at all, i.e. they have zero area in the limit. Therefore, a different definition of an element is required.

For NURBS compatible subdivision surfaces of odd degree, two knot intervals, for the parametric u and v direction, are assigned to each quad in the control mesh, see Section 2.2.3 and Figure 2.15. All faces in the control mesh where both knot intervals are non-zero can be considered an element in terms of FE, as they map to some surface area in the limit. Faces with zero knot intervals are not considered elements, but only provide knot information and support vertices for other elements.

Hence, NURBS compatible subdivision-based elements are defined by non-zero knot spans, like elements for NURBS-based IGA [48].

3.1.2 *Boundaries*

For open uniform B-spline surfaces, and by extension also subdivision surfaces like Catmull-Clark, the boundary of the limit surface does generally not coincide with the boundary of the control polygon. Without any special rules, a Catmull-Clark subdivision surface *shrinks* at the boundary with each subdivision step.

To overcome this, and let the limit surface interpolate the boundary curve defined by the boundary control points, special rules have been developed for subdivision schemes like Loop [47] and Catmull-Clark [34].

However, these rules change the underlying representation of the surface, i.e. boundary elements of a Catmull-Clark surface using these rules cannot be evaluated using B-spline basis functions. While this is generally not a problem for design, it makes evaluation of the surface more complex for analysis.

Therefore, most previous IGA methods based on subdivision, i.e. [7, 25, 103], use an approach proposed by Schweitzer [82] in which the boundary of the surface is modeled either explicitly or implicitly using additional *ghost vertices*. This allows to define the boundary using just regular uniform elements, without special rules. Using these ghost vertices, the tangent/normal of the surface at the boundary can also be controlled, but as noted by Green [44] this approach unnecessarily restricts *all* derivatives at the boundary to zero. This results in undesired flat surface regions when using symmetry for design or to accelerate analysis. Additionally, corners and extraordinary vertices on the boundary also pose some difficulties using the ghost vertices approach. Another issue pointed out by Green [44] is that this approach requires a constraint for each boundary vertex to maintain the position of the *ghost vertex*, even if both translation and rotation of the vertex are unconstrained.

In contrast to traditional subdivision schemes like Catmull-Clark, the NURBS compatible subdivision scheme allows for non-uniform knot vectors, like NURBS. Using non-uniform knots, and in particular multiple knots for Bézier end conditions, the boundary can be modeled directly, without any special rules or additional *ghost vertices*, as explained in Section 2.2.4.3. Also, corners can be defined exactly by using multiple knots in both parametric directions of a boundary element. Further, the surface normal at the boundary can be intuitively controlled by just the boundary vertices and their immediate neighbors, simplifying handling of boundary conditions and symmetry constraints. Using this technique does not restrict higher derivatives like curvature, unlike the *ghost vertices* approach for other subdivision schemes. Also, this approach does not require any constrains for maintaining the boundary configuration.

However, some restrictions also apply to this technique, as the NURBS compatible subdivision scheme does not support multiple knots at extraordinary vertices. Therefore, extraordinary vertices cannot be placed on boundaries defined using multiple knots, which is usually not a problem for design.

Also worth pointing out is that boundaries defined using multiple knots create knot spans with a zero interval. For odd degree surfaces, where knot spans correspond to edges/faces in the control polygon, this leads to faces in the control mesh corresponding to a zero knot interval. However, as the *elements* for IGA are only non-zero knot intervals, these faces do not represent an element for analysis. This is in contrast to IGA based on Catmull-Clark, where all faces of the control mesh are considered an element for analysis. Therefore, using the same control polygon, a NURBS compatible subdivision surface may define fewer elements for analysis than a Catmull-Clark surface, depending on the knot vectors defined.

3.1.3 *Evaluating Irregular Regions*

While regular regions of a NURBS compatible subdivision surface, where all vertices have valence four, can be evaluated as a standard tensor product B-spline surface, irregular regions near extraordinary vertices require special treatment.

In the following, multiple techniques for evaluating the surface and derivatives within irregular regions are discussed.

3.1.3.1 *Patching*

One approach to evaluate any point on a subdivision surface is to convert irregular regions to regular ones. The idea is to, conceptually, remove irregular regions from the subdivision surface and instead add regular surface patches which cover the removed surface area. In the following, a patch is generally defined as a single rectangular B-spline surface, possibly containing several elements in terms of FE.

Different techniques are available to do so, often using surface patches of higher degree or with many degrees of freedom to achieve the required continuity between the patches themselves and the rest of the subdivision surface. But generally, the regular surface patches can only approximate the irregular region of the subdivision surface they replace. An exception to this are flat surface regions, where patches can exactly represent a given surface.

One particular algorithm called *Patching Catmull-Clark Meshes* (PCCM) was presented by Peters [69]. This algorithm enables converting a Catmull-Clark subdivision surface into a set of smoothly joining bicubic NURBS patches, one for each quad in the Catmull-Clark control mesh. The resulting regular NURBS patches can then be used to evaluate the approximated subdivision surface at any point. This is similar to a technique known as *Bézier extraction* in the context of T-spline-based IGA, where a T-spline surface is converted into a set of NURBS/Bézier patches for analysis [84].

For use with NURBS compatible subdivision surfaces of degree 3, the PCCM algorithm has been adapted to support non-uniform knot vectors in the initial control mesh and rational weights for the control points. Both of these features are not supported by Catmull-Clark subdivision surfaces and are therefore not handled by the original PCCM algorithm.

The PCCM algorithm works in two stages:

1. Knot Insertion
2. Corner Smoothing

For the knot insertion step, a submesh is defined for each quad in the Catmull-Clark control mesh. The submesh includes all subquads

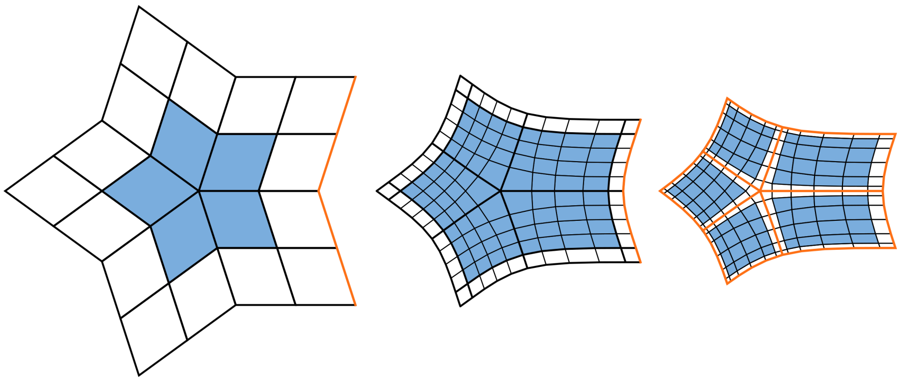


Figure 3.1: A subdivision control mesh with a non-uniform knot vector (left) is converted to five NURBS patches (right). Edges highlighted in orange are setup as Bézier-like interpolating boundaries using multiple knots. Faces marked in blue indicate elements for analysis, i.e. faces that have non-zero surface area in the limit. The resulting NURBS patches are derived from the second subdivision level (center) of the initial control mesh by knot insertion and corner smoothing.

at subdivision level l of the quad and one ring of direct and diagonal neighbor faces around these subquads. For NURBS compatible subdivision surfaces $l > 1$ is required for the conversion, unlike with Catmull-Clark where $l > 0$ is sufficient in some cases. This is due to the support for non-uniform knot vectors and an implementation detail of the NURBS compatible subdivision algorithm, which evens out large knot intervals during the first subdivision step. Also, faces which do not define any surface area in the limit, e.g. due to zero knot intervals, are not converted. Figure 3.1 shows such regions in white whereas faces with non-zero surface areas are colored blue.

The submesh is then interpreted as the control mesh of a degree 3 NURBS patch. Around extraordinary vertices some control points are excluded to define a regular control grid, as required for NURBS. For Catmull-Clark surfaces, Peters [69] assumed uniform knot vectors for these initial NURBS patches. In contrast, for the NURBS compatible subdivision scheme the actual knot intervals defined on the underlying subdivision control mesh are assigned to the NURBS patch, as in the example in Figure 3.1. This way, the existing parameterization of the subdivision surface, e.g. to define boundaries, is directly used for the resulting NURBS patches.

The parameterization of the patch is then changed by knot insertion [14] to have Bézier end conditions. Additional knots are inserted into the knot vector, new control points are added and existing control points are updated accordingly. For Catmull-Clark, Peters [69] always inserts a fixed number of knots to define the desired parameterization.

Using the NURBS compatible subdivision scheme, the multiplicity of some knots may already be greater than one, because the parameterization is taken directly from the subdivision surface. In that case, the multiplicity is only increased as required for interpolating boundaries of the patch.

Due to the Bézier boundaries, the resulting surface patch interpolates the four corner control points exactly. If one of them is an EV, it is repositioned directly onto the subdivision limit surface. For Catmull-Clark, an explicit formula for the limit position is given in [45], for the NURBS compatible subdivision scheme it can be derived by eigen analysis of the subdivision matrix for the given submesh.

Because all patches generated this way interpolate the shared boundary to all neighbor patches, they all join with at least C^0 continuity. In regular regions, away from EVs, C^2 is maintained, as for the subdivision limit surface. To improve the continuity near EVs, the control points of the generated patches are updated in the next step.

In the corner smoothing step, the position of control points near EVs are changed to get C^1 continuity across the edges of all patches. Here, the formulas from Peters [69] can be applied unchanged also for the NURBS compatible subdivision scheme, as they are applied to the previously generated NURBS patches with Bézier boundaries, which are now independent of the subdivision representation.

The resulting collection of NURBS patches joins with C^1 continuity near EVs and is C^2 everywhere else. And because they provide a regular tensor product definition of the whole subdivision surface, these patches can be used to evaluate exact derivatives anywhere on the subdivision surface.

However, near EVs, the subdivision limit surface is only approximated by the extracted patches, unless the surface is flat. Therefore, derivatives can deviate from the real derivatives of the subdivision limit surface.

The approximation is also visible in the form of surface artifacts near EVs. To achieve C^1 across all patches, the corner smoothing of the PCCM algorithm creates undesirable flat areas around EVs in some cases. This change leads to further distortions in the neighborhood of EVs, which is noticeable especially for EVs with even valence greater than 4.

Another drawback for analysis is that one element of the initial subdivision control mesh is converted to a NURBS patch with at least 4×4

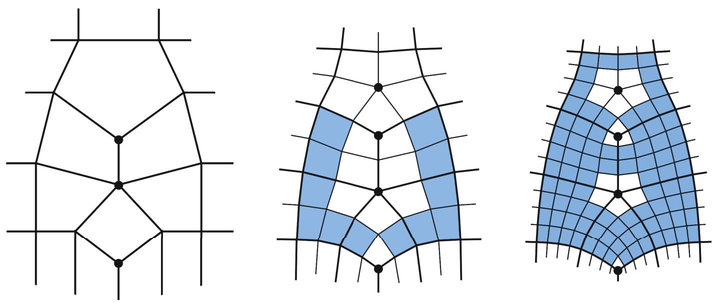


Figure 3.2: Subdividing an initial Catmull-Clark control mesh (left) introduces only regular regions (blue) away from EVs (marked with •). After the first subdivision step (center), the mesh consists only of quad faces and the number of EVs is constant. The second subdivision step (right) and every following step shrink the irregular regions around EVs.

elements, as demonstrated in Figure 3.1. This increases the number of elements by a factor of at least 16. As the number of elements is directly related to computation time, this significantly impacts the time needed for analysis.

While this approach allows to get exact derivatives anywhere on the approximated surface, it has some drawbacks that make it less suitable for analysis. Drawbacks are that the patching increases the number of elements and that near EVs, the resulting patches deviate from an initial curved subdivision surface.

3.1.3.2 *Eigen Evaluation*

A direct method to evaluate subdivision surfaces at arbitrary parameter values was presented by Stam, first for Catmull-Clark [91] and later also for Loop subdivision [92]. His approach builds on the observation that the number of EVs, after the first subdivision step, is constant in a given subdivision surface. With each subdivision step, irregular regions around EVs shrink, while only regular regions are introduced away from EVs. This is illustrated in Figure 3.2 for Catmull-Clark subdivision. Now, the idea is that, after a sufficient number of subdivision steps, all EVs are separated by regular regions and any point away from an EV eventually lies within a regular region. For Catmull-Clark, regular regions of the surface can be evaluated as uniform bi-cubic B-spline patches. Equivalently, for NURBS compatible subdivision, regular regions are non-uniform B-spline patches of any odd degree.

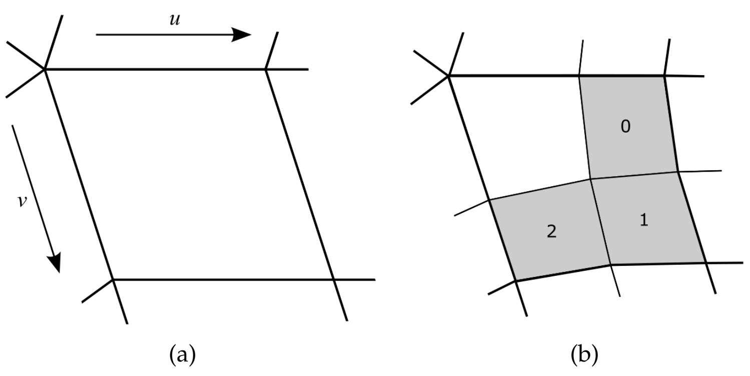


Figure 3.3: For Catmull-Clark, one subdivision step creates three regular sectors (gray) which can be evaluated as bi-cubic B-splines. Repeated subdivision recursively creates more regular sectors in the remaining irregular region.

While subdividing until a point of interest is within a regular region is conceptually simple, it is not practical, as the required subdivision refinement is computationally expensive and may require large amounts of memory for complex meshes. Stam [91] showed how to transform the subdivision operation into its eigenspace, where subdivision is just a scaling operation.

The subdivision refinement operation can be expressed as a matrix multiplication. Let C_0 be k control points surrounding an isolated EV. For evaluation of Catmull-Clark surfaces Stam introduced two matrices: The matrix A is the subdivision matrix that creates k new points from k old control points. Applied to the initial face shown in Figure 3.3a this results in the unlabeled sector next to the EV in Figure 3.3b. The extended subdivision matrix $\bar{A}$ additionally introduces new points defining new regular faces in the region surrounding the EV. For Catmull-Clark, these new faces correspond to the sectors 1, 2 and 3 in Figure 3.3b. After $n = \lfloor \min(-\log_2(u), -\log_2(v)) \rfloor$ subdivision steps, any local parameter location $(u, v) \in [0, 1]^2$ away from the EV is within a regular sector that can be evaluated. Using the two subdivision matrices A and $\bar{A}$, the new points $\bar{C}_n$ for the n-th subdivision level are defined by the relation

$$\bar{C}_n = \bar{A}C_{n-1} = \bar{A}A^{n-1}C_0 \qquad n \geqslant 1\,. \tag{3.1}$$

A subset of points $\bar{C}_n$ are the control points of regular B-spline patches introduced by the subdivision refinement. For Catmull-Clark, sectors that can be evaluated as a B-spline are shaded in gray in Figure 3.3. A picking matrix P_i is used to identify the vertices required for the regular sector i in the refined mesh, which can then be multiplied with

B-spline basis functions $N(u,v)$ to evaluate sector $s_{i,n}$ at arbitrary parameter values (u,v):

$$s_{i,n}(u,v) = C_0^T (P_i \bar{A} A^{n-1})^T N(u,v)\,. \tag{3.2}$$

Stam showed that it is possible to avoid explicit subdivision and rewrite the evaluation from Equation 3.2 using the eigenvalues Λ and eigenvectors V of the subdivision matrix A and control points projected into eigenspace $\hat{C}_0 = V^{-1} C_0$:

$$s_{i,n}(u,v) = \hat{C}_0^T \Lambda^{n-1} (P_i \bar{A} V)^T N(u,v)\,. \tag{3.3}$$

The subdivision matrix only depends on the valence of the EV. Therefore, the rightmost terms can be precomputed for any sector and desired valence into a set of eigenbasis functions x

$$x(u,v,i) = (P_i \bar{A} V)^T N(u,v) \tag{3.4}$$

which, together with the eigenvalues Λ and inverse eigenvectors V^{-1} of A form the so called eigenstructure. This precomputed eigenstructure allows for an efficient evaluation of irregular regions at arbitrary parameter values:

$$s_{i,n}(u,v) = \hat{C}_0^T \Lambda^{n-1} x(u,v,i)\,. \tag{3.5}$$

The general approach described above also applies to the NURBS compatible subdivision scheme, but the details to implement these steps differ from Stam [91]. Here, Stam's evaluation algorithm is extended to support higher degree subdivision surfaces. In particular, a different numbering scheme is used which simplifies collecting the required control vertices for surfaces of different degrees. Also, as the number of sectors that can be evaluated as regular B-splines increases with degree, a consistent method to enumerate these sectors is presented. Because the implementation of the NURBS compatible subdivision scheme is currently limited to odd degrees, only odd degree surfaces are discussed.

Due to the increasing support of higher degree basis functions, at least $(d+1)/2$ subdivision steps are required to isolate all EVs in a subdivision surface of degree d. Because of the knot insertion strategy for NURBS compatible subdivision [18] this also has the effect of creating regions with only uniform knot values around EVs. Therefore, only the case of a single EV surrounded by uniform knot intervals is considered here.

Subdivision matrices A and $\bar{A}$ differ for each valence and for each degree. The indices of the vertices used for the subdivision matrix

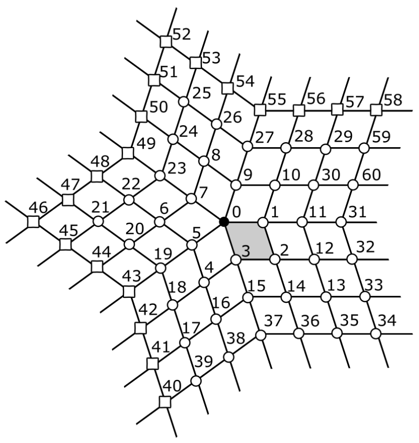


Figure 3.4: To evaluate a point within an irregular face (gray) near an EV with valence 5 of a surface of degree 5, three rings of vertices around the EV, marked with •, are collected and numbered as shown. Support vertices of the surface patch are indicated by ◦. Additional vertices marked by □ are not required for evaluation, but are still collected to simplify the implementation.

are defined by collecting $r = (d+1)/2$ rings of vertices around the EV, starting at the EV. To simplify the collection of vertices for each degree and valence, the indexing scheme includes vertices not strictly required for evaluation. Figure 3.4 shows the indices of three rings of vertices around an EV of valence 5 for a degree 5 subdivision surface.

To extract from all refined vertices those needed for evaluation of the regular patch region i the picking matrix P_i is employed. Each row with index j of P_i contains zeros except for column k, which contains 1. The index k corresponds to the index of the j-th support vertex needed for the regular B-spline patch defining this sector. The size of the picking matrix depends on the valence of the EV and degree of the surface. Its number of rows is equal to the number of vertices in a single B-spline patch of the given degree, i.e. 16 vertices for degree 3, 36 vertices for degree 5. Its number of columns is equal to the number of collected vertices according to the indexing scheme, e.g. 60 for a valence 5 EV on a surface of degree 5 as shown in Figure 3.4.

Assuming the EV is located at the top left corner of the initial quad, sectors are numbered by traversing the subdivided children of that face in clockwise, depth-first order. The top left child of the first subdivision level, next to the EV, is not visited. Figure 3.5 shows sector indices i and which sectors can be evaluated, highlighted in blue, for three subdivision levels. In contrast to Figure 3.3 evaluating Catmull-Clark, no sectors can be evaluated as B-spline surfaces after one subdivision step for the NURBS compatible subdivision scheme. Due to

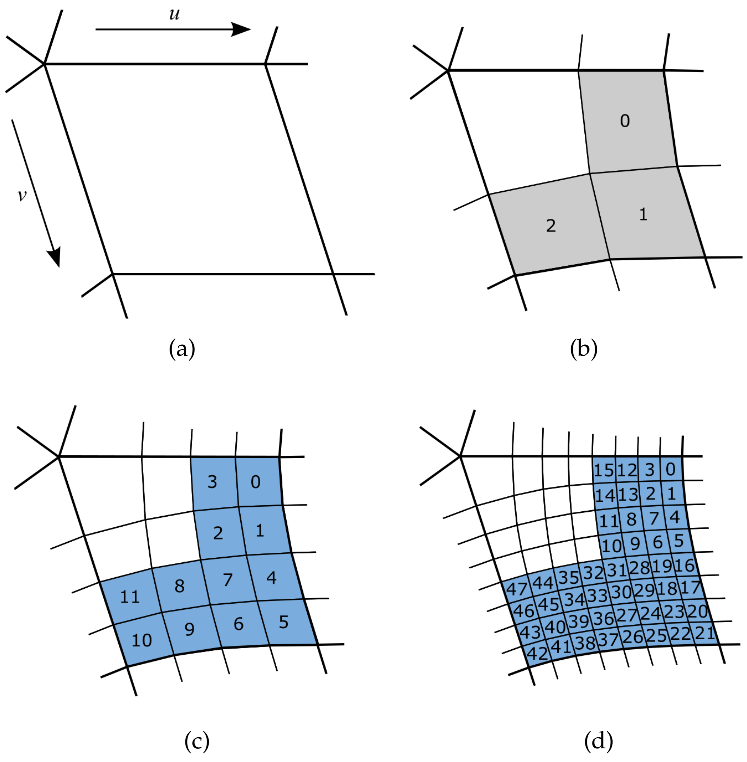


Figure 3.5: For the NURBS compatible subdivision scheme, the support region of an EV is larger than for Catmull-Clark. No sector can be evaluated as a B-spline after one subdivision step (b). Two subdivision steps (c) are required to define regular sectors (blue) of degree 3 from an initial face with an EV (a). For higher degree surfaces, even more subdivision steps are necessary, i.e. three steps for degree 5 (d).

the size of the support region of an EV in the NURBS compatible subdivision scheme, where stencil entries have been tuned to guarantee bounded curvature at and around the EV, sectors shaded in gray in Figure 3.5b cannot be evaluated. After the second subdivision step, 12 sectors can be evaluated if the surface is of degree 3, as illustrated in Figure 3.5c. More subdivision steps are required to evaluate subdivision surfaces of higher degree around EVs. With increasing degree, the number of sectors that can be evaluated as a B-spline surface after a sufficient number of subdivision steps increases. Figure 3.5d shows 48 sectors that can be evaluated for a degree 5 surface, which only appear after three subdivision steps.

To create the regular sectors for the picking matrix from the initial control points, as required by Equation 3.4, the extended subdivision matrix $\bar{A}$ therefore performs r subdivision steps for the NURBS compatible subdivision scheme.

Like Stam, the presented approach also assumes that the EV is located at the origin of the local parameterization of the patch. For given (u, v) coordinates, the corresponding sector index i and sector-local (u_i, v_i) coordinates within this sector can be determined by virtually following the quad-tree defined by the subdivision levels from the initial quad to the smallest refinement. The (u_i, v_i) coordinates are adjusted along the way so that they are always in the range $(u_i, v_i) \in [0, 1]^2$ for the current sector, as shown in the procedure from Listing 3.1.

Listing 3.1: Computing the sector index and local parameterization for a given parameter location.

```python
def sector_from_uv(u, v, degree):
    u_i, v_i = u, v
    r = (degree + 1) / 2
    i = 0
    for step in range(r):
        LEVELS = r - step - 1
        SECTORS_PER_QUAD = pow(pow(2, LEVELS), 2)
        if u_i > 0.5 and v_i <= 0.5:
            u_i = 2.0 * u_i - 1.0
            v_i = 2.0 * v_i
        elif u_i > 0.5 and v_i > 0.5:
            i += SECTORS_PER_QUAD
            u_i = 2.0 * u_i - 1.0
            v_i = 2.0 * v_i - 1.0
        elif u_i <= 0.5 and v_i > 0.5:
            i += 2 * SECTORS_PER_QUAD
            u_i = 2.0 * u_i
            v_i = 2.0 * v_i - 1.0
        else:
            i += 3 * SECTORS_PER_QUAD
            u_i = 2.0 * u_i
            v_i = 2.0 * v_i
        return i, (u_i, v_i)
```

To evaluate an irregular patch at the patch-local parameter location $(u, v) \in [0, 1]^2$, first the number of subdivision steps n needs to be determined so that the desired parameter location is within a regular sector. Like for Catmull-Clark, the number of steps is defined as $n = \lfloor \min(-\log_2(u), -\log_2(v)) \rfloor$. During subdivision, the coordinates of the parameter location are scaled to $(u_s, v_s) = (2^{n-1}u, 2^{n-1}v)$. These scaled coordinates are in turn used to find the corresponding sector index i and sector-local parameters (u_i, v_i) as defined by Listing 3.1.

Having i, n and (u_i, v_i), the regular sector can be evaluated, without explicitly subdividing, by using the precomputed eigenstructure in the same way as in Equation 3.5:

$$s_{i,n}(u_i, v_i) = \hat{C}_0^T \Lambda^{n-1} x(u_i, v_i, i)\,. \tag{3.6}$$

This enables fast evaluation of irregular regions of any NURBS compatible subdivision surface at arbitrary parameter values.

One drawback of this approach for NURBS compatible subdivision surfaces is that the precomputed eigenstructure gets very large with increasing degree. For higher degree surfaces, the support of the basis functions is larger, requiring more subdivision steps to get regular sectors. As a result, the number of sectors that can be evaluated per face increases, as shown in Figure 3.5. As the eigenstructure for each sector needs to be precomputed for each supported valence and degree, this limits the practical use of this approach. For valences from 3 to 12 the eigenstructure has a size of approximately 1 MB for degree 3. For higher degrees the size increases to 13 MB, 143 MB and 1.29 GB for degrees 5, 7 and 9 respectively. Due to this rapid growth in size it becomes inefficient to precompute these structures for degrees higher than 9.

Due to the re-parameterization of the surface during the implicit subdivision to evaluate the point of interest in a regular sector, derivatives evaluated with this approach need to be scaled back to the initial parameterization. Stam [91] suggested to scale the p-th derivative by a factor of 2^{np}, where n is the number of subdivision steps. For the NURBS compatible subdivision scheme the additional $r-1$ subdivision steps of the extended subdivision matrix must be taken into account and derivatives need to be scaled by $2^{(n+r-1)p}$. As already pointed out by Stam, this scaling causes derivatives close to an EV, where many subdivision steps are required to evaluate points and n becomes large, to diverge.

Despite this issue, this approach is favored for evaluating irregular surface regions because it always evaluates the exact geometry of the limit surface, not only an approximation as for the patching method described in Section 3.1.3.1. The effects of diverging derivatives and improvements to reduce them for analysis are discussed in the next section.

For the remainder of this thesis, the presented extension to Stam's algorithm is used to evaluate quantities of all surfaces, except when noted otherwise.

3.1.4 *Numerical Integration*

For analysis, a common requirement is to integrate arbitrary functions over a domain, represented by the surface geometry. To do so, numerical integration techniques like Gaussian quadrature [94] are used.

Gaussian quadrature allows to integrate functions on a parametric surface by only evaluating it at certain specific parameter values x_i, the quadrature points, and compute the final result as a weighted sum of these individual results with weights w_i:

$$\int_{-1}^{1} f(x)\,dx \approx \sum_{i=1}^{k} w_i f(x_i) \tag{3.7}$$

A k point Gaussian quadrature rule enables integrating polynomials of degree $2k-1$ exactly. Therefore, for surfaces of bi-degree d, a $\frac{d+1}{2} \times \frac{d+1}{2}$ Gaussian quadrature rule is used. The integration is exact for regular, uniform, non-rational parts of a subdivision surface, i.e. where polynomial B-spline basis functions are used. However, the surface around an EV is not a single B-spline patch and cannot be exactly represented by a polynomial. As a result, errors in analysis results are observed in regions near EVs, as will be shown in Section 3.2. Similarly, regions with non-uniform knot vectors or rational basis functions also lead to errors using standard Gaussian quadrature, as shown in Table 3.1.

To improve analysis results, the mesh can be refined (h-refinement), the degree of the surface can be increased (p-refinement) or the numerical integration can be improved. Numerical integration can be improved for example by using more quadrature points. Also, for other subdivision schemes, alternative quadrature rules were proposed to improve numerical integration, e.g. mid edge quadrature for Loop subdivision [50]. This section focuses on improving Gaussian quadrature for numerically integrating NURBS compatible subdivision surfaces.

Because using many quadrature points for integration increases computation time for analysis significantly, *adaptive* quadrature is proposed as a way to efficiently improve analysis results in regions where standard Gaussian quadrature is not exact.

Based on Gaussian quadrature, I present a simple adaptive quadrature scheme for arbitrary subdivision surfaces which adaptively increases the number of quadrature points in three cases:

- for elements with non-uniform knot vectors
- for rational elements, and
- for elements within the support of an EV.

The exact number of quadrature points used for each of these cases requires a trade-off between accuracy and performance. Here, they were chosen to get a significant improvement in accuracy over standard quadrature, but still get reasonable performance for common mesh configurations, as the performance heavily depends on the actual mesh used.

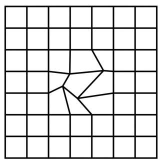
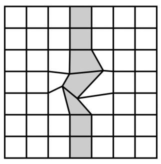
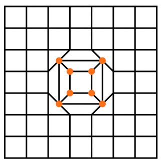
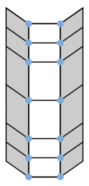
In the first case, for elements with non-uniform knot vectors, the number of quadrature points is increased by one in each parametric direction, e.g. instead of a 2×2 Gauss rule, a 3×3 quadrature rule is used for a surface of degree 3. In other words, for elements of degree d with non-uniform knot vectors $\frac{d+3}{2} \times \frac{d+3}{2}$ Gaussian quadrature is used.

Similarly, for rational elements the number of quadrature points is increased by two, leading to $\frac{d+5}{2} \times \frac{d+5}{2}$ Gaussian quadrature for rational elements of degree d.

Finally, there are two cases around extraordinary vertices: elements directly adjacent to an EV and elements which have an EV within their support. Experiments, comparing the integrated surface area with varying number of quadrature points to known reference values, showed that using at least a 4×4 Gaussian quadrature rule around EVs improves integration accuracy significantly. This is in accordance to [105] where a 4×4 Gaussian quadrature for both regular and irregular elements of Catmull-Clark subdivision surfaces was used to improve results. For better convergence of analysis results under h-refinement, the remainder of this thesis uses a 10×10 quadrature rule for elements directly adjacent to an EV, whereas for all other irregular regions a 5×5 rule is used, unless specified otherwise. For higher degree surfaces standard $\frac{d+1}{2} \times \frac{d+1}{2}$ Gaussian quadrature is used if $\frac{d+1}{2}$ is larger than the predefined number of adaptive quadrature points.

Table 3.1 lists the absolute errors of standard Gaussian quadrature and the proposed adaptive integration scheme for integrating the surface area of different subdivision surfaces of degree 3 with a known area. The grid has a reference surface area of 4.0. Different variations are tested: with uniform knot vectors, with non-uniform knot vectors with a zero knot interval (shaded in gray), and with multiple EVs (marked in orange). Additionally, a rational control mesh for a quarter cylinder with a reference area of 4.71239 is compared. Its con-

Table 3.1: Comparison of integration errors of standard Gaussian quadrature and adaptive integration for different control meshes.

				
Mesh	Uniform	Non-Uniform	EV	Rational
Gaussian	$1.77 \cdot 10^{-15}$	$2.17 \cdot 10^{-4}$	$3.15 \cdot 10^{-2}$	$1.07 \cdot 10^{-2}$
Adaptive	$1.77 \cdot 10^{-15}$	$8.88 \cdot 10^{-16}$	$2.97 \cdot 10^{-5}$	$1.74 \cdot 10^{-5}$

trol mesh has zero knot intervals at the boundary (gray) and control points with a rational weight of 0.804738 (marked in blue). In all cases, the adaptive integration scheme significantly reduces the absolute error of the computed surface area at the cost of increased computation time.

Improvements of analysis accuracy as a result of using this adaptive quadrature scheme will be further discussed in Section 3.2.1.

3.1.5 *Forces and Constraints*

To actually be able to solve the system of equations resulting from integrating the thin shell equations, boundary conditions in the form of constraints and forces need to be applied to the equations. The physical interpretation of this requirement is that without any constraints the object is free floating in space, and any applied force will move it infinitely. Therefore, in three dimensions, at least six DOF need to be constrained to prevent any rigid body translation and rotation. While constraints are necessary to restrict any rigid body motion of the surface, and to enforce other boundary conditions, forces are used to simulate different real world environmental influences.

There are different types of forces that can be applied for analysis. In the simplest case, a force vector $\boldsymbol{f} = (f_x, f_y, f_z)$ is applied directly to a single control point of the subdivision control mesh. In classical FEM, these forces are typically called *point forces* or *point loads*, because they are only applied locally to a single node. For IGA, however, this is generally not true. Here, a force applied to a single subdivision control point spreads to all other basis functions that have non-zero support at this control point. To define a real point force, acting only on a single subdivision control point, a non-uniform knot vector with multiple knots, similar to the boundary case, can be used to define an interpolating control point, which is only influenced by a single basis function. Point forces can be exactly applied at such interpolating control points. This is an advantage of the NURBS compatible subdivision scheme, which allows non-uniform knot vectors, over Catmull-Clark or Loop, where point forces cannot be accurately defined. Note that for thin shell analysis such an interpolating control point can only be defined at the boundary of the surface, as multiple knots reduce the continuity of the surface to C^0 at the interpolated point, while Kirchhoff-Love shells require C^1.

To apply forces to any location away from the subdivision control points, evaluating the corresponding basis functions at some parameter location (u, v) yields a set of weights that can be used to spread the force to all relevant basis functions. Similar to above, this also means that the force is not a point force, acting on a single node, but is spread over a larger region depending on the basis functions. Except at interpolating control points, as described above, exact point forces cannot be defined with subdivision based IGA.

Another common type of force is pressure or body force. The difference between them is that while pressure only acts on the surface of the material, body forces are applied throughout the thickness/volume of the material. While the forces mentioned previously act only locally, pressure and body forces are defined per unit area/volume and are therefore numerically integrated over the region of interest.

To constrain the solution space for IGA, different types of boundary conditions can be applied. The most common type of boundary conditions are *Dirichlet conditions*, which prescribe the function value of the solution, i.e. the displacements for thin shells, at certain locations.

In its simplest form, the displacement of a subdivision control point is prescribed to a constant vector $\boldsymbol{c} = (c_x, c_y, c_z)$. For thin shells, this amounts to constraints for the corresponding three unknowns, for x, y, z displacement, in the system of equations, as shown in Equation 3.8.

$$u_i = c_x \qquad u_{i+1} = c_y \qquad u_{i+2} = c_z \tag{3.8}$$

Different methods can be used to enforce the prescribed values of these unknowns when solving the system of equations, some of them are explained e.g. in [40] or [39]. One approach is to (conceptually) partition the system of equations from Equation 2.52 into equations corresponding to unknown DOF, in the following indicated by subscript u, and equations corresponding to DOF prescribed by Dirichlet conditions, using subscript d. This leads to a system of equations of the form:

$$\begin{bmatrix} \boldsymbol{K}_{\mathrm{dd}} & \boldsymbol{K}_{\mathrm{du}} \\ \boldsymbol{K}_{\mathrm{ud}} & \boldsymbol{K}_{\mathrm{uu}} \end{bmatrix} \begin{bmatrix} \boldsymbol{u}_{\mathrm{d}} \\ \boldsymbol{u}_{\mathrm{u}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f}_{\mathrm{d}} \\ \boldsymbol{f}_{\mathrm{u}} \end{bmatrix} \tag{3.9}$$

The unknowns in $\boldsymbol{u}_{\mathrm{d}}$ are prescribed and thus known, therefore the equations from the first row of Equation 3.9 can be discarded or replaced with trivial equations. The remaining equations can be written as:

$$\boldsymbol{K}_{\mathrm{ud}}\boldsymbol{u}_{\mathrm{d}} + \boldsymbol{K}_{\mathrm{uu}}\boldsymbol{u}_{\mathrm{u}} = \boldsymbol{f}_{\mathrm{u}} \tag{3.10}$$

Moving the first term, which is also known, to the right-hand side yields the modified system of equations with the applied constraints:

$$\boldsymbol{K}_{\mathrm{uu}}\boldsymbol{u}_{\mathrm{u}} = \boldsymbol{f}_{\mathrm{u}} - \boldsymbol{K}_{\mathrm{ud}}\boldsymbol{u}_{\mathrm{d}} \tag{3.11}$$

Note that for IGA based on approximating subdivision schemes, like Catmull-Clark or the NURBS compatible subdivision scheme, the displacement of the limit surface is generally not equal to the displacement of the subdivision control points, which are the degrees of freedom for analysis.

To actually constrain the displacement of the limit surface, *Lagrange multipliers* can be used, as explained e.g. in [39]. They are added as additional rows and columns to the stiffness matrix and allow more general constraints of the form:

$$\alpha_1 u_i + \alpha_2 u_j + \cdots + \alpha_n u_m = c \tag{3.12}$$

The coefficients α_i for all Lagrange multiplier constraints define a matrix $\boldsymbol{A}$. Similarly, the right hand sides of the constraints, i.e. c in Equation 3.12, are collected into a vector $\boldsymbol{b}$. Together, an augmented

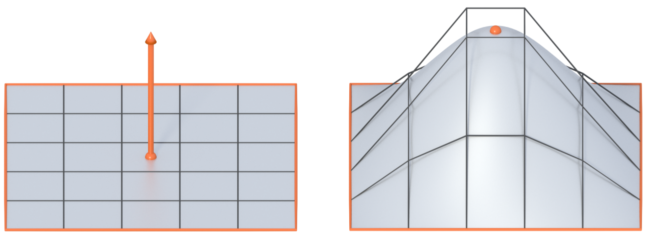


Figure 3.6: Displacing the subdivision limit surface using a Lagrange multiplier constraint.

system of equations is defined that includes the Lagrange multipliers $\boldsymbol{\lambda}$ as additional unknowns:

$$\begin{bmatrix} \boldsymbol{K} & \boldsymbol{A}^T \\ \boldsymbol{A} & 0 \end{bmatrix} \begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{\lambda} \end{bmatrix} = \begin{bmatrix} \boldsymbol{f} \\ \boldsymbol{b} \end{bmatrix} \tag{3.13}$$

These unknowns $\boldsymbol{\lambda}$ can be interpreted as the constraint forces required to satisfy the Lagrange multiplier constraints.

By using the values of the subdivision limit stencil for a particular point on the limit surface as factors α_i for the corresponding unknowns, the displacement of the limit surface can be prescribed for analysis. Again, to constrain the x, y, z displacement of a particular point on the limit surface, three separate constraints are necessary. Figure 3.6 shows an example for exactly displacing a point on the limit surface independent of the control points. The control points are moved automatically by the thin shell simulation to satisfy the constraint.

Green [44] shows more use cases of this type of constraints. One example is to constrain the tangent of the boundary of a subdivision surface without also constraining all derivatives to zero, as is the case for other implementations based on Loop or Catmull-Clark discussed in Section 3.1.2. Using the NURBS compatible subdivision scheme, additional Lagrange multiplier constraints are not necessary as boundaries are defined with multiple knots. In this case, a simpler type of constraint can be used to restrict boundary tangents.

To define symmetries or to fix tangents at the boundary, two unknowns u_i and u_j can be linked so that their value is directly related by some factor β:

$$u_i = \beta u_j \tag{3.14}$$

Using $\beta = 1$, the two unknowns are constraint to have the same value. If this is applied to unknowns of a boundary control point and its immediate neighbor, the orientation of the tangent is fixed at this point, as described in Section 3.1.2.

On the other hand, for a boundary explicitly modeled using ghost vertices, $\beta = -1$ can be used for the unknowns of the control points immediately inside and outside of the boundary to maintain the correct position of the ghost vertices (assuming the boundary control point itself is fixed).

3.2 COMPARISON OF ANALYSIS RESULTS

In this section, the isogeometric analysis method based on NURBS compatible subdivision surfaces, as described in Section 3.1, will be evaluated regarding correctness and convergence with various numerical example problems commonly used in the literature.

3.2.1 *Poisson Equation*

The first example does not yet use the subdivision-based Kirchhoff-Love formulation, but instead demonstrates solving a simpler Poisson equation. This equation can also be found as an example problem in other IGA work e.g. [56, 65].

The Poisson equation from Equation 3.15 is solved on the square domain $\Omega = [-1, 1]^2$ with homogeneous Dirichlet boundary conditions on the surface boundary $\partial\Omega$. The operator ∇^2 in Equation 3.15 is the Laplace operator on the manifold.

$$\begin{aligned} -\nabla^2 u &= 1 \quad \text{in } \Omega \\ u &= 0 \quad \text{on } \partial\Omega \end{aligned} \tag{3.15}$$

The reference solution for this problem, in form of a series, as provided by Lee [56], is:

$$\begin{aligned} u(x, y) = \frac{1 - x^2}{2} - \frac{16}{\pi^3} \sum_{\substack{k=1 \\ k \text{ odd}}}^{\infty} \frac{\sin(k\pi(1+x)/2)}{k^3 \sinh(k\pi)} \times \\ (\sinh(k\pi(1+y)/2) + \sinh(k\pi(1-y)/2)) \end{aligned} \tag{3.16}$$

Having a reference solution for the whole surface domain allows a detailed comparison of the expected and actual analysis results.

Figure 3.7 shows (a) the parameterization of the domain defined by a regular subdivision surface of degree 3 with (b) the solution of the Poisson equation from Equation 3.15. Note that the subdivision control mesh, overlaid with dashed lines in (b), has more faces near the boundary, compared to the parameterization. The reason is that

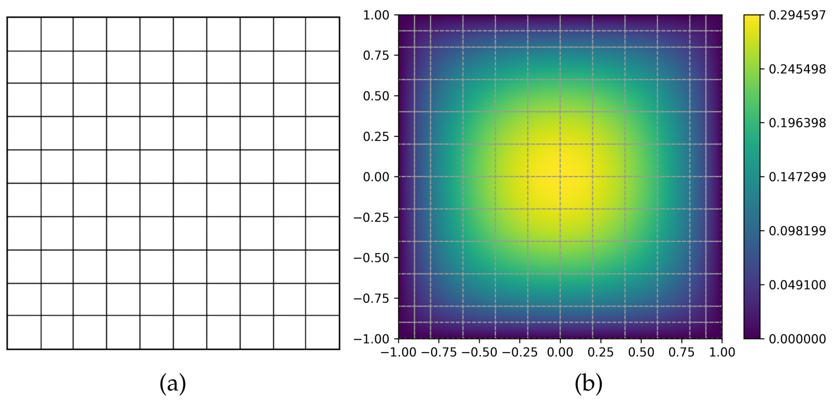


Figure 3.7: A regular parameterization of the domain (a) is used to solve the Poisson equation. The numerical result (b) is overlaid with the subdivision control mesh used to define the elements of the parameterization.

the boundary is defined using multiple knots, as explained in Section 3.1.2. Faces corresponding to zero knot intervals vanish in the parameterization and are thus not considered an *element* for analysis.

Figure 3.8 compares the errors of solutions using (a) standard Gauss quadrature and (b) the adaptive quadrature presented in Section 3.1.4. Because all elements of the domain are regular, adaptive quadrature is only used near boundaries, where knot vectors are non-uniform because they contain multiple knots. Even for this case, adaptive quadrature significantly reduces the overall error of the solution. The L^2 error of the result in Figure 3.8a, using standard Gauss quadrature, is 0.00265 while for the result in Figure 3.8b, using adaptive quadrature, it reduced to 0.00215. The remaining errors are mainly due to the coarse control grid, which cannot accurately represent the solution. A more accurate result can be obtained with h-refinement, i.e. by subdividing the control mesh and thus reducing the overall element size.

To test the effect of extraordinary vertices on the result, and to show improvements of adaptive quadrature, the domain has also been parameterized using an irregular subdivision surface of degree 3, shown in Figure 3.9a. This parameterization has slightly more elements compared to the previous regular example and therefore leads to lower errors in most cases. The effect of adaptive quadrature is shown in Figure 3.9b, where regions near EVs and near the boundary have more quadrature points than regular regions, as explained in Section 3.1.4.

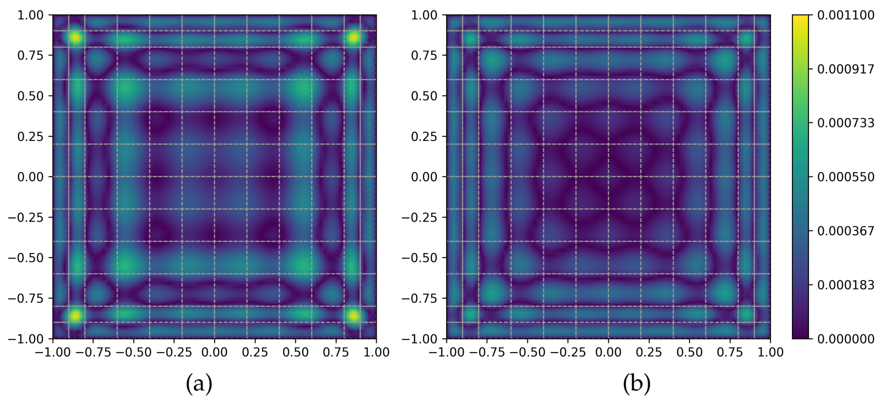


Figure 3.8: Error of the numerical solution of the Poisson equation on the regular domain using (a) standard Gauss quadrature and (b) adaptive quadrature.

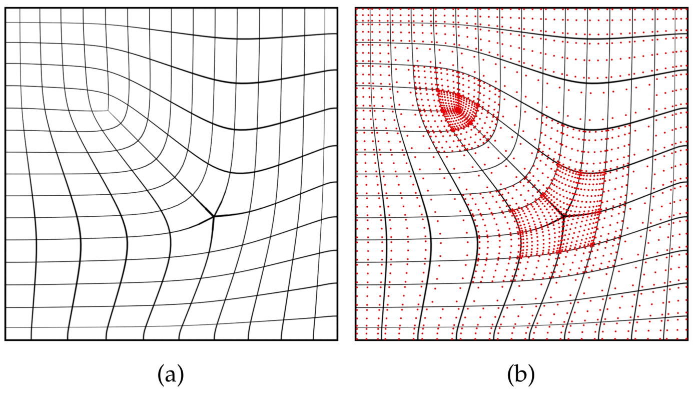


Figure 3.9: Irregular parameterization of domain for Poisson equation (a) and location of quadrature points for adaptive quadrature (b).

The results using standard Gauss quadrature in Figure 3.10a show a larger error in regions near extraordinary vertices. Using the adaptive quadrature scheme from Section 3.1.4 the errors in Figure 3.10b are uniformly small over the whole domain, unaffected by extraordinary vertices. This improvement is also reflected in the L^2 error, which is 0.00222 with standard Gauss quadrature and only 0.00109 with adaptive quadrature.

To compare the rate of convergence under h-refinement of the different parameterizations and integration methods, multiple subdivision levels of each variant are computed. Following [7], the measure of element size h is defined as $h = \sqrt{\frac{A}{m}}$, with A being the surface area of the domain and m the number of elements. The results are shown in Figure 3.11. With the adaptive quadrature scheme, the L^2 error decreases for both the regular and the irregular parameterization approximately with order $O(h^3)$ for these degree 3 surfaces, which is

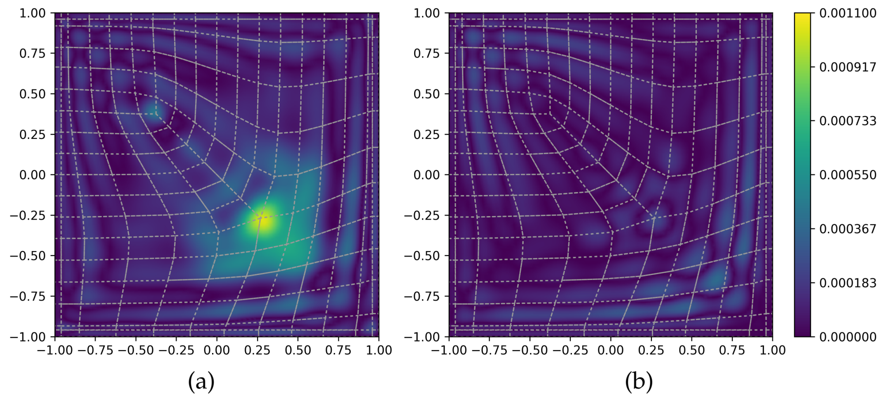


Figure 3.10: Error of the numerical solution of the Poisson equation on an irregular domain using (a) standard Gauss quadrature and (b) adaptive quadrature.

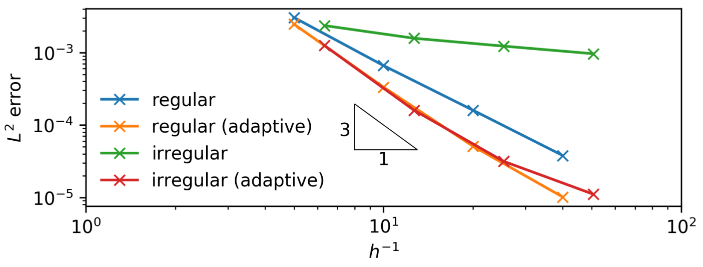


Figure 3.11: Convergence of solutions using h-refinement when solving the Poisson equation on different domains and with different quadrature rules.

optimal according to [65] and matches their results using Catmull-Clark subdivision. However, they needed to evaluate the subdivision surface at subdivision levels up to seven to get good accuracy near EVs. Note that without adaptive quadrature, irregular regions show slow convergence. Also, the boundaries of the regular mesh, defined using non-uniform knot vectors, have a slightly negative effect on convergence if not evaluated using adaptive quadrature. A possible explanation for this is that these regions are not uniformly refined during subdivision.

3.2.2 *Clamped Plate*

For the next example, a clamped plate subject to a uniform load is simulated using subdivision-based Kirchhoff-Love thin shell elements. Clamping the plate also constrains the normal at the boundary, in addition to the displacement of the boundary. As this requires accurate treatment of the boundary, including derivatives, this example

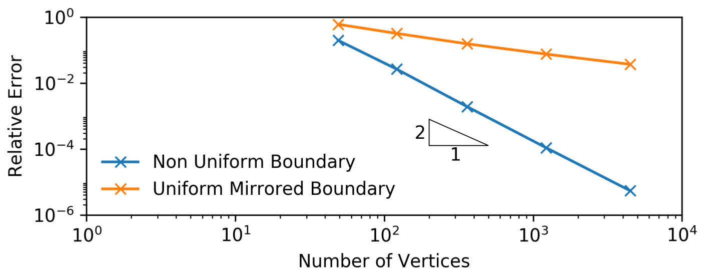


Figure 3.12: Convergence of solutions for the clamped plate problem using different representations for the boundary.

is used to compare the boundary representation using non-uniform knot vectors, explained in Section 3.1.2, to boundaries modeled with uniform elements, as used for example in [7, 25, 103].

The domain of the square plate with length $L = 10$ is defined using regular meshes, similar to the regular input for the Poisson example in Figure 3.7. The first variant defines the boundary using uniform elements by mirroring vertices inside the boundary to the outside, creating additional ghost vertices defining the boundary. The second variant makes full use of the NURBS compatible subdivision scheme and defines the boundary using multiple knots, creating Bézier end conditions, as explained in Section 3.1.2.

The materials are chosen as in [58], where this example problem is used to compare the convergence of shells with a high slenderness ratio. The Young's modulus is $E = 1.092 \cdot 10^6$, Poisson's ratio is $\nu = 0.3$ and the thickness is $t = 1.0 \cdot 10^{-5}$ to define a plate with slenderness ratio of $L/t = 10^6$. The uniform load is defined as $p = 1.0$.

The reference solution for this example is discussed in [96], where the maximum displacement u of the square plate is defined as

$$u_{ref} = \frac{0.001265319087pL^4}{D} \quad \text{with} \quad D = \frac{Et^3}{12(1-\nu^2)} \tag{3.17}$$

Figure 3.12 shows the convergence of the relative error $|\frac{u-u_{ref}}{u_{ref}}|$ for the two mesh variants under refinement through subdivision. As already observed by Green [44], the error for the mesh with boundaries defined using uniform elements converges slowly. Green solved this by using Lagrange multipliers to enforce the clamping at the boundary, instead of using constraints for the boundary control points and their immediate neighbors. However, this approach adds new unknowns, the Lagrange parameters, to the system of equations, increasing the time needed to solve the system.

In contrast, using non-uniform knot vectors with multiple knots to define the boundaries leads to optimal convergence in the order of $O(N^2)$, where N is the number of control points in the mesh, without any additional unknowns.

3.2.3 *Patch Test*

The patch test is often used to assess whether finite elements are convergent and if they are implemented correctly. However, the patch test is only an indicator and is neither necessary, nor sufficient to assess the usefulness of a finite element implementation.

The patch test tests various rigid body motions and constant strain states.

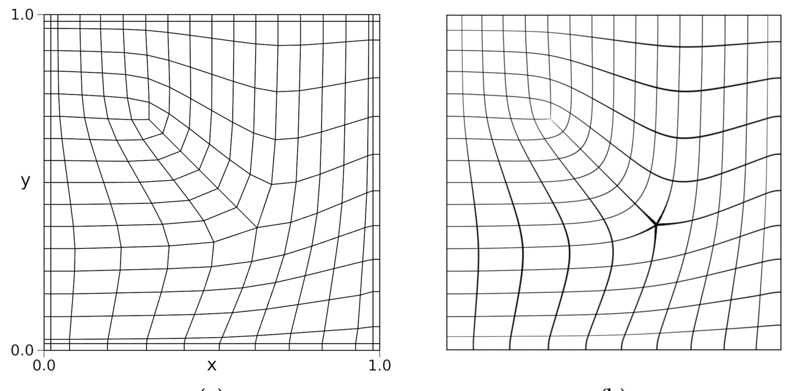
According to Cottrell *et al.* [30], an important property to pass the patch test is that the basis functions form a partition of unity. While this is indeed important, because having a partition of unity is required for the geometry to be affine invariant, it is not enough.

Most subdivision schemes like Loop, Catmull-Clark or the NURBS compatible subdivision scheme do form a partition of unity, but finite elements using these representations do generally not pass the patch test. The reason is that using Stam's method for evaluation, as described in Section 3.1.3.2, derivatives cannot be evaluated exactly close to extraordinary vertices.

The same is true for T-splines containing extraordinary points (which, for T-splines, are often called star points). To represent a T-spline surface with extraordinary points, a subdivision scheme called T-NURCCS is employed. To pass the patch test using T-splines, a technique called Bezier extraction is used, which transforms the T-spline surface into regular NURBS patches around extraordinary vertices. The same technique can also be used for subdivision surfaces, e.g. as described by Peters [69] for Catmull-Clark surfaces and discussed in Section 3.1.3.1 for NURBS compatible subdivision.

This allows to pass the patch test using subdivision-based isogeometric analysis as demonstrated in the following examples.

The setup for all tests is similar to patch tests by Scott [85]. For all tests an irregular degree 3 surface, with two EVs, is used to define the unit square domain $\Omega = [0, 1]^2$. The control mesh and parameterization of this initial mesh are shown in Figure 3.13. The boundary $\partial\Omega$



(a) (b)

Figure 3.13: Control mesh (a) and parameterization (b) of surface used for patch tests.

of the surface is constraint differently for each test. In most cases, displacements u_x along the x-axis or u_y along the y-axis are prescribed for the whole boundary $\partial\Omega$. For the stretching tests, some constraints affect only a certain part of the boundary, i.e. b_x constrains the x component of boundary vertices along the y-axis with $x = 0$ and b_y restricts the y component of boundary vertices along the x-axis with $y = 0$.

The performed patch tests and their boundary conditions are:

- Non-rigid in plane rotation

$$u_x = 0.1y$$
$$u_y = -0.1x$$

- Horizontal shearing

$$u_x = 0.1y$$
$$u_y = 0.0$$

- Vertical shearing

$$u_x = 0.0$$
$$u_y = 0.1x$$

- Stretching along the x-axis

$$u_x = 0.1x$$
$$b_y = 0.0$$

- Stretching along the y-axis

$$b_x = 0.0$$
$$u_y = 0.1y$$

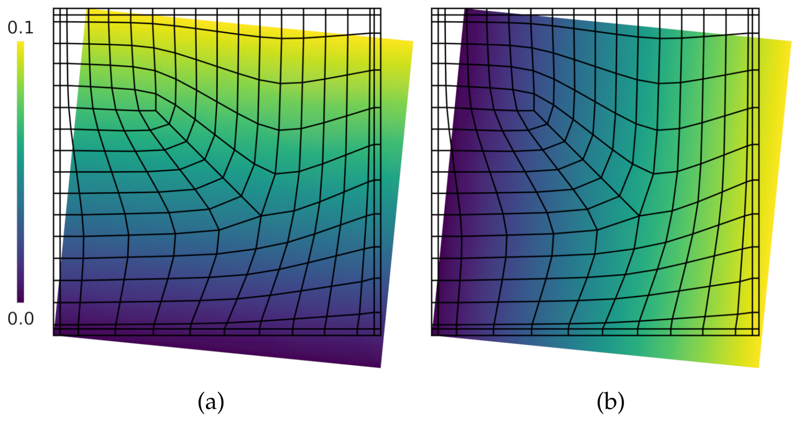


Figure 3.14: Results for the non-rigid rotation test using the PCCM method showing the expected linear displacement profiles in x-direction (a) and y-direction (b).

- Rigid translation on the x-axis

$$u_x = 0.1$$
$$u_y = 0.0$$

- Rigid translation on the y-axis

$$u_x = 0.0$$
$$u_y = 0.1$$

For the simulated material of the shell the Young's modulus $E = 1$ and Poisson's ratio $\nu = 0.3$ are used. The thickness, although not relevant for these examples, is set to $t = 0.1$.

Both evaluation methods described in Section 3.1.3 are compared. Additionally, for the evaluation based on Stam's algorithm from Section 3.1.3.2 standard Gauss quadrature is compared to the adaptive quadrature scheme discussed in Section 3.1.4. For the other evaluation method, using the PCCM algorithm to extract NURBS patches, all elements are evaluated using 3×3 Gauss quadrature, to account for the non-uniform parameterization of these extracted patches.

Table 3.2 summarizes the maximum errors for the displacements of all evaluated patch tests computed using double precision floating-point numbers. Only the PCCM evaluation method passes all patch tests with all errors being zero up to machine precision. While the adaptive quadrature scheme is an improvement over standard Gauss quadrature, it is still not sufficient to pass all patch tests exactly.

The results for all patch tests using the PCCM method are shown in Figures 3.14 (rotation), 3.15 (shear), 3.16 (stretch) and 3.17 (transla-

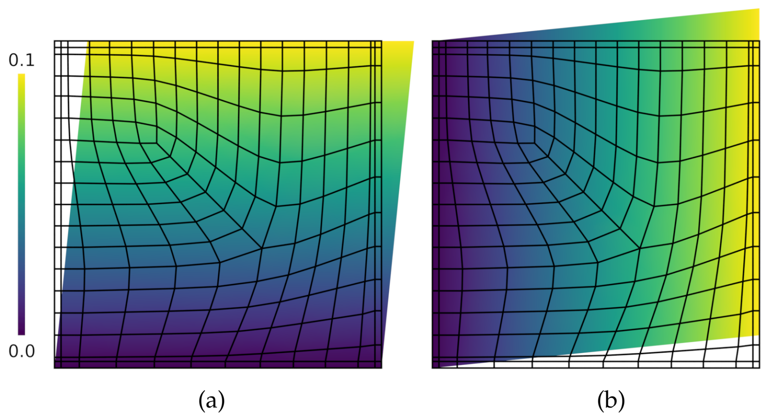


Figure 3.15: Results for the horizontal (a) and vertical (b) shearing tests. Using the PCCM method results in exactly linear displacement profiles in x-direction (a) and y-direction (b).

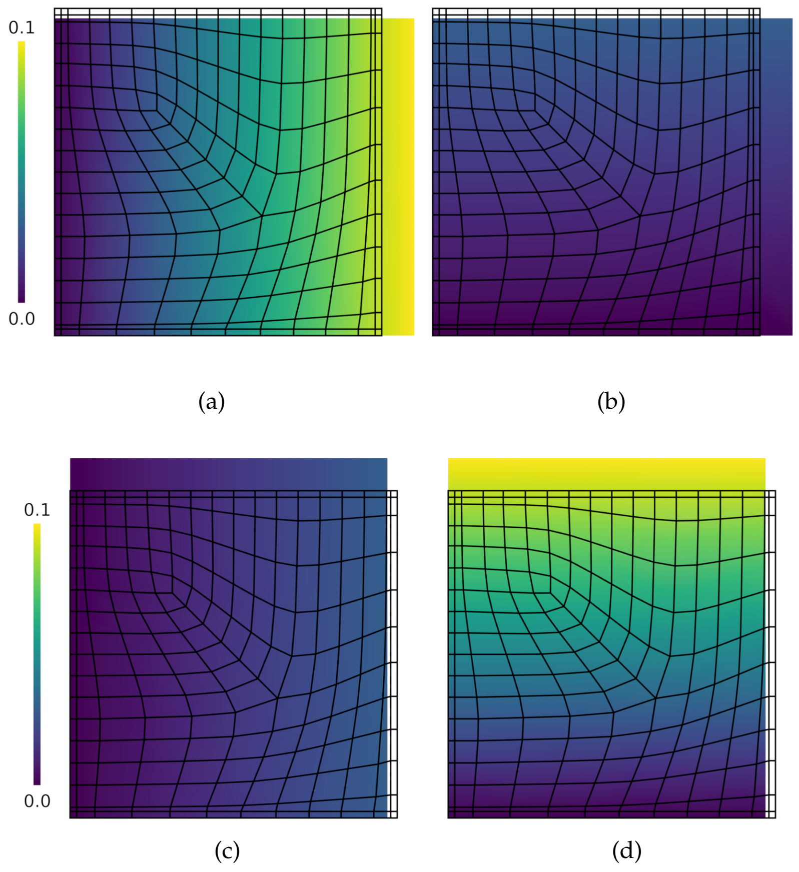


Figure 3.16: Results for the stretch patch tests in x-direction (a, b) and y-direction (c, d). In all cases, the PCCM method reproduces exactly linear displacement profiles in both x-direction (a, c) as well as in y-direction (b, d).

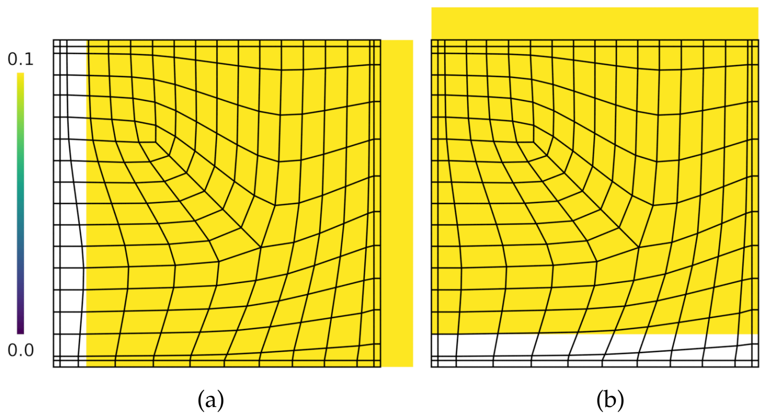


Figure 3.17: Results for the translation patch tests in x-direction (a) and y-direction (b). With the PCCM method, exactly constant displacements are achieved in both cases.

tion), demonstrating exactly linear or constant displacements for all tests.

Table 3.2: Maximum errors for different patch tests.

Test	PCCM	Stam	Stam (adaptive)
Rotation	2.51E-16	4.13E-16	2.37E-16
Shear Horizontal	2.25E-16	8.37E-05	2.42E-06
Shear Vertical	1.67E-16	8.37E-05	2.42E-06
Stretch X	2.42E-16	2.01E-04	4.59E-06
Stretch Y	3.49E-16	1.98E-04	4.56E-06
Translation X	2.97E-16	3.39E-15	3.29E-15
Translation Y	3.06E-16	3.35E-15	3.30E-15

It has been shown that IGA based on NURBS compatible subdivision surfaces is able to exactly pass the patch test using a patch extraction method, similar to how T-splines pass the patch test [85]. To the author's knowledge, this has not been shown before for any subdivision scheme like Catmull-Clark, and was by some considered a weak point of subdivision surfaces for isogeometric analysis.

However, despite required to pass the patch test, for the rest of this thesis the patch extraction method is not used for evaluation of the subdivision surface. The main reasons are that, for practical use, the evaluation method based on Stam's approach is faster, with similar accuracy, and is able to exactly evaluate curved surfaces near EVs, which are only approximated using the patch extraction approach.

3.2.4 *Internal Pressure*

In this example a cylindrical shell subject to internal pressure is simulated. Similar problems are solved for example in [48] and [103]. Noteworthy about this example is that there exists an analytic reference solution by Timoshenko and Woinowsky-Krieger [97] for the whole surface. Therefore, this example is ideally suited for a detailed comparison of different surface representations. In the following, the importance of a rational representation and advantages of non-uniform parameterization and higher degree surfaces are highlighted.

For reference, the analytic solution for the displacement u from [97], to which all results are compared, is provided in Equation 3.18. For a cylinder with length L, radius R and shell thickness t the solution is valid for $x \in \left[-\frac{L}{2}, \frac{L}{2}\right]$. This solution assumes that the open boundaries of the cylinder are simply supported, i.e. their position is fixed, but the tangents are free.

$$
\begin{aligned}
u(x) = -\frac{pL^4}{64D\alpha^4}\Bigg(1 &- \frac{2\sin\alpha\sinh\alpha}{\cos 2\alpha + \cosh 2\alpha}\sin\beta x\sinh\beta x \\
&- \frac{2\cos\alpha\cosh\alpha}{\cos 2\alpha + \cosh 2\alpha}\cos\beta x\cosh\beta x\Bigg) \qquad (3.18)\\
\text{with} \qquad \alpha = \frac{\beta L}{2} \qquad \beta &= \left(\frac{Et}{4R^2D}\right)^{\frac{1}{4}} \qquad D = \frac{Et^3}{12(1-\nu^3)}
\end{aligned}
$$

For the following examples, a cylinder with length $L = 5.0$ and radius $R = 1.0$ is simulated with an assumed material of thickness $t = 0.1$ with Young's modulus $E = 10^5$ and Poisson's ratio $\nu = 0.0$.

Using NURBS compatible subdivision surfaces, a quarter of the cylinder can be defined exactly using a single rational patch. The solution for the whole cylinder is obtained by symmetry which is ensured by applying symmetry conditions to the corresponding boundaries of the quarter cylinder. To provide the necessary DOF for deformations along the length of the cylinder, a surface with three equally sized elements along the length is used as the initial representation, as shown in Figure 3.18a. Denser representations of the exact cylinder are obtained by simply subdividing.

Figure 3.19 shows the deformed results of the first refinement level of the initial surface for degrees 3, 5 and 7. The color indicates the absolute distance to the reference solution. There are too few degrees of freedom in the vertical direction to exactly represent the solution,

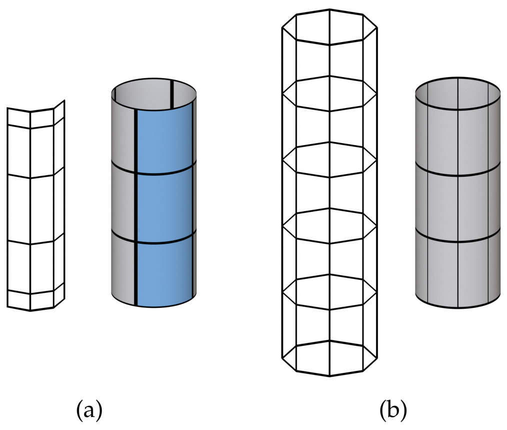


Figure 3.18: For the internal pressure example, two surface variants are compared: a rational quarter cylinder (a) and a Catmull-Clark approximation of a full cylinder (b). The rational surface uses symmetry to define the full cylinder and can be modeled with surfaces of different degree. Here, the degree 3 control mesh is shown for the given parameterization.

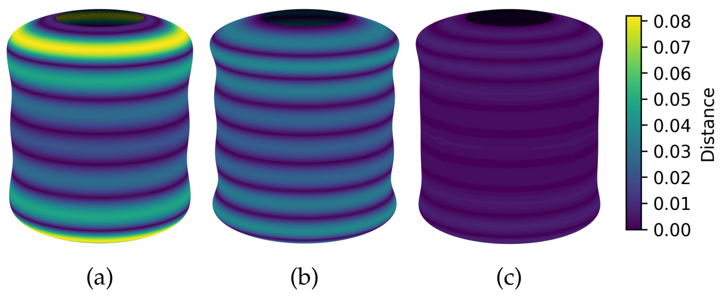


Figure 3.19: Errors for simulated internal pressure for the first refinement of rational surfaces of degree 3 (a), 5 (b) and 7 (c).

resulting in a wavy pattern in the error visualization. These errors vanish for results using more degrees of freedom along the length. Higher degree surfaces perform visibly better using the same number of elements in the surface. The degree 7 result is already close to the reference solution after the first refinement. Further subdivision, to provide more DOFs along the length of the cylinder, leads to results close to the reference solution, as shown in Figure 3.20 for the second refinement level of the surfaces.

Modeling the exact cylinder surface is not possible with Catmull-Clark due to lack of a rational representation. To get a good approximation of the cylinder surface, the full cylinder is modeled without symmetries. For the initial coarse surface the cylinder is approximated with eight control points along its circumference, sampled from an exact circle. Similar to the rational surfaces, three equally

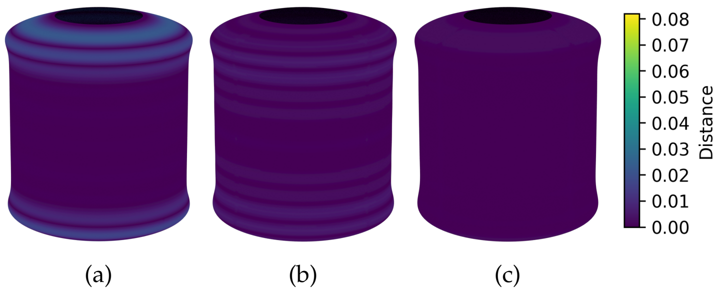


Figure 3.20: Errors for simulated internal pressure for the second refinement of rational surfaces of degree 3 (a), 5 (b) and 7 (c).

sized elements are defined along the length of the cylinder. The initial control mesh and the corresponding surface parameterization is shown in Figure 3.18b. Denser representations of the surface are also obtained via subdivision, as for the rational surfaces. Initial inaccuracies due to the non-rational representation will therefore also be present in the denser version of the geometry.

However, if the control points of the subdivision surface are placed on a circle of a certain radius r, the actual limit surface will have a radius smaller than that, because Catmull-Clark is an approximating subdivision scheme. To make up for this, the control points for the Catmull-Clark cylinder are placed on a circle with a larger radius to approximate a circle of radius r_{target} in the limit. The radius r depends on the number of control points n used to define the circle and can be derived from the limit stencil $S = [1, 4, 1]/6$ of the corresponding uniform B-spline curve of degree 3 by solving

$$\left\| \sum_{i=0}^{3} cp(r, n)_i S_i \right\| = r_{\text{target}} \tag{3.19}$$

for some curve segment of three control points cp from a circle centered at the origin with radius r which is approximated with n control points. For a Catmull-Clark cylinder modeled with eight control points around its circumference, the radius $r = 1.10819418755439$ results in a limit surface with a radius of approximately 1.0.

Figure 3.21 shows the results for the second and the third refinement of the initial cylinder geometry. The Catmull-Clark solution for the second refinement in Figure 3.21a has a larger error than the first and second refinements using NURBS compatible subdivision of degree 3 shown in Figure 3.19a and Figure 3.20a respectively. The reason for the larger errors near the top and the bottom of the cylinder is the different handling of boundaries. For the same number of elements along the height, the Catmull-Clark solution has less degrees of freedom inside the surface, because some vertices are outside the bound-

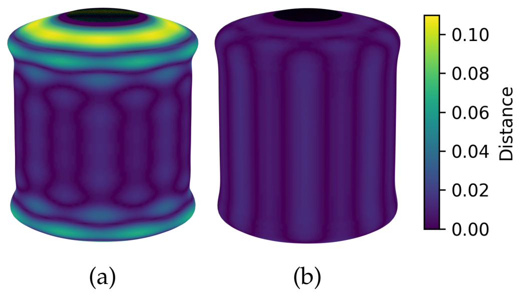


Figure 3.21: Errors for simulated internal pressure for the second (a) and third (b) refinement of a Catmull-Clark surface. The vertical lines of error are due to the inaccurate representation of the cylindrical geometry.

ary to define the boundary curve. In contrast, using a non-uniform parameterization to define the boundary for NURBS compatible subdivision surfaces, all control points are inside the surface. Therefore, the complex solution near the top and bottom boundaries can be better represented.

Also worth noting is that the Catmull-Clark solutions show visible vertical artifacts, which can be seen in Figure 3.21 and are also clearly visible in [103]. These artifacts resemble the initial control mesh using eight vertices along the circumference to define the cylinder and are also present in all results for the denser, subdivided meshes. The source of these errors is the initial non-rational control mesh, which is only an approximation of an exact cylinder. The geometric error introduced in this control mesh is also present in all subdivided meshes because the limit surface does not change with subdivision. To actually remove these errors for meshes with more DOFs, the denser meshes should be again sampled from an exact cylinder, not created by subdivision. However, this is only possible in simples cases like this, where the geometry is known to be a cylinder. In a larger structure, such artifacts due to geometric errors in the initial control mesh cannot be easily removed. This highlights the importance of a rational representation to exactly define common surfaces like circles, cylinders, spheres and other conic sections.

Figure 3.22 summarizes the results for the different geometry representations at different refinement levels. For all results, the L^2 error has been computed for only one quarter of the surface, to normalize between the variants using symmetry or not. The Catmull-Clark results are noticeable worse than the solutions using NURBS compatible subdivision surfaces of degree 3. Initially this is due to having less degrees of freedom near the boundary. With more degrees of freedom, like in Figure 3.21b where the solution at the boundary

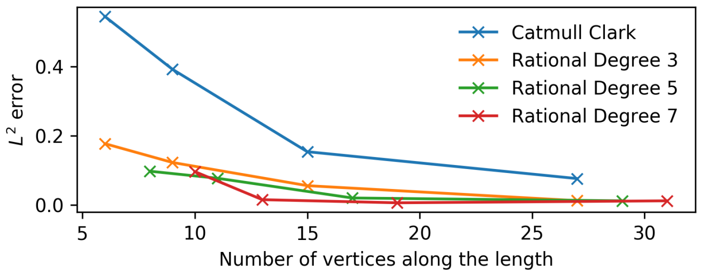


Figure 3.22: Convergence of L^2 error under mesh refinement for simulating internal pressure with different surface representations.

can be well represented, there remains an error from the geometric approximation of the exact cylinder. On the other hand, NURBS compatible subdivision surfaces, especially using higher degrees, perform well.

3.2.5 *Shell Obstacle Course*

The shell obstacle course consists of several benchmark problems defined in [11] and [60] and is commonly used in the literature to test the accuracy of (isogeometric) thin shell implementations, e.g. in [9, 25, 44, 48, 52]. While there are reference results given for all of the examples, they are not computed analytically, but are the result of a very dense simulation. Therefore, a slight deviation from these results is expected for different implementations, as is the case e.g. in [25, 52] and others.

All examples are described by simple cylindrical or spherical surfaces. While such surfaces can be exactly represented using rational NURBS compatible subdivision surfaces, they can only be approximated by other subdivision schemes such as Catmull-Clark.

Another advantage of the NURBS compatible subdivision scheme is the support for higher degree surfaces, which improve convergence rates, as will be demonstrated in the following.

These examples also demonstrate the use of symmetries. The main reason to use symmetries is that exact point forces, as required by some of the problems, can only be applied to interpolated control points, i.e. control points which are only supported by a single basis function. If multiple basis functions have support on a control point, each force applied to this control point spreads to the neighboring control points, according to the basis functions. Such interpolating control points can be defined using multiple knots, as used on the boundary. But without additional constraints they can only be located on the boundary to not violate the continuity requirements for Kirchhoff-Love shells. Therefore, symmetries are used to define surface boundaries and interpolating points at the location where a point force is required. Further, the use of symmetries allows to only simulate a part of the whole surface which reduces computation time significantly.

To compare isogeometric analysis based on NURBS compatible subdivision surfaces to existing implementations the following results for the shell obstacle course also include results using Catmull-Clark subdivision, i.e. only uniform elements of degree 3 with an explicitly modeled boundary using ghost vertices, as used in e.g. [7, 25, 103].

Because Catmull-Clark surfaces cannot exactly represent conic sections, like the spherical and cylindrical shapes for these examples, all control vertices for Catmull-Clark are placed exactly on the analytic surface to approximate it. One exception to this is near symmetry boundaries, where the control points need to line up to define the symmetry conditions. Further, denser meshes cannot be derived by subdivision as this would keep the approximation error of the coarsest mesh, as seen in the previous example in Section 3.2.4. Therefore, each denser control mesh is constructed such that every control point is sampled directly from the conic. Another point to note is that Catmull-Clark surfaces do not support true point forces, because interpolating control points cannot be defined without special rules.

3.2.5.1 *Scordelis-Lo Roof*

The first example of the shell obstacle course is a roof shape defined by a segment of a cylinder under gravitational load. The initial surface as well as the deformed result are shown in Figure 3.23. The displacements of the result have been scaled for the deformed surface by a factor of 10 to make the deformation visible. Though not required for this example, symmetries were used to reduce the time needed for the simulation by analyzing only one quarter, as indicated by the area colored in blue.

The cylindrical geometry of the surface has length $L = 50$ with a radius of $R = 25$. The surface of the roof is restricted to a segment of the cylinder with an opening angle of $\alpha = 80°$. The roof is constrained by two rigid diaphragms at the curved boundaries of the surface. For the simulated material of the roof a thickness $t = 0.25$ is assumed with Young's modulus $E = 4.32 \cdot 10^8$ and Poisson's ratio $\nu = 0.0$. The gravitational load on the surface is $p = 90.0$ per unit area.

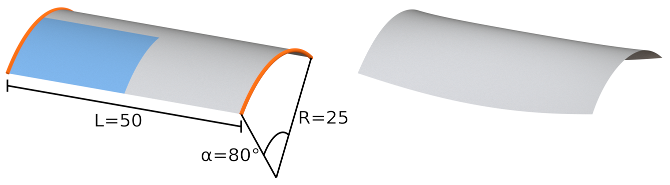


Figure 3.23: Setup and deformed result for the Scordelis-Lo Roof problem.

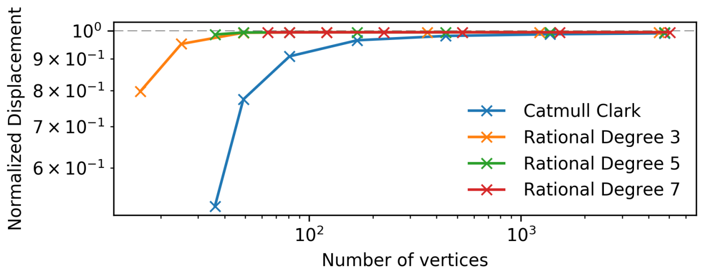


Figure 3.24: Convergence of results for the Scordelis-Lo Roof problem with different types of surfaces.

To evaluate the convergence of the implementation using NURBS compatible subdivision surfaces, several meshes with increasing number of vertices and different degrees are simulated. The coarsest surface for each degree is represented by just a single rational element which defines the cylinder geometry exactly. Denser meshes are obtained via subdivision of the initial mesh. For the Catmull-Clark results, the coarsest mesh needs already 9 elements to approximate the cylinder and to define the boundary.

The simulation results of the different variants are presented in Figure 3.24. All displacements, measured at the center of the initially straight edge of the roof, have been normalized by the reference value of 0.3024 provided in [11]. Note that both axis of the plot are in log scale to better highlight the differences of the different results. All rational surface variants using NURBS compatible subdivision quickly converge to a value close to the reference result. The actual converged value is slightly lower than the reference, but similar to results by Kiendl *et al.* [52]. The results using higher degree surfaces are already close to the converged value with just a single element.

In contrast, the results for the Catmull-Clark surfaces show a slower convergence. There are two main reasons for this. First, Catmull-Clark surfaces cannot represent conic sections, like cylinders, so there is some geometric error already in the surface representation. And second, the boundary handling using ghost vertices is not ideal to enforce the symmetry conditions, as already observed in the results for the clamped plate problem in Figure 3.12. The improved boundary

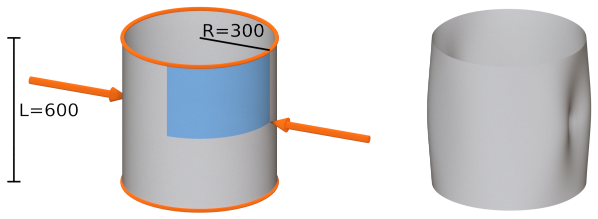


Figure 3.25: Setup and deformed result for the Pinched Cylinder problem.

constraints by Green [44] could improve convergence for Catmull-Clark, as well as simulating the full roof, without symmetries. But the first approach requires adding many additional unknowns to the system, one for each vertex on the boundary, and simulating the full roof would require approximately four times the number of degrees of freedom.

3.2.5.2 *Pinched Cylinder*

The second problem is a cylinder subject to two opposing point forces on the middle of its length with rigid diaphragms at the boundaries. Figure 3.25 shows the initial geometry of the problem and the final solution, where the displacement has been scaled to make it visible. To correctly apply the point loads and to speed up analysis, symmetries are used and only one eight of the cylinder, the blue region in Figure 3.25, is simulated.

The cylinder with length $L = 600.0$ and radius $R = 300.0$ has an assumed thickness of $t = 3.0$. The material is defined with Young's modulus $E = 3.0 \cdot 10^6$ and Poisson's ratio $\nu = 0.3$. The applied forces have a magnitude of $F = 1.0$.

As for the Scordelis-Lo roof example, the coarsest surface using NURBS compatible subdivision is defined by a single element and subdivision is used to generate denser control meshes for the same geometry. Also similar to the previous example, each Catmull-Clark control mesh is sampled from an exact cylinder to approximate the analytical surface. Again, the coarsest Catmull-Clark surface consists of 9 elements to approximate the cylinder and to define the boundaries.

Figure 3.26 compares the displacement results of the different surface variants normalized by the reference displacement of $1.8248 \cdot 10^{-5}$ at the points where the forces are applied, as given in [11]. These results also show that higher degree surfaces converge significantly faster. Degree 7 results are already close to the reference with just one

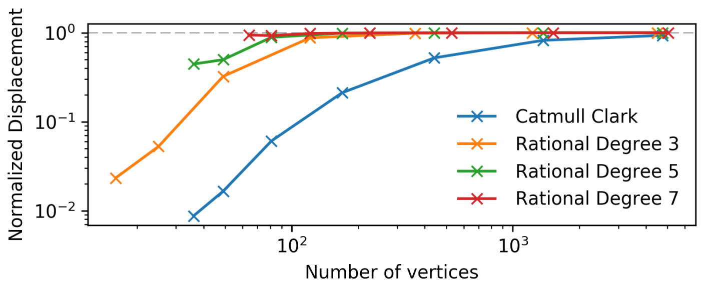


Figure 3.26: Convergence of results for the Pinched Cylinder problem with different types of surfaces.

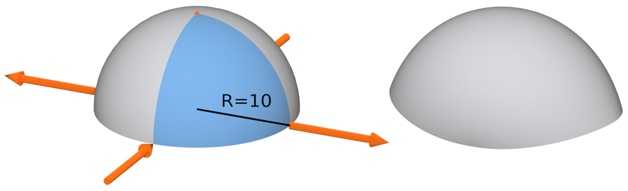


Figure 3.27: Setup and deformed result for the Hemisphere problem.

element. Again, the Catmull-Clark solution convergences slower than the comparable non-uniform, rational surfaces of degree 3.

3.2.5.3 *Hemisphere*

The last problem of the shell obstacle course is a hemispherical shell subject to four point loads on its edge, as shown in Figure 3.27. Only one quarter of the hemisphere with added symmetry conditions is simulated to correctly apply the point loads. The center point on top of the hemisphere is constraint to prevent rigid body movement.

The radius of the hemisphere is $R = 10.0$. For the material, a thickness of $t = 0.04$ is used with Young's modulus $E = 6.825 \cdot 10^7$ and Poisson's ratio $\nu = 0.3$. All applied loads have a magnitude of $F = 2.0$.

Similar to the other problems in the obstacle course, the initial NURBS compatible subdivision surface is defined by a single rational patch and refined using subdivision to add more degrees of freedom. The region around the center point on top of the hemisphere is modeled using degenerated quads. Because only one quarter of the hemisphere is modeled, and because extraordinary vertices are not supported on (symmetry) boundaries with multiple knots, using an extraordinary vertex for the center point is not possible in this case. All meshes for the Catmull-Clark variant are again sampled from the ex-

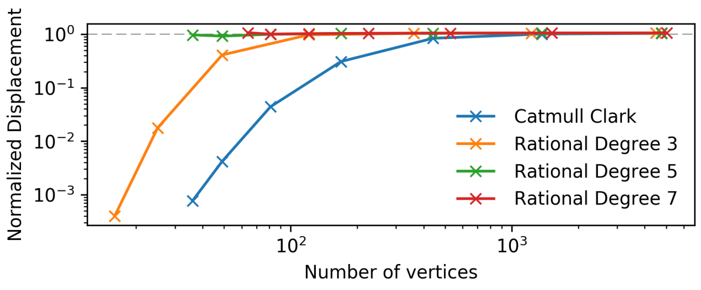


Figure 3.28: Convergence of results for the Hemisphere problem with different types of surfaces.

act hemisphere, except for vertices needed to define the symmetry conditions.

The results of all variants in Figure 3.28 are compared to the reference displacement of 0.0924 provided in [11]. The displacement is again measured at the points where the forces are applied. Similar to the other results in this obstacle course, higher degree surfaces perform significantly better and NURBS compatible subdivision surfaces of degree 3 compare favorable to Catmull-Clark.

3.2.6 *Arbitrary Geometry*

The example problems demonstrated so far all use simple geometries, like planar, cylindrical or spherical shapes. Only for such simple shapes an analytic reference solution can be computed in some cases. As a result, all of these shapes can be represented exactly by a single NURBS surface, without the need for patching.

In contrast, real-world problems can seldom be described by such simple geometries. Typically, a complex CAD design consists of several NURBS patches which need to be manually aligned by the designer to provide the necessary continuity. During simulation, the joins between these patches also need to maintain their continuity to prevent sharp bends or seams. For isogeometric analysis of NURBS surfaces, this can be achieved for example with the techniques described in [53] or [24].

Using subdivision surfaces, complex surface topologies can be easily designed and analysed without manually maintaining patches and their continuity. For example, the chair model in Figure 3.29b, inspired by designs of *Zaha Hadid Architects*, can be defined by a single subdivision control mesh, shown in Figure 3.29a.

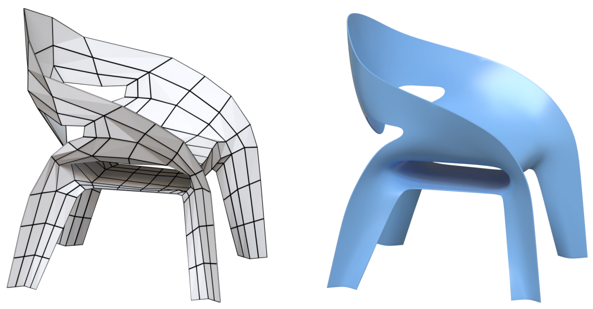


Figure 3.29: Even complex designs like this chair can be modeled with a single subdivision control mesh (a), which defines a smooth limit surface (b).

To verify the accuracy of isogeometric analysis on such designs, the analysis results need to be compared to traditional FEM software, because there is no reference solution available. In the following, isogeometric analysis results for the chair model from Figure 3.29 are compared to results computed with the commercial FEM software *Abaqus*.

In a traditional FEM workflow, typically a NURBS-based CAD model is loaded into the FE software and meshed into a FE mesh for analysis. This preprocessing step is often a time consuming semi automatic process which can take up to 80% of the total analysis time [30]. Creating a good FE mesh requires experience from the analyst as fully automatic meshing procedures may define inappropriate or bad quality elements.

For the IGA-based simulation, the designed subdivision surface of degree 3 can be used directly for analysis. However, to provide enough DOF to represent an accurate solution, the initial control mesh is subdivided once for analysis.

To analyse a subdivision surface in Abaqus, it needs to be exported to a NURBS-based representation, as Abaqus cannot process subdivision surfaces. The export to NURBS is performed with the adapted PCCM patching algorithm presented in Section 3.1.3.1. Each face of the subdivision control mesh is exported to an individual NURBS patch, maintaining at least C^1 continuity between patches. The collection of NURBS patches can then be loaded into Abaqus, where the surface can be meshed into a FE mesh.

The exported NURBS-based representation of the chair model consists of 280 individual patches, shown in Figure 3.30a. In Abaqus,

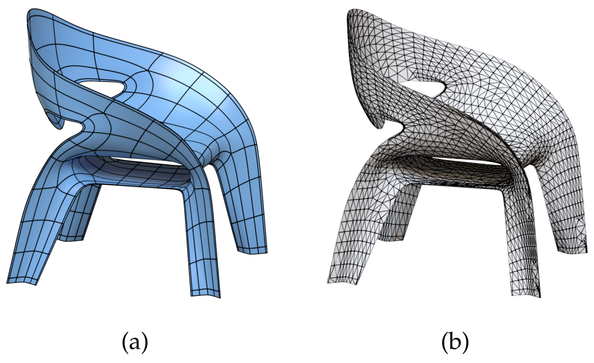


Figure 3.30: Individual NURBS patches (a) are meshed into a FE triangle mesh for use in Abaqus (b).

the resulting surface is meshed into the triangle FE mesh shown in Figure 3.30b. This triangle mesh is then used for thin shell analysis in Abaqus using the STRI3 element type, which implements the Kirchhoff-Love theory of thin shells with linear triangular elements.

For both analysis methods, IGA and Abaqus, the same material settings and boundary conditions are applied. The assumed material of the chair has a thickness of $t = 0.005\,m$ with Young's modulus $E = 3 \cdot 10^9\,N/m^2$ and Poisson's ratio $\nu = 0.35$. The position of three legs of the chair is fixed on the ground. On the front corner of the fourth leg, an upwards point force of $200\,N$ is applied. Additionally, a gravitational body force of $-10290\,N/m^3$ for the self weight of the material is applied.

In Figure 3.31 the deformed results for both methods are compared. For visualization, the displacements have been scaled by a factor of 5. The surfaces are colored based on the magnitude of the deformation. The result from Abaqus in Figure 3.31a is visually identical to the IGA result in Figure 3.31b using the designed NURBS compatible subdivision surface of degree 3 directly. The maximum displacements of these results are $0.0343\,m$ for Abaqus and $0.0358\,m$ for IGA.

These results show that the presented IGA approach based on NURBS compatible subdivision surfaces leads to accurate results which are in accordance with commercial FEM software. But using IGA, no intermediate export and meshing steps are necessary.

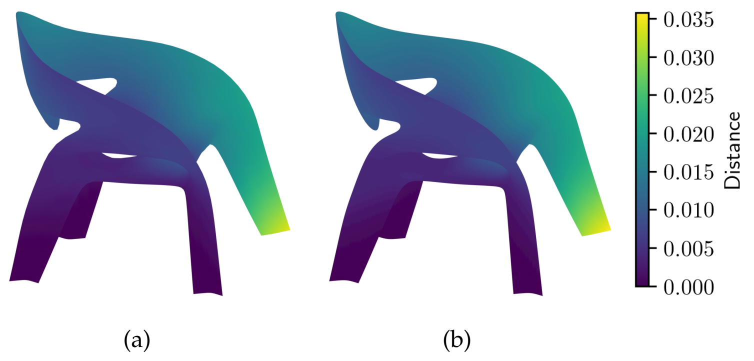


Figure 3.31: The deformation, scaled by a factor of 5 for visualization, computed with the commercial software Abaqus (a) is visually identical to the IGA result using the designed subdivision surface directly (b), demonstrating the high accuracy of subdivision-based IGA.

3.3 ARTIFACTS

Having design and IGA integrated into a single environment has many advantages and enables new uses cases, as further discussed in the next chapters. However, now the CAD representation has to satisfy not only aesthetic requirements for design, but also the requirements for analysis. This is an important aspect the designer using such a system must be aware of.

The CAD representation used for IGA affects analysis much in the same way the quality of the simulation mesh affects the reliability of traditional FEA. What has been termed *analysis-aware modeling* by Cohen *et al.* [28] requires an understanding of how different modeling choices and features of a CAD representation may influence analysis.

In this section limitations of CAD representations that may lead to artifacts in the analysis results are discussed. The designer cannot take into account all sources of artifacts, as this would require knowing the analysis results upfront. Often it is only experience of what is expected from analysis which enables the analyst to define the mesh layout which leads to accurate simulation results. However, keeping limitations of the CAD representations in mind while creating the design can improve the accuracy of analysis and possibly avoid misinterpretations of analysis results.

In the following, three pitfalls that may be encountered in IGA based on any B-spline-based surface representation, e.g. NURBS, T-spline or subdivision surfaces, are briefly discussed.

3.3.1 *Degrees of Freedom*

A difference in the requirements for design and for analysis is the amount and distribution of DOF to describe the geometry. For design, it is common to only specify the smallest amount of DOF required to exactly represent the geometry. These DOF are placed adaptively as required. Large, low-frequency regions only need a few DOF, while high-frequency details may need many DOF. In contrast, an analysis result can only be as accurate as the available DOF allow it to be, i.e. there must be enough DOF to represent an accurate solution. The result of a simulation may require DOF which are not required to define the design. For example, a flat plate can be defined with just a few control points, but a deformed plate, after applying forces to it, requires more control points to define its new deformed shape.

Therefore, although even the coarsest control mesh can be used for analysis because it describes the smooth geometry exactly, the CAD representation has to offer enough DOF to be able to represent high-frequency components of the analysis result. Using subdivision surfaces as a basis for IGA the number of DOF can be increased easily by subdividing the geometry (h-refinement) to create different levels of detail.

Having more DOF to define the geometry also enables the designer to include small geometric details in the surface definition which are not part of an initial coarse design. These small geometric details will also influence analysis results.

Figure 3.32 highlights the differences in results and problems that can arise from different levels of detail and additional geometric details. The coarse mesh on the left results in a large deformation, because it has too few DOF to accurately represent the solution for the given forces and constraints. The subdivided control mesh in the center, describing exactly the same surface, results in a small, but visible deformation. However, the subdivided mesh with added details on the right, having the same number of DOF, results in almost no displacement at all. This is because the slightly rippled surface is much stiffer than the perfectly flat surface. While typically the differences are not as extreme as in this example, it is important to consider the amount of DOF and the influence of geometric details on the structural stability of surfaces.

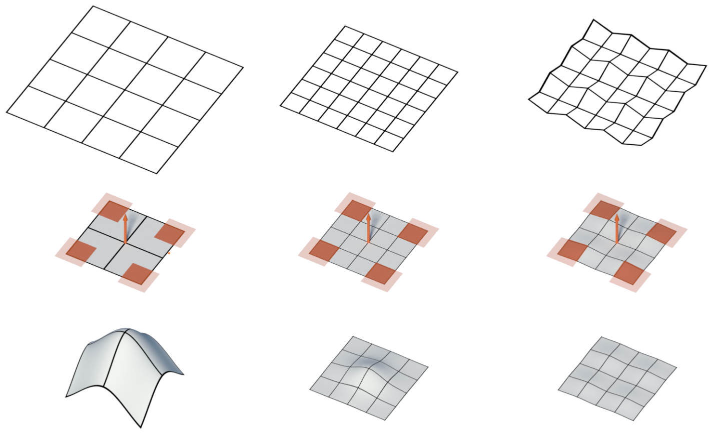


Figure 3.32: Simulation of different levels of detail of a subdivision surface can lead to very different results. The top row shows three control meshes defining similar surfaces, from left to right: the initial coarse control mesh, a subdivided control mesh derived from the coarse control mesh, and a subdivided control mesh with added details not present in the coarse mesh. The corresponding surfaces are shown in the center row, together with forces and constraints highlighted in orange. The corners are fixed in place while a force is applied at the center of each surface. The results in the bottom row, computed using the same forces and constraints for each surface, are very different. Either due to different DOF (left, center), where more DOF improve the accuracy of the result. Or due to different surface geometry with different stiffness properties (center, right).

3.3.2 *Mesh Orientation*

An important aspect in modeling free-form surfaces is the careful alignment of geometric features with the grid lines of the control mesh. Using B-splines, any feature oriented skew to the grid lines of the control mesh may exhibit artifacts in form of surface rippling along the feature [5, 6, 76–78].

When using the CAD geometry for analysis it is important to be aware that this artifact will be included in the analysis of the surface and thus lead to flaws in the results of the analysis. The designer's choice of how to orient the parameter domain for modeling purposes may lead to different simulation results, since the underlying geometry will have changed.

Figure 3.33 shows the problem definition of a circular plate subject to uniform force. The orange parts of the plate have been constrained and cannot respond, while other parts of the surface respond to the load by bending downward. A vertical feature appears through the

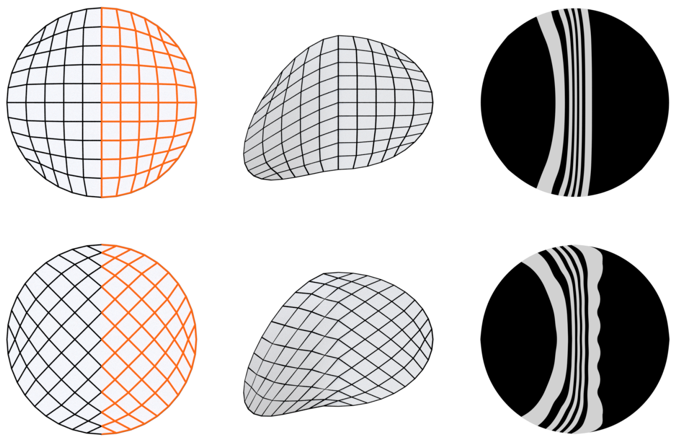


Figure 3.33: Unaligned features can cause surface artifacts in simulation results. Half of the control meshes on the left is constraint, the rest is subject to a uniform force. The deformed results are shown at an angle in the center. The feature line caused by the deformation runs along the grid lines at the top, but skew to the grid at the bottom example. Ripples are observed in the latter case due to the limitations of the CAD representation, as observed in the visualization of reflection lines on the deformed surfaces on the right.

centre of the plate. The problem formulation is the same on top and bottom, but the representation of the surface is rotated by 45 degrees. The feature runs along the grid lines of the representation in the top example, and the analysis result presents a well defined feature. The resulting feature runs diagonal to the grid lines in the bottom example and the artifact appears as ripples in the analysis results, clearly visible when looking at reflection lines corresponding to the deformed surface.

3.3.3 *Rational Expressions*

Next to considering the mesh layout, it is also important to choose the correct surface representation to define the intended design. If a CAD representation does not support rational expressions conic sections can only be approximated. The behavior of thin shells in particular is strongly dependent their geometry.

Figure 3.34 highlights the importance of an accurate description of the geometry for a problem that has already been discussed in Section 3.2.4. Figure 3.34a shows the control mesh and parameterization of a cylindrical pipe subjected to internal pressure. This pipe is modeled without using a rational representation and thus only approx-

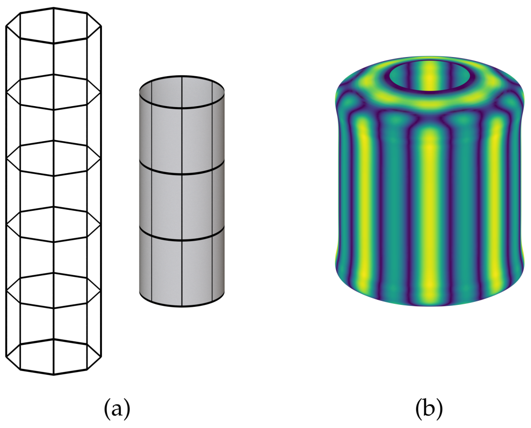


Figure 3.34: Not using a rational representation to define conic sections can lead to artifacts in the analysis results. The non-rational control mesh and parameterization in (a) approximate a cylindrical pipe subject to internal pressure. The analysis result in (b), colored by error to the analytical reference solution, shows an artifact corresponding to the initial non-rational control mesh, even though the mesh has been subdivided three times for analysis.

imates an exact cylinder. Figure 3.34b highlights the error for IGA using a control mesh based on this geometry that has been subdivided three times to create a dense mesh for which an accurate result could be expected. However, the result shows an uneven distribution of the deformation due to the CAD shape not being truly circular. The frequency at which the lateral artefact varies reflects the number of control points chosen along the latitude of the initial control mesh. This is because subdividing the control mesh does not change the limit surface, so the initial approximation error is kept.

As demonstrated in Section 3.2.4, using a rational representation for the same problem avoids these artifacts. A workaround for simple cases using non-rational surfaces is to sample the control points of denser meshes from the exact surface, as was done in Section 3.2.4, instead of subdividing a coarse mesh. This allows to define denser meshes which better approximate the desired geometry, thus reducing the error. Another workaround is increasing the degree to create a better approximation of the desired geometry and to improve analysis results.

3.4 summary

Isogeometric analysis based on the NURBS compatible subdivision scheme combines the advantages of subdivision-based implementations and NURBS based analysis. The NURBS compatible subdivision scheme is therefore a suitable geometry representation to satisfy the requirements of Research question 1.

The support for non-uniform knot vectors improves handling of boundaries and symmetries compared to implementations based on other subdivision schemes, and is fully compatible to NURBS-based analysis. At the same time, the NURBS compatible subdivision scheme allows to analyse arbitrary topology surfaces, unlike analysis based on NURBS.

Irregular regions around extraordinary vertices can be evaluated using multiple techniques presented in this chapter. Depending on the use case, these methods have different advantages, providing either exact geometry or exact derivatives.

To improve accuracy, especially in regions near extraordinary vertices, a simple adaptive numerical integration scheme can be used. This is also beneficial for rational surfaces and regions with non-uniform knot vectors. Additionally, for flat surfaces, the patching algorithm presented in this chapter provides exact derivatives everywhere on the surface and therefore allows IGA based on the NURBS compatible subdivision scheme to exactly pass the patch test.

The NURBS compatible subdivision scheme also simplifies the treatment of certain constraints, i.e. at the boundary, while allowing to define all forces and constraints possible with either subdivision surfaces or NURBS. Unlike other subdivision-based IGA methods, NURBS compatible subdivision surfaces allow to specify exact point forces, acting only on a single subdivision control point.

NURBS compatible subdivision surfaces have been successfully used to solve Poisson and thin shell problems with high accuracy and it has therefore been further confirmed that it is a suitable surface representation to answer Research question 1.

The subdivision representation enables the definition of arbitrary topology geometry as a single surface. This enables analysis of complex domains without the special treatment required for patch-based approaches, e.g. as in [53] or [24]. On the other hand, all advantages of NURBS-based approaches over traditional subdivision surfaces have also been demonstrated for this subdivision-based representation:

- a rational representation to exactly define conic sections,
- higher degree surfaces for improved convergence, and
- non-uniform parameterization to represent boundaries.

However, using IGA with surfaces intended for design poses several difficulties. A few common pitfalls designers need to be aware of were discussed. Knowing these helps interpreting unexpected simulation results and how to avoid common problems.

4 INTEGRATED DESIGN AND ANALYSIS

This chapter is based on research that has been published in the following conference papers and articles:

A. Riffnaller-Schiefer, U. H. Augsdörfer, and D. W. Fellner, "Isogeometric analysis for modelling and design," in *EG 2015 - Short Papers*, B. Bickel and T. Ritschel, Eds., The Eurographics Association, 2015, pp. 17–20

U. H. Augsdörfer and A. Riffnaller-Schiefer, "On the convergence of modeling and simulation," *IEEE Computer Graphics and Applications*, vol. 37, no. 4, pp. 8–13, 2017, ISSN: 0272-1716

A. Riffnaller-Schiefer, U. H. Augsdörfer, and D. W. Fellner, "Interactive physics-based deformation for virtual worlds," in *2017 International Conference on Cyberworlds (CW)*, 2017, pp. 88–95

A. Riffnaller-Schiefer, U. H. Augsdörfer, and D. W. Fellner, "Physics-based deformation of subdivision surfaces for shared virtual worlds," *Computers & Graphics*, vol. 71, pp. 66 –76, 2018, ISSN: 0097-8493

Using the same geometry representation for both design and analysis allows direct analysis of CAD geometries, without meshing surfaces into a *simulation mesh*. This already leads to huge timesavings compared to classical FEM. However, in most cases, the analysis functionality is separate from the tools used to create the designs. So even though IGA allows to use the same geometry representation, there are still two different environments for design and analysis. The next natural step is to merge design and analysis into a single environment, providing designers direct access to analysis features.

Having integrated isogeometric thin shell analysis directly in a design software allows to perform typical analysis tasks like computing deformations and visualization of stresses directly within the design environment. Designers can get immediate feedback on the physical plausibility of the design already during early design phases of product design. Without leaving the modeling application the designer can iterate through different design concepts and use the analysis feedback to *guide* the design.

This chapter explores the integration of isogeometric analysis into a modeling environment, combining design and analysis. This cannot only be used for structural analysis within the design environment, but also provides the user with new instruments for design beyond shape analysis.

4.1 FEEDBACK ON GEOMETRIC PROPERTIES

Being able to evaluate and integrate arbitrary functions on the designed limit surface simplifies the implementation of many common surface analysis tasks required in design software.

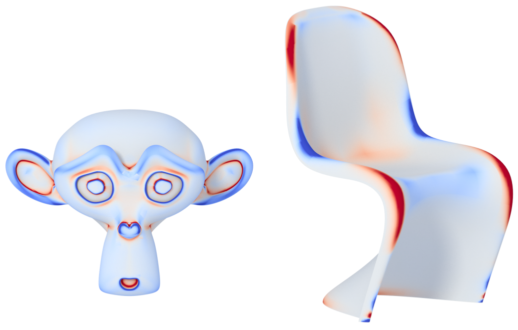
A common task in CAD is the evaluation of the surface area of the designed geometry, for example to estimate the amount of material required for fabrication. This feature is readily available in most parametric or NURBS-based CAD and engineering software, but is missing in most modeling applications based on subdivision surfaces. Even if the area of freeform surfaces can be evaluated, it is often only approximated based on the surface area of a subdivided polygonal mesh.

With isogeometric analysis functionality inside modeling applications, arbitrary functions may be integrated on the designed limit surface and e.g. surface areas can be computed without creating a densely subdivided polygonal mesh.

Similarly, other surface quantities requiring derivative information, like for example different curvature measures, can also be evaluated with high accuracy on the designed limit surface. Figure 4.1 shows the visualization of curvatures, evaluated using the IGA framework, directly on subdivision surfaces.

Additionally, the designer can assign different material properties to the designed shape to evaluate physical quantities like the total mass of a surface with an assumed thickness.

Such quantities are currently not available in general purpose software for modeling subdivision surfaces, but can be easily added to an integrated system based on IGA. Because design for fabrication increasingly involves freeform surfaces, such measures and tools to assist the designer also become more important.



(a) Mean curvature (b) Gaussian curvature

Figure 4.1: Surface properties like curvature can be efficiently computed and visualized on subdivision surfaces using IGA tools.

4.2 DIRECT ANALYSIS

Testing a designed model is an important part of product design. Often, structural analysis to check the products integrity for various environmental forces is only performed after the design stage. In case problems are found, designers need to re-design to fit the requirements, trying to prevent the problems. To safe time and shorten the design-analysis cycle, designs are therefore often overly conservative.

To achieve a fast product development cycle, problems must be detected and handled as early as possible [1]. Therefore, to provide feedback to the designer already during the early prototyping phase, IGA-based analysis can be directly integrated into the modeling environment, enabling immediate feedback for the designer.

To facilitate integrated analysis, the user interface (UI) of the design application needs to provide several analysis related tools. First, the user needs to be able to assign physical material properties to the designed surfaces. The material determines the behavior of the surface in response to environmental impact. For the thin shell simulation, an isotropic material is defined by three values: the thickness t of the material, its Young's modulus E and its Poisson's ratio ν. For example, to simulate a sheet of aluminum with a thickness of 1mm, the material is defined by

$$t = 0.001\text{m} \quad E = 6.9 \cdot 10^{10}\text{N}/\text{m}^2 \quad \nu = 0.32$$

To simulate its environment in the real world, forces can be applied to the designed object. There can be various different types of forces,

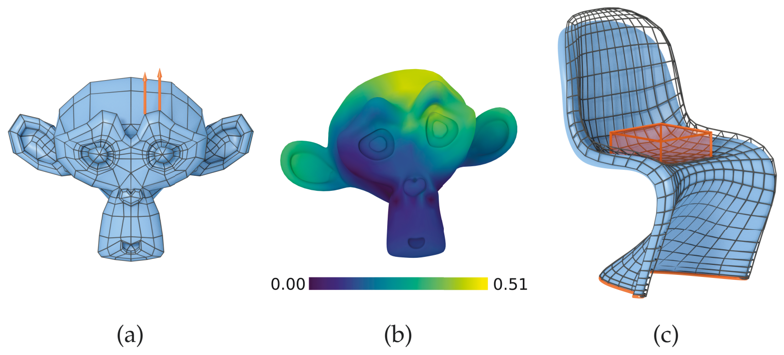


Figure 4.2: Two point forces pulling two control points upward have been applied to the monkey subdivision surface in (a), constraints on the back fix it in place. The resulting deformation is applied and visualized by coloring the surface based on the absolute distance of the displacement (b). The chair in (c) is constraint at the bottom and loaded with a uniform pressure, which is restricted to the region within the orange proxy geometry. The deformed surface is shown together with the initial control mesh to highlight the deformation.

as discussed in Section 3.1.5, which need to be made available in the UI provided within the modeling software. For example, forces can be applied directly to a control point of the subdivision surface to simulate a localized force, as shown in Figure 4.2a. Larger area forces like pressure can be integrated over some region of the subdivision surface, which can be defined for example using proxy geometry as in Figure 4.2c. And global body forces like gravity can be defined for the whole object.

To actually perform a simulation, the boundary conditions of the problem need to be specified, i.e. some constraints need to be applied. In three dimensions, at least six DOF need to be constraint to restrict translation and rotation on each axis and therefore prevent any rigid body movement. Again, there can be various different types of constraints, as discussed in more detail in Section 3.1.5. A common requirement for simulations is to fix some part of the surface in place, so that it does not move. This is achieved by restricting the displacements at interpolating control points, commonly used at the boundary, to zero. Figure 4.2c shows an example of constraining the boundary in this way. Note that using subdivision surfaces, fixing a single non-interpolating control point does not fix the limit surface in place. A convenient way to directly restrict the limit surface is to use Lagrange multiplier constraints, also discussed in Section 3.1.5. They allow the user to prescribe the displacement of any point on the limit surface directly.

After the user specified materials, forces and constraints, the thin shell simulation can be performed. The result of the isogeometric simulation is the displacement field for the subdivision surface, using the same control points as the initial surface as DOFs.

Different simulation results require different visualization methods. Some information, like displacement, can be directly applied to the geometry, while others, like stresses, are best visualized by coloring the surface. Applying the resulting deformation directly to the control points of the subdivision surface allows the designer to easily switch, or even animate, between deformed and undeformed configurations. Displacements can also be color coded and visualized on the initial or deformed surface. This lets the user quickly identify areas of the surface which deform the most. Both of these visualizations are combined in Figure 4.2b, while Figure 4.2c shows just the deformed surface together with the initial control mesh. Stresses, for example the von Mises stress, are computed from the displacements in a post-processing step. The von Mises stress is a scalar field which is related to the yield strength of a material. It is therefore an important indicator for physical durability, allowing the designer to immediately identify critical regions. The designer can respond by changing the design to reduce stresses in order to increase the structural stability of the design.

4.3 SHAPE OPTIMIZATION

Instead of simply providing visual feedback of stresses on the design the system may offer the designer improved versions of the intended design.

In classical finite element analysis, the shape optimization result is a finite element mesh, i.e. a dense triangle or quad mesh, which is unsuitable for design purposes. Therefore, to introduce the optimized design back into the product development cycle, the CAD design has to be recreated from the optimized simulation result. Interfacing models seriously limits the scope of state-of-the-art approaches in shape optimization.

Because in IGA the degrees of freedom are the same for design and analysis, optimization of the CAD geometry can be performed directly on the control points of a subdivision surface. An IGA-based optimization can therefore directly use the DOF of a subdivision surface as the *design variables* which are updated during the optimization process. It is also possible to specify different design variables for

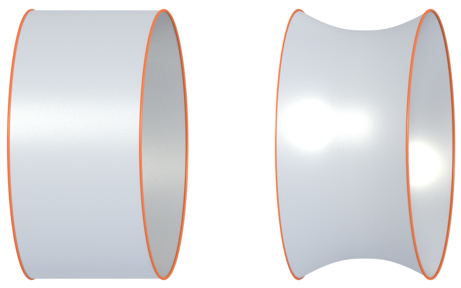


Figure 4.3: Minimization of the surface area of a cylinder with fixed boundaries. The optimization result is a catenoid like minimal surface.

the optimization, independent of the DOF of the subdivision surface, which somehow map to the control points of the subdivision surface. For example, for symmetric surfaces, multiple control points can be grouped into a single design variable for the optimization, to exploit symmetry and improve the performance of the optimization. Also, parameters of parametric models can be used as design variables for optimization. For example, the radius of a parametric cylinder model may be used as DOF for the optimization, affecting the position of all control points in the mesh.

Regardless of how DOF are assigned for optimization, an important property of isogeometric optimization is that the optimization result can be directly used for the design again. This allows the designer to use optimization in his design workflow in exactly the same way as any other modeling tool.

Possible applications of optimization in product design include structural optimization or minimization of surface area or volume. Structural optimization aims to improve stability of an object in response to a predefined force with certain boundary conditions. Minimization of surface area or volume aims to save material for fabrication. Figure 4.3 shows the minimization of the surface area of a cylinder, where the DOF for the optimization are the radii of the inner subdivision control points. The radius at the boundary is fixed. The resulting surface minimizes the surface area while maintaining the designed shape at the boundaries.

4.4 MODELING MODES

The same features used to compute deformations for analysis and optimization can also be used to create new modeling modes. This provides the designer with physics-based modeling tools in addition to the existing CAD tools.

There are already many existing techniques for physics-based modeling [16, 64] and some of them can also be adapted to work with isogeometric analysis. In the following, two types of physics-based design tools are presented for modeling using the isogeometric thin shell simulation: constraint-based and force-based modeling. These modeling modes can be used in combination to create physics-based deformations that are difficult to model manually.

By defining prescribed displacements for certain control points or certain points on the limit surface, as explained in Section 3.1.5, a deformed surface can be created. Performing a thin shell simulation with these constraints in place ensures that the prescribed displacements are applied and additionally deforms the unconstrained regions of the surface according to its material properties. This allows to create complex deformations with only a few constraints.

In addition to manually defining constraints at certain control points of the subdivision surface, more intuitive user interfaces to specify constraints can be provided.

For example, prescribed displacements can be interactively specified using a drag and drop interface. The user just moves the control points to be constraint to their desired position, the unconstrained parts of the surface immediately deform to satisfy the constraints based on a linear simulation. Figure 4.4 demonstrates the process of interactively deforming a cactus mesh by just dragging some of the control points to their new position to which they are constraint.

The same drag and drop interface can also be used directly for the limit surface. The user picks a point on the subdivision limit surface and drags it to the desired position. To satisfy this constraint of the limit surface, a Lagrange multiplier constraint is added to the system of equations.

Another way to intuitively define constraints are sketch-based techniques, which have already been used in various ways for modeling [29, 66]. Here, the designer is able to sketch three-dimensional curves next to an existing subdivision surface. Control points close to the sketched curve are then constraint to the curve, while unconstrained parts are deformed to satisfy the minimal energy condition. This leads to a smooth, naturally deformed surface approximating the sketched curves.

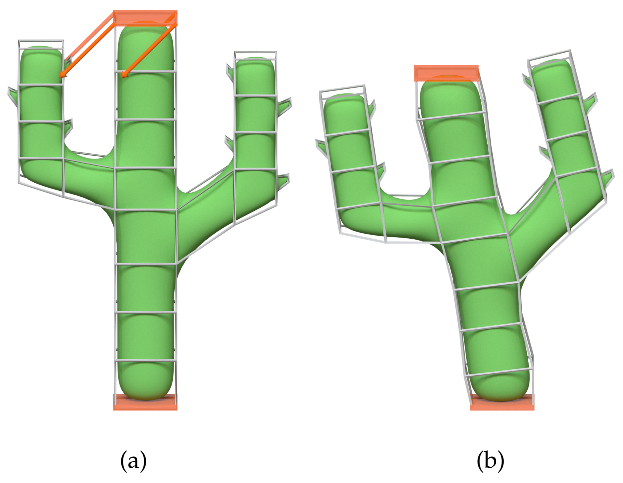


Figure 4.4: Interactively changing the overall shape of a cactus model by moving only four control points. In (a) the control points within the orange highlighted area are constraint by the designer. Control points at the bottom are fixed in place while constraints at the top are defined by dragging the selected control points to their new desired position. The result in (b) is a deformed mesh which satisfies these constraints with minimal energy. Small geometric details on the cactus follow the overall deformation as intended by the designer.

The creation of constraints for the subdivision control vertices from the sketches is done in a two step process. First, the sketch curves are discretized into sequences of points s_i defining line segments. For each point s_i, let the closest control vertex of the subdivision surface be c_j. Each control vertex c_j has an associated set T_j of potential constraint positions to which s_i is added in this step. Afterwards, offset vectors are computed from each control vertex c_i to all potential constraint positions $s_j \in T_i$. The offset vector with the shortest magnitude is applied as a displacement constraint for each c_i.

This two step processes ensures that only one subdivision control point can snap to any given position on the sketch curves, preventing unwanted contractions and folds in the surface. Also, control vertices which are not close to the sketch curves do not get any constraints assigned.

Instead of adding constraints to any control vertex close to the sketches, this assignment can also be limited to a selection of control vertices. This allows the designer to specifically select the parts of the surface which should snap to the sketch curves. This is achieved by limiting the search to selected control vertices c_j in the first step.

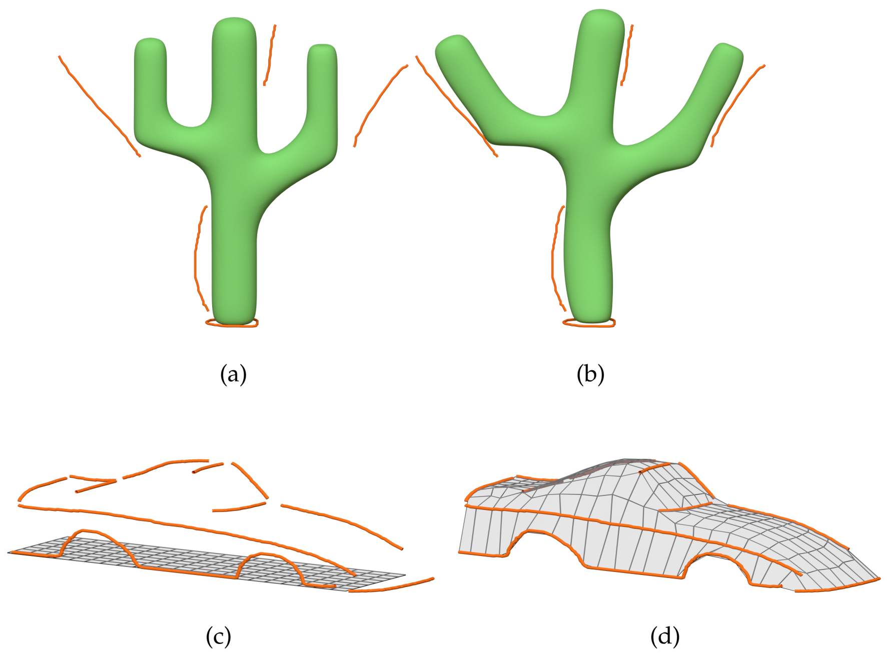


Figure 4.5: Sketch-based posing of figures and a car shape created from a flat plate. Sketched three-dimensional curves drawn by the user (a, c) define constraints on nearby control points to create deformed surfaces (b, d) using the thin shell simulation.

Possible use cases for this technique are for example to quickly sketch the desired outline for an existing surface, or to pose existing models. It is also a fast alternative to manual placement of control points for creating an initial draft of a shape. Details can then be added later using traditional techniques, maybe after subdividing the control mesh to add more degrees of freedom. Figure 4.5 shows examples created using the described sketching technique.

In addition to constraints, forces can be applied to a surface to create a deformed shape in the same way as explained for analysis in Section 4.2 to simulate the environment of a real world object.

The same types of forces which are used for analysis can also be applied by the designer for modeling purposes, e.g. local forces at control points or pressure. To allow more flexible force definitions, the applied forces can also be scripted using custom force functions. Custom force functions are short scripts which are evaluated during integration of the subdivision surface and provide a flexible way to define arbitrary forces. This allows to programmatically setup complex forcing scenarios. A design software may provide a predefined set of force definitions or even allow the user to define arbitrary custom force functions. For example, inflation of an arbitrary closed surface

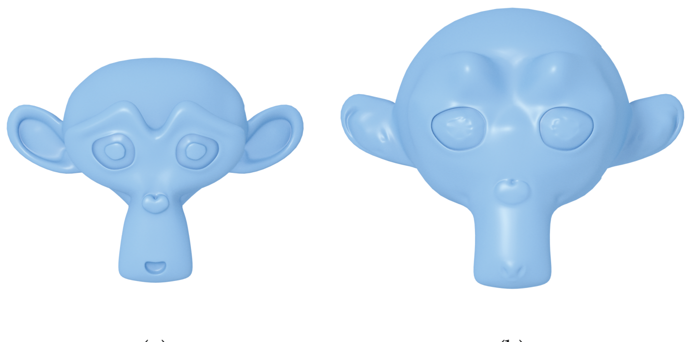


Figure 4.6: A monkey head subdivision surface (a) is inflated by simulating an internal pressure to give it a balloon like look (b).

can be simulated by defining an internal pressure along the normal direction as follows:

```python
def inflate(element, shape_funcs, ref_co, cur_co):
    # get position of control points for current element
    cur_pos = element.get_node_positions(cur_co)
    # compute tangents using first derivatives
    tan = shape_funcs.df.dot(cur_pos)
    # compute surface normal
    n = np.cross(tan[0], tan[1])
    # return pressure along surface normal
    return n / np.linalg.norm(n) * 5e7
```

This function is evaluated for each quadrature point during numerical integration and can return an according force for each. The resulting force gets automatically distributed to the corresponding DOF with basis functions having support on this quadrature point. The result of applying this inflation force function with a non-linear thin shell simulation, e.g. as presented in [26], to create a balloon like look for a complex mesh is shown in Figure 4.6. Creating such an inflated version of the initial surface by manually moving control points is time consuming and hard to get right.

4.5 INTERFACES

To test and integrate the presented techniques in a design workflow, several interfaces have been created to access the isogeometric thin shell simulation. Plugins to several existing modeling applications demonstrate the seamless integration of IGA into a modeling environment to provide different features, ranging from integrated analysis to interactive sketch-based modeling. A web-based interface allows to upload geometry, perform predefined simulations on it, and to view and compare simulation results from any web browser. Finally, a HTTP based API provides access to the simulation for any web enabled device or application.

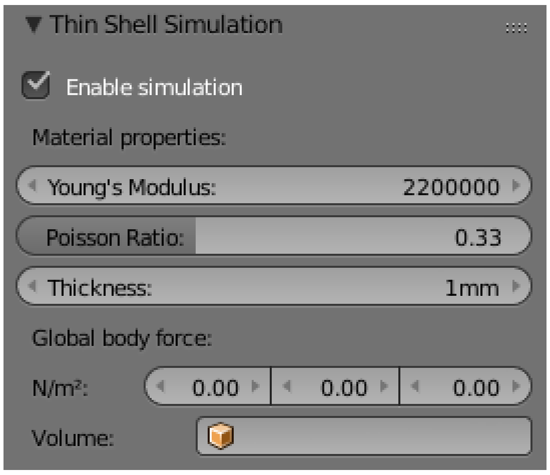
4.5.1 *Blender*

The free and open source 3D content creation software Blender, available at https://www.blender.org, offers many tools for modeling, animating and rendering, among other things. It is used in many areas ranging from modeling for games to animating and rendering complete movies. With the advent of 3D printing it has also become an increasingly popular tool to design three-dimensional objects for manufacturing.

However, not being a typical CAD software, Blender has no built-in support to check the structural integrity of a designed object, or to interface with external engineering software to do so. To add these features, several plugins have been developed to integrate isogeometric thin shell analysis into Blender. As discussed in Section 4.2, this requires additional user interface elements to be able to define materials, constraints, forces, and similar settings.

For design, Blender has built-in support for Catmull-Clark subdivision surfaces. However, to take advantage of higher-degree, non-uniform, rational surfaces in both design and analysis, the NURBS compatible subdivision scheme has been integrated into Blender via an additional custom plugin. This allows to take advantage of both NURBS and subdivision features using a single geometry representation inside Blender. Additionally, the plugin allows to export NURBS compatible subdivision surfaces designed in Blender as standard patch-based NURBS surfaces in the IGES or STEP file formats. This export functionality is based on the technique described in Section 3.1.3.1, and enables Blender to interface with other CAD and engineering software.

Based on the new surface representation, another plugin adds the isogeometric thin shell simulation to Blender. New user interface elements, shown in Figure 4.7, are defined in Blenders object properties panel to let the user assign required material parameters and global body forces to individual surfaces. The body force can also be restricted to the volume of certain proxy geometries, as previously used in Figure 4.2c. Additionally, constraints and forces can be assigned to control points of the surface directly in the 3D viewport, where the



▼ Thin Shell Simulation

✓ Enable simulation

Material properties:

Young's Modulus: 2200000

Poisson Ratio: 0.33

Thickness: 1mm

Global body force:

N/m²: 0.00 0.00 0.00

Volume:

Figure 4.7: Simulation parameters like the material of surfaces or global body forces can be specified directly in Blender using a custom plugin.

surface is designed and visualized. Figure 4.8a shows a subdivision surface in Blenders 3D viewport where several constraints and forces have been applied to control points of the surface. The constraints at the bottom are marked by red squares and forces are indicated by blue lines, visualizing their direction and their relative magnitudes. After simulation, the deformed surface, shown in Figure 4.8b, is immediately shown inside Blender and can be used for example to further change the design or to create an animation.

Instead of adding constraints manually to individual control points, they can also be defined through a sketching interface as discussed in Section 4.4. In Blender, the *Grease Pencil* tool is used to let the user quickly draw 3D sketches directly in the viewport. The Grease Pencil tool uses the current view as the drawing plane for the sketched curves. The depth component of the sketches is defined by the 3D cursor of Blender or by geometry in the scene. After drawing and changing the sketches in the viewport, the user can start the surface deformation by pressing a keyboard shortcut or clicking a menu item. Figure 4.9 shows the Blender user interface where a few sketches have been drawn in the viewport to define a simple slide shape. After constraints based on the sketches have been applied to the initial flat surface (left), the deformed surface is computed using isogeometric analysis. The final smooth subdivision limit surface is shown on the right.

Using sketches, a first draft of the overall shape of a surface can be quickly designed, without moving control points manually. Sketched curves are also easily modified, allowing quick iteration of the de-

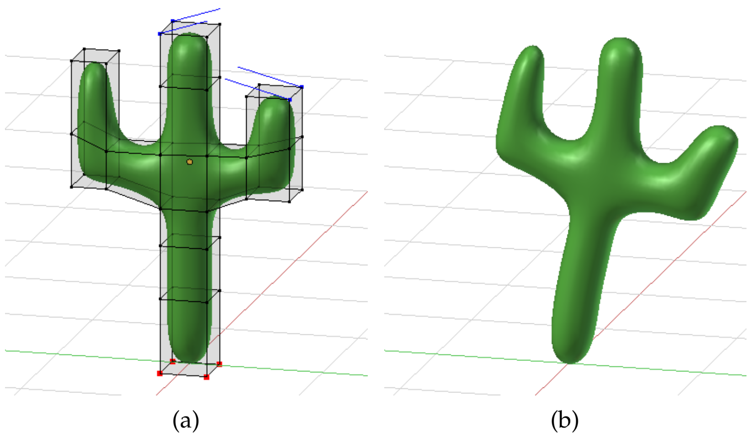


Figure 4.8: The 3D viewport in Blender is used to design and visualize the subdivision surface. Constraints and forces can be directly assigned to control points of the surface and in (a) they are also visualized in the viewport as red squares and blue lines respectively. The simulation result (b) is immediately available inside Blender.

signed shape. The control mesh of the resulting deformed surface can then be used to further improve the design.

4.5.2 *Maya*

Similar to the Blender integration, Komposch [55] integrated tools based on the presented subdivision IGA framework also into *Autodesk Maya*. Maya is one of the leading applications in the entertainment industry, used e.g. to create assets and effects for computer games, tv series and movies.

To explore this interface, the built in Catmull-Clark subdivision surfaces provided in Maya are used as the surface representation for analysis. Catmull-Clark surfaces are a subset of NURBS compatible subdivision surfaces of degree 3, so they can be easily used with the presented IGA platform. The analysis itself is integrated as a set of plugins using the Python API of Maya to the define the user interface, provide tools, and render in the viewport. For example, a similar user interface as in Blender is provided to change the material settings for a surface, shown in Figure 4.10. Here, also a name can be assigned to the physical material for later reference in a collection of defined materials.

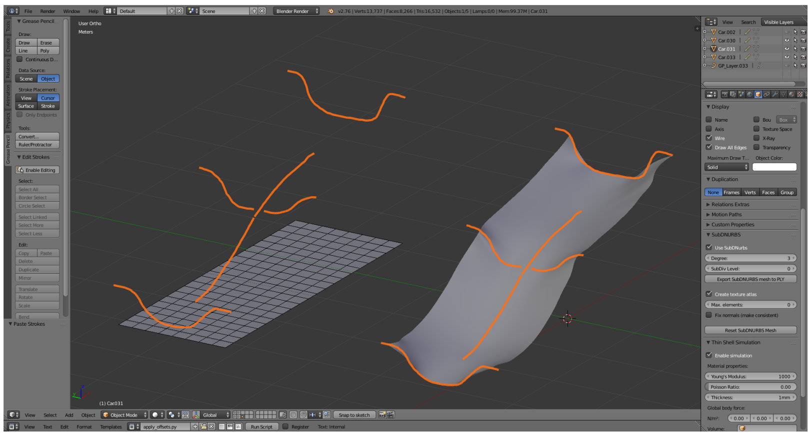
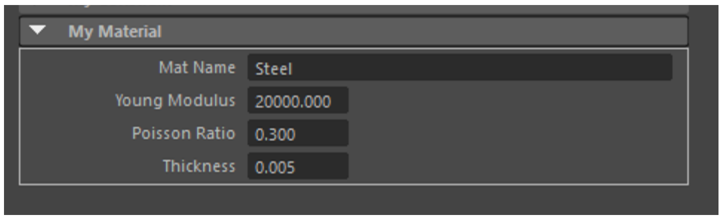


Figure 4.9: Using sketches to apply constraints, a basic slide shape can be created by drawing only a few curves (orange) in Blender. The resulting smooth surface on the right is created by constraining the control mesh of the plate on the left to the sketch lines.



My Material

Mat Name	Steel
Young Modulus	20000.000
Poisson Ratio	0.300
Thickness	0.005

Figure 4.10: Physical material settings of surfaces can be edited within Maya and saved to a library of materials. © *Florian Komposch [55], used with permission.*

The visualization of constraints assigned to individual control points of the subdivision surface changes based on the nature of the constraint. Fixed coordinates of a control point, which are not allowed to move along that coordinate axes during simulation, are visualized as colored triangles, according to the coordinate axes in Maya. On the other hand, constraints prescribing a certain displacement to a control point are drawn as arrows representing the displacement in the viewport. Figure 4.11 illustrates these types of constraints.

The specified constraints and forces are then used to compute a deformed surface. Figure 4.12 demonstrates how this can be used to quickly deform complex objects. Moving individual control points manually would be time consuming and it would be difficult to achieve a physically plausible deformation.

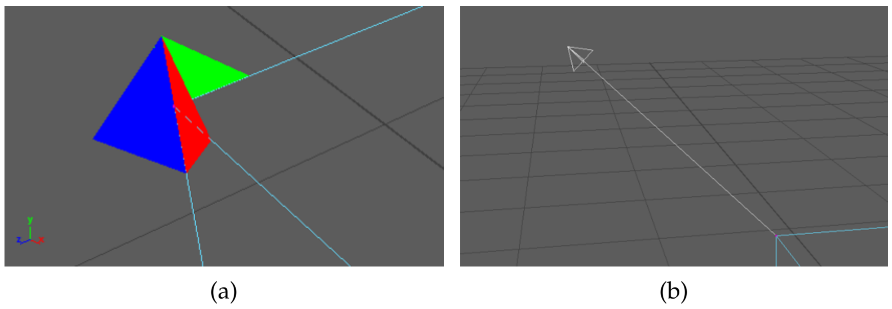


Figure 4.11: Constraints specified for control points of the subdivision surface are directly visualized in Maya's viewport. Fixed control points (a), i.e. the displacement along an axis is restricted, are indicated by colored triangles for each fixed axis. Prescribed displacements (b) are indicated by arrows. © *Florian Komposch [55], used with permission.*

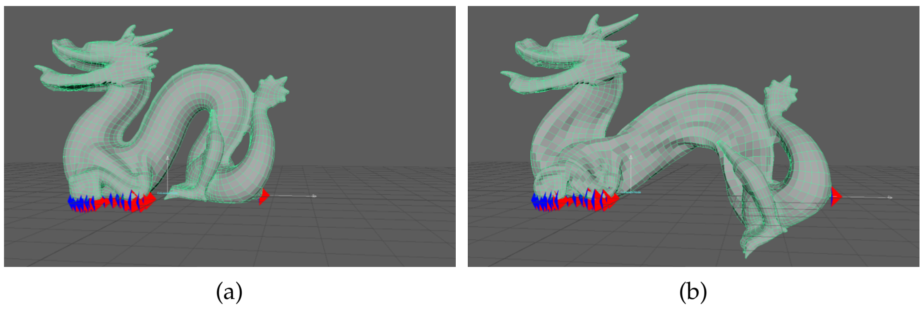
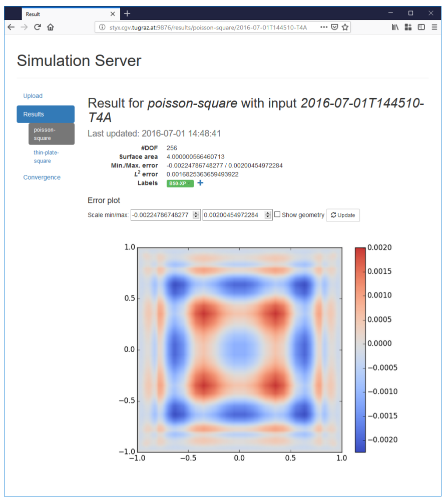


Figure 4.12: IGA tools integrated into Maya can be used to quickly deform complex meshes. Constraints and forces are defined on the initial surface (a) and the simulation computes a deformed surface (b) which is readily available in Maya for further use. © *Florian Komposch [55], used with permission.*

4.5.3 *Analysis Web Application*

Directly integrating the IGA framework into different software works well and provides a tight integration in the user interface of the respective software. However, it is not practical to write plugins for every design software as there are many design applications. To make IGA available to a wide audience of users, independent of the software they use, I investigated the possibility of making it available as a service.

To explore this approach, a web application has been built that enables users to upload designed geometry to a server. The web application offers several pre-defined simulations, which can be run on the uploaded geometry. The simulation itself is run directly on the server. This enables using the simulation also on less powerful de-



Simulation Server

Result for *poisson-square* with input *2016-07-01T144510-T4A*

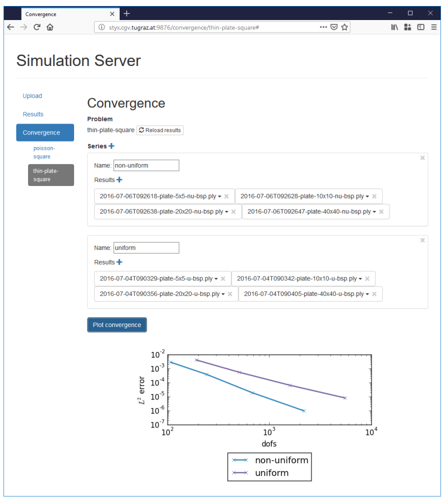
Last updated: 2016-07-01 14:48:41

#DOF	256
Surface area	4.000000566460713
Min./Max. error	-0.00224786748277 / 0.00200454972284
L^2 error	0.0016825363659493922
Labels	BS0-XP

Error plot

Scale min/max: -0.00224786748277 0.00200454972284 ☐ Show geometry

(a)



Simulation Server

Convergence

Problem

thin-plate-square

Series

Name: non-uniform

Results

2016-07-06T092618-plate-5x5-nu-bsp.ply
2016-07-06T092628-plate-10x10-nu-bsp.ply
2016-07-06T092638-plate-20x20-nu-bsp.ply
2016-07-06T092647-plate-40x40-nu-bsp.ply

Name: uniform

Results

2016-07-04T090329-plate-5x5-u-bsp.ply
2016-07-04T090342-plate-10x10-u-bsp.ply
2016-07-04T090356-plate-20x20-u-bsp.ply
2016-07-04T090405-plate-40x40-u-bsp.ply

(b)

Figure 4.13: A web interface allows to upload and simulate different geometries. Analysis results can be examined (a) and compared (b) directly in the web browser.

vices. After the simulation finishes, the results can also be examined directly in the browser. Additionally, the web application offers tools to compare different analysis results. Figure 4.13 shows a prototype web interface which can be used to upload and simulate different geometries and then analyse the results directly within a web browser. Individual results can be examined in detail and several results can be interactively grouped and compared, e.g. by plotting error measures for each result.

This interface has a number of advantages. Long running simulations can be conveniently started and their progress can be checked on the go using a mobile device. Also, the analysis results can be inspected and compared from any web browser. Further, this interface enables computation intensive analysis also for lower end devices like tablets by offloading the computation to a server. For example, simple modeling interfaces for mobile devices, e.g. based on procedural modeling, can be used to quickly create shapes. The web-based analysis application can then be used to check physical requirements, e.g. before sending the design to a 3D printer.

However, a browser based interface does not allow to integrate the analysis service into other applications, it can only be accessed through a web browser.

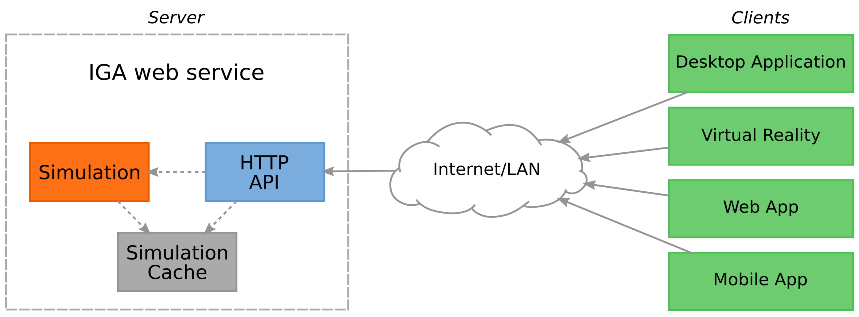


Figure 4.14: Architecture of the isogeometric analysis web service. Using the HTTP API, many different clients have access to the thin shell simulation.

4.5.4 *HTTP API*

To provide the thin shell simulation to different applications running on different types of devices, an HTTP API to access the simulation running on a central server is defined. This web-based simulation service can be accessed from any web enabled client, ranging from apps running on mobile devices to VR applications running on high end workstations. This separation of the simulation and the client allows for flexible use cases. For example, the simulation may run on a powerful server to be accessed by multiple mobile devices, or, the simulation can run on the same computer as the client to minimize latency for high performance VR applications.

Figure 4.14 shows an architectural overview of the isogeometric thin shell simulation web service. The server itself consists of three main parts:

- the thin shell simulation,
- an in-memory cache to store results, and
- the web server providing the HTTP API.

Each of these parts can be easily extended or replaced. For example, in addition or as a replacement for the thin shell simulation another simulation back-end could be added. Similarly, other APIs can be added to access the simulation, for example a WebSocket-based API to allow for bidirectional communication. However, this thesis focuses only on the HTTP API to access the isogeometric thin shell simulation.

The thin shell simulation is responsible for computing the simulation result for a given set of forces and constraints. Analysis results may be stored in the simulation cache for efficient access. The API provides easy access to this for clients, as detailed in the following.

To access the simulation service, an API based on the HTTP protocol is provided. The main advantage of the HTTP protocol is that it is available almost everywhere. This enables using the web service from any client, ranging from web applications running in a browser, or mobile applications running on tablets or phones to VR applications displayed in a CAVE or head mounted display (HMD). Another advantage is that distributed clients can seamlessly connect to the web service over the Internet, without requiring special rules, e.g. for firewalls, because the HTTP protocol is allowed in most cases, which might not be true for a custom protocol based on UDP or TCP.

The API is kept intentionally simple for ease of use and is split into two parts: the core API to compute deformations, and optional additional functionality that is useful in different applications.

4.5.4.1 *Core API*

A small core API is enough to provide the minimum functionality to compute deformations. It consists of the following HTTP endpoints:

`POST /geometries?namespace=` to upload a new geometry resource to the web service. The request data includes the geometry and material parameters. This returns a unique `geom_id` for the geometry, based on a hash of its optional `namespace` and the geometry and material data. This allows multiple clients to share the same geometry. If the clients work in the same namespace, the `geom_id` of a particular geometry is equal for all clients. Therefore, all clients share the same geometry and its deformation results. At the same time, using different values for the optional `namespace` enables clients to upload the same geometry, but treat them as different resources. Once the geometry is uploaded, the server starts computing the stiffness matrix corresponding to the geometry to setup the system of equations.

`POST /geometries/{geom_id}/results` to compute a deformation for the geometry with the given `geom_id`. All data required to compute the deformation, like constraints and forces, is sent in the body of the POST request. This endpoint returns the displacements of the DOF together with a monotonically increasing `res_id` value, starting at `1`.

If the stiffness matrix for this geometry is not yet available in the cache, this endpoint waits until the computation finishes and then computes the deformation.

The data format used for each of these endpoints can be tailored to the specific use case, e.g. a compressed binary format to save bandwidth, or a JSON encoding for ease of use in a web browser environment. An example binary format is specified in Appendix A.

4.5.4.2 *Additional Functionality*

The core API to compute deformations can be extended with additional functionality on the server, which is required for example to synchronize deformations across multiple clients and to implement more complex use cases. In the following, several optional endpoints are presented that provide additional useful features for the simulation service. Many additional application specific API endpoints are possible and can be easily added.

`GET /geometries` and
`GET /geometries/{geom_id}` to list and download subdivision control meshes and material data for geometries uploaded by a different client.

`GET /geometries/{geom_id}/results` and
`GET /geometries/{geom_id}/results/{res_id}` to list all computed deformation results for the given geometry and to download individual results. These endpoints can be used, among other things, to implement undo functionality into clients.

Any client that can send HTTP requests can be used to access the simulation web service, this includes e.g. desktop applications, web browsers, mobile applications and most game engines.

4.6 SUMMARY

This chapter presented applications of subdivision-based isogeometric analysis to extend traditional freeform surface modeling techniques. Not only can IGA be used to query additional information about the designed surfaces, like surface areas or curvature, it can also be used as a tool to design and deform surfaces. Therefore, RESEARCH QUESTION 2 can be answered in the affirmative.

Using isogeometric thin shell analysis integrated into a modeling application, constraint-based and force-based techniques can be used to compute deformed shapes. Additionally, the isogeometric framework can also be used as part of an optimization process, e.g. to improve the stability of a designed structure or to minimize its surface area.

Based on constraint-based and force-based deformations, new modeling tools can be provided to designers. An intuitive drag and drop interface enables designers to quickly change the overall shape of a design by just moving a few control points. A sketch-based interface can be used to sketch feature lines of a design, to which the surface automatically adapts. This can also be used to quickly create figure poses. All of these techniques have been integrated into existing modeling applications to validate their use cases. While this section answered questions regarding the user interface, writing a complete IGA plugin for each application is not ideal and therefore does not completely satisfy Research question 3.

For making this framework available to a wide audience, two service-oriented approaches were discussed. First, a web application has been presented which allows to upload geometry and use it for different predefined simulations. Analysis results can then be viewed and compared directly in a web browser. Additionally, this section presented a flexible client-server architecture of a simulation web service. The defined HTTP API provides this simulation to all web enabled devices, enabling physically accurate deformation also in resource constrained environments like mobile devices or web applications. This flexible solution of integrating IGA completes the requirements to satisfy Research question 3.

5 PHYSICAL DEFORMATIONS FOR VIRTUAL WORLDS

This chapter is based on research that has been published in the following peer-reviewed papers:

A. Riffnaller-Schiefer, U. H. Augsdörfer, and D. W. Fellner, "Interactive physics-based deformation for virtual worlds," in *2017 International Conference on Cyberworlds (CW)*, 2017, pp. 88–95

A. Riffnaller-Schiefer, U. H. Augsdörfer, and D. W. Fellner, "Physics-based deformation of subdivision surfaces for shared virtual worlds," *Computers & Graphics*, vol. 71, pp. 66 –76, 2018, ISSN: 0097-8493

While the previous chapter covered using analysis to design virtual objects which may eventually be manufactured in the real world, this chapter uses IGA to bring real world physics into virtual worlds.

To create immersive interactive virtual worlds, it is important to not only provide plausible visuals, but also to allow the user to interact with the virtual scene in a natural way. While rigid-body physics simulations are widely used to provide basic interaction, realistic soft-body deformations of virtual surfaces are challenging and computationally expensive and therefore typically not offered.

Virtual worlds are commonly accessed by multiple users with various different devices ranging from mobile devices to powerful workstations, making consistent soft-body deformations even harder to realize due to differences in computing and rendering capabilities.

To provide realistic deformations to virtual worlds on all kinds of devices, a client-server architecture is proposed. Deformations are computed on a central server and results are available to multiple clients. By using subdivision surfaces to represent the geometry and to perform the simulation computations, bandwidth usage is kept low and different clients can easily render different levels of detail according to their capabilities. This enables collaboration of different types of clients, like web applications, mobile devices or CAD applications, using the same geometry.

Many existing techniques allow deformation of surfaces. Good summaries can be found in [64] and [16]. The various approaches differ in physical accuracy and performance. However, many of them are either

- not based on physical principles, and therefore often require tweaking of many parameters to achieve the desired results, or
- hard to synchronize, because the data requires a lot of bandwidth or because devices with different rendering capabilities require different meshes, or
- too slow to allow real-time interaction.

5.1 INTERACTIVE SIMULATION

With all building blocks for the thin shell simulation from Chapter 2 and Section 3.1, a linear static deformation of a subdivision surface can be computed by constructing the stiffness matrix $\boldsymbol{K}$ and applying the desired forces and constraints to the system of equations. Unfortunately, this is generally too slow for interactive use of the simulation, with building the stiffness matrix being the most time consuming part.

However, as observed in [74], if the goal is to apply different forces and constraints to the same initial subdivision surface, the stiffness matrix stays the same and has to be computed only once for the initial surface. Therefore, to use the isogeometric thin shell simulation interactively, the stiffness matrix is precomputed for the initial geometry and subsequently used with different sets of forces and constraints to compute deformed surfaces.

For the use case of interactive deformation in virtual worlds, the linear simulation provides plausible deformations in many cases, even where overall deformations are large. One notable exception are large rotational deformations which can cause artifacts where the surface area increases unrealistically, as shown in [64].

5.2 SERVER

The server for the interactive simulation is based on the architecture and API presented in Section 4.5.4.

After the geometry is uploaded, the thin shell simulation on the server is responsible for precomputing the stiffness matrix, as described in Section 5.1. After this precomputation is performed, deformations can be computed quickly. The precomputed matrix and computed deformation results are stored in the in-memory cache for fast access.

For the specific use cases of the interactive simulation, the API defined in Section 4.5.4 is extended with the following HTTP end points:

`GET /geometries/{geom_id}/latest_result?have=` can be used to get the latest deformation result for the given geometry. The optional parameter `have` can be used by the client to indicate which `res_id` it already has. The `res_id` `0` is special in that it indicates that the client only has the initial, undeformed geometry. If there is a newer result, with a higher ID, available, the server returns the deformed geometry and `res_id` of the latest result. If the client already has the latest result, the server can keep the request open and send a response once a new result is available. This minimizes the number of round trips between client and server. Alternatively, the server could also return an empty response to the client if keeping the request connection open is not desired or not supported by the server.

`GET /geometries/{geom_id}/limit_comb?u=&v=&face-index=` is used to get the linear combination of control points defining the limit point for a certain parameter location (u, v) on the face with index `face-index` in the subdivision control mesh. This returns the linear combinations for the limit position as well as for the tangents at the given location. This can be used by the client to get linear combinations needed e.g. for Lagrange multiplier constraints of the limit surface. Using this API, the client does not need to implement complex evaluations of the subdivision surface itself.

5.3 CLIENT

Any application or device that can send HTTP requests can be used as a client for the interactive simulation server.

Here, for ease of implementation, Catmull-Clark subdivision surfaces, the industry standard in the entertainment industry, are used on the client, which are a subset of the NURBS compatible subdivision scheme used for IGA on the server.

For simple use cases the client needs to provide only the geometry data for a Catmull-Clark control mesh and a set of forces and constraints to compute a deformed surface. In some cases, e.g. to apply constraints to the subdivision limit surface, more information is required.

5.3.1 *Catmull-Clark Parameter Estimation*

To provide plausible interaction with an object at arbitrary points on the surface, Lagrange multiplier constraints on the limit surface are used. To apply a constraint on the limit surface, the linear combination of control points is needed that defines the limit position, as discussed in Section 3.1.5. To compute these linear combinations, either within the client application or with the additional API from Section 5.2, the (local) parameter location of the desired point on the corresponding subdivision patch is required. Figure 5.1 summarizes a simple approach to estimate the parameter values of the limit point using a slightly extended implementation of Catmull-Clark subdivision on the client. This method is similar to some pre-tessellation methods to evaluate Catmull-Clark surfaces for ray tracing [12]. Here, the tessellation is used to estimate the parameter location of a ray intersecting the surface, e.g. for user interaction or collision detection, and not to evaluate the limit surface itself.

During each subdivision step producing the dense mesh for rendering, the index i of the original *root* face (Figure 5.1 top left) in the control mesh, from which each subdivided face originated, is stored as r. Additionally, for each face in the subdivided mesh the index of the corresponding corner vertex in its parent face, i.e. 0, 1, 2 or 3 for quads, is stored. This index is stored for every subdivision level and defines the path p from the root face to a given face in the quadtree of subdivided faces (Figure 5.1 top right and bottom left). So, a face at subdivision level 2 stores two values, e.g. $[2, 1]$ for the right most face just above the vertical center. For rendering, each quad is split into two triangles (Figure 5.1 bottom right), each of which refers to the same information stored for the quad. This information requires one additional index and 2 bits per subdivision level of additional storage for each subdivided face. For the NURBS compatible subdivision scheme, which additionally allows just horizontal/vertical splitting of a face, more bits to cover all cases would be needed.

To find the control face and parameter location of a ray intersecting the Catmull-Clark surface, the intersection is first computed with the triangle mesh used for rendering. If there is an intersection, the previ-

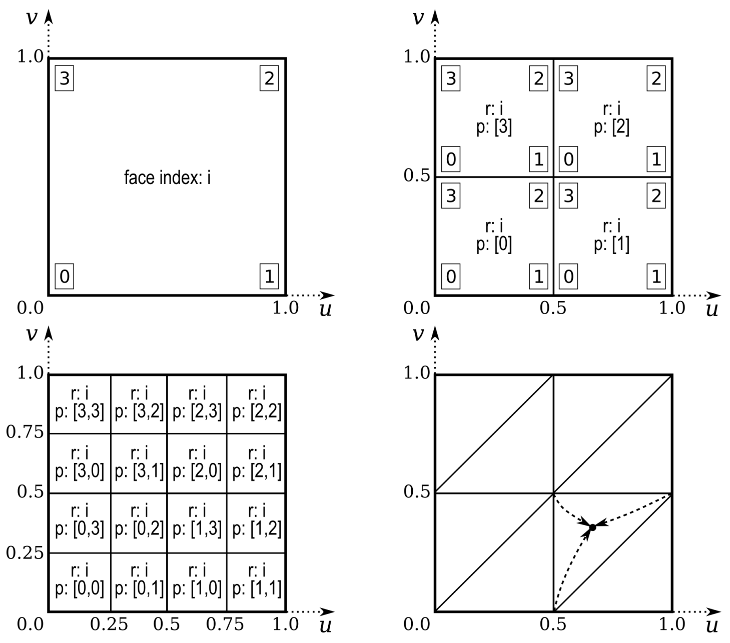


Figure 5.1: Estimating parameter location of an intersection with a tessellated Catmull-Clark subdivision surface.

ously stored information for that triangle/quad is retrieved. From path p the possible range of parameter values can be derived, i.e. the face with path $[2, 1]$ contains the (u, v) parameter domain from $(0.75, 0.5)$ to $(1.0, 0.75)$. The first path item 2 limits the u domain to $[0.5 \dots 1.0]$ and the v domain also to $[0.5 \dots 1.0]$. The second item 1 then restricts these domains further, for u to $[0.75 \dots 1.0]$ and for v to $[0.5 \dots 0.75]$.

An approximation of the final parameter value can be computed as a linear interpolation of the corresponding range boundaries based on the barycentric coordinates of the ray intersection with the triangle (Figure 5.1 bottom right). This assumes that there is a mapping from the triangle index to the quad face of the subdivided surface storing the necessary values. This is usually the case if all quads are just split into two triangles for rendering, like in the bottom right image in Figure 5.1. The higher the subdivision level of the mesh used for intersection testing, the more accurate the approximation of the parameter values.

The face index in the control mesh used for evaluation is found by using the stored index r of the selected face. The linear combination of control points defining the limit position at the approximated parameter location are derived from the Catmull-Clark basis functions. Because the simulation service already needs to perform such evaluations for the isogeometric thin shell analysis, see Section 3.1.3, it can also provide functionality to compute these linear combinations, as described in Section 5.2. However, the face index and parameter location always need to be provided by the client.

5.4 LEVEL OF DETAIL SIMULATION

One advantage of subdivision surfaces is that the level of detail for the visualization can be easily adapted by rendering a different subdivision level of the surface. Therefore, objects close to the virtual camera can be rendered at a high subdivision level, resulting in many triangles to represent a smooth surface, while objects far away can be rendered at a low subdivision level or even without subdivision.

Subdivision can also be applied to increase the DOF of the surface for analysis. Providing surfaces with more DOF to the simulation leads to more accurate and detailed deformation results. However, the number of DOF also directly affects the time needed to compute the deformation, as will be discussed in more detail in Section 5.6. Therefore, there is always a trade-off between accuracy and computation time.

To achieve a fast response time but eventually also get an accurate result, a parallel simulation scheme can be used. The basic idea is to perform the simulation on multiple subdivision levels of the surface at the same time. The result for the coarse mesh can be computed quickly and can therefore be presented to the user immediately. Meanwhile, a variant of the same surface with more DOF takes longer to compute, but provides the user with a more accurate deformation result, once it is available.

The parallel simulation can be controlled entirely by the client, without modifying or extending the HTTP API from Section 4.5.4. However, it assumes that the server can handle multiple requests simultaneously, to compute several deformations in parallel. To be able to simulate multiple subdivision levels of a surface, the client uploads each level as a separate geometry to the server. In other words, instead of just uploading the initial control mesh of an object, the client uploads the initial control mesh and additionally also subdivides this control mesh and uploads the resulting subdivided geometry as a separate object to the server. It is important to note that in this case both control meshes define the exact same subdivision limit surface, on which all simulations are based.

While simply subdividing the control mesh for the simulation could also be performed by the server, often, additional details are added to the subdivision surface for visualization on the client, e.g. using displacement maps. The client can also add these additional visualization details as geometric details to the subdivided control mesh by displacing the control points. This way, the details in the geometry are part of the simulation and can therefore lead to a more accurate deformation result.

However, as discussed in Section 3.3.1, using different levels of detail of a surface and adding small geometric details can significantly change the simulation result. This needs to be considered when providing deformation results based on different levels of detail.

Another challenge with this approach is to find corresponding constraints and forces for the subdivided control meshes. Often, the client only has constraints and forces for the initial control mesh, e.g. because only the coarse initial control polygon is shown in the user interface for manipulation, but a more accurate, higher resolution simulation result of the limit surface should eventually be visualized. In that case, the client needs to derive constraints and forces for the higher subdivision level, which has different DOF, from the user interaction with the initial control points. Having the corresponding constraints and forces, the client can then separately request deformation results for each subdivision level from the server.

Simple constraints of single DOF, as mentioned in Section 3.1.5, are difficult to apply exactly on a subdivided control mesh. The reason is that the influence of a single DOF of the coarse control mesh on the surface is spread to multiple DOF in the subdivided mesh. If the constraint itself is spread in a similar way, multiple neighboring DOF are constraint, instead of just one. Unfortunately, this not only results in the desired displacement, defined by the constraint, but also prevents any rotation of the surface, as all neighboring DOF are constraint, which is not desired.

Interestingly, in many cases a good approximation of the constraint on the coarse control mesh is to just apply the same constraint to the corresponding control point of the subdivided control mesh, as demonstrated in Figure 5.2. As the thin shell simulation tries to minimize the deformation energy required to satisfy the constraints, control points nearby constrained DOF are automatically moved similar to the defined constraint to avoid deforming the surface, if possible. And due to the Catmull-Clark subdivision rules, moving a control point and its one-ring neighborhood directly relates to the movement of the corresponding region on the subdivision limit surface. Therefore, neglecting the influence of other constraints and forces, a single displacement constraint causes approximately the same displacement

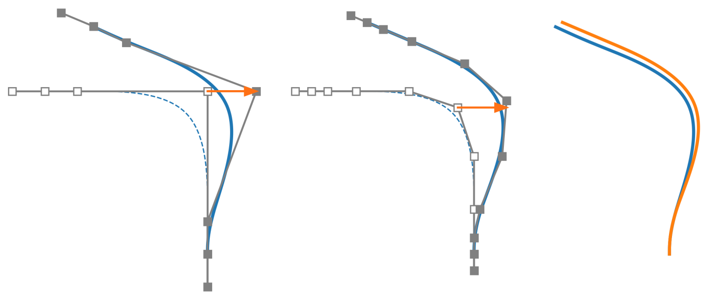


Figure 5.2: The same constraint, a displacement of one x-coordinate visualized with an orange arrow, is applied to different subdivision levels of the same curve (left, center), of which the bottom end is fixed. Even though the DOF of the curves are different, the resulting deformation is similar. In the comparison on the right, the blue result is the constraint applied to the original control polygon and the orange result is for the subdivided curve. By minimizing the energy caused by the deformation, the simulation automatically moves nearby control points in a similar way as the constraint DOF. Therefore, the limit surface moves approximately as specified by the single displacement constraint, independent of the number of DOF.

of the limit surface, independent of the subdivision level it is applied to.

Forces applied at DOF are also valid on the subdivided control mesh, but the area of influence is reduced because the support of each DOF on the limit surface shrinks with each subdivision level. However, the user must be given a consistent analysis feedback and the force applied needs to be consistent and irrespective of the level of subdivision. To spread the force to the same region as on the initial coarse control mesh, the force vectors can be subdivided according to the Catmull-Clark subdivision rules in the same way as the control points are subdivided. This way, a force applied to a single DOF in the coarse control mesh is split into multiple forces, applied to the corresponding DOF and its one-ring neighborhood in the subdivided control mesh. The subdivision rules ensure that the total force applied to the surface stays constant.

Instead of defining forces and constraints on the control mesh, problems associated with this approach can be overcome by using Lagrange multipliers as described in Section 3.1.5. Constraints defined directly for the limit surface are also valid for subdivided control meshes, because they are defined for a certain position on the subdivision limit surface, independent of the control points. However, the linear combination of DOF that define the limit constraint need to be computed separately for each control mesh, as shown in Section 5.3.1. Limit constraints always ensure that surfaces displace exactly the desired distance, independent of the control mesh. Therefore, they are the preferred way to apply constraints from direct user interaction.

5.5 APPLICATIONS

This section presents several applications using the proposed simulation web service in various environments for different use cases.

5.5.1 *Multi-Client*

A challenge of multi-client VR environments is the correct and efficient synchronization of the different clients. This especially includes the synchronization of soft-body deformations. If a dense polygon mesh is used for deformation, transferring the deformed surface can require a lot of bandwidth. On the other hand, if each client computes the deformations independently, inconsistencies can appear due to different timing or computing capabilities of the devices.

Using the central simulation web service based on Catmull-Clark subdivision surfaces to synchronize deformed surfaces across multiple devices overcomes these issues. First, only the subdivision control mesh needs to be synchronized between clients, using less bandwidth. And second, all clients get the exact same deformation because it is only computed once by the server for any given geometry. Each client then performs Catmull-Clark subdivision on the control mesh to derive the smooth limit surface for visualization. This allows each client to render different levels of detail of surfaces without inconsistencies, because the limit surface is fully defined by the control mesh and the subdivision scheme.

Figure 5.3 shows two devices, a workstation and a tablet, running the same application connected to the central simulation server. Surface deformations caused by user interaction are immediately synchronized between both devices, using the `latest_result` API endpoint described in Section 5.2.

Currently, the applied forces and constraints are not synchronized between clients, only the deformation results are shared. Therefore, deformations requested from one device override previous results requested on other devices, as usually only the last deformation result is shown. The server and the API could be extended to also store and synchronize the constraints and forces provided by the requesting

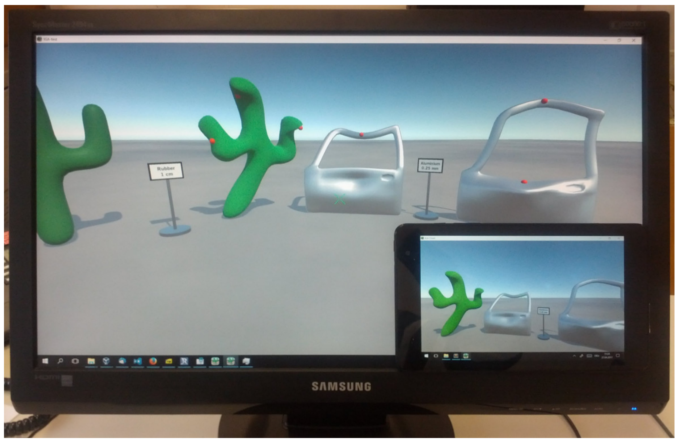


Figure 5.3: Soft-body deformations are synchronized across multiple devices. Both the workstation and the tablet run the same application connected to the simulation web service. Deformations caused by user interaction are immediately synchronized between both devices.

client together with the deformation result. This would allow multiple clients to deform an object alternately. A difficulty with this approach are race conditions due to simultaneous deformation requests by multiple clients. However, synchronizing constraints and forces is currently not implemented and this application therefore has a *master* client for each object who defines the applied constraints and forces.

One use case for this type of synchronization is rendering multiple projections of the same scene on multiple devices, as it is usually done in a CAVE [32] environment or on a large tiled display [46]. This way, it is ensured that all projections render the exact same soft-body deformation, controlled by a single master device.

5.5.2 *High Performance VR*

Current state-of-the-art computer graphics techniques can be used to create immersive virtual worlds that *look* almost real, however these worlds do not *feel* real because the interaction with objects is limited. Many real world objects are not rigid and users expect to be able to deform them. To improve realism, interactive soft-body deformation of virtual objects can be provided using the simulation web service. However, visualizing such virtual worlds using immersive head mounted displays like the HTC Vive or Occulus Rift requires high frame rates and low latency processing of user inputs.

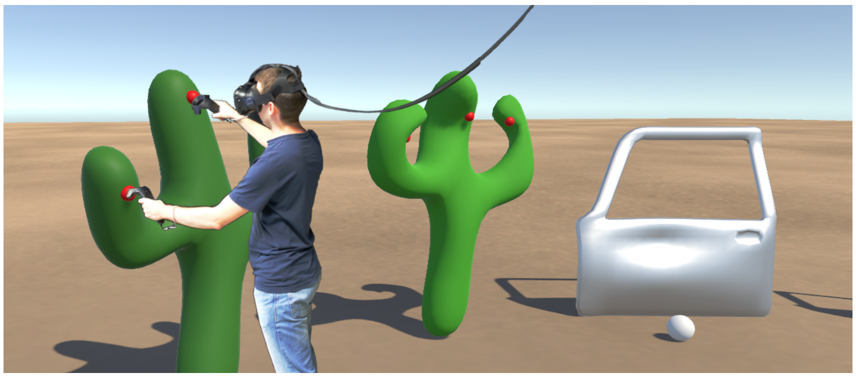


Figure 5.4: Subdivision surfaces can be interactively deformed in an intuitive way using hand controllers and a head mounted display. Users interact directly with the subdivision limit surface and can define constraints, visualized with small red spheres, by pushing and pulling the surface. Additionally, users can also pick up and move rigid body objects, like the white ball in the scene, which can also cause deformations by colliding with surfaces.

To ensure high performance and low latency of the VR application, the deformation service can run on the same high end workstation as the visualization. It does not slow down the rendering loop of the visualization because it runs in a separate process and network latency is minimized by running the server and the client on the same computer.

In a prototype application for the HTC Vive HMD, shown in Figure 5.4, interactive deformation has been implemented using a simple push/pull interface using the tracked hand controllers. Users can simply grab and move a surface directly with the controllers in their hands, similar to the real world. Grabbing the surface adds a constraint on the subdivision limit surface which attaches the surface to the users hand and which is updated when the user moves the controller. To compute the point selected by the user on the surface, a ray based on the position and orientation of the users hand is intersected with the triangulated mesh used to render the subdivision surface. The face index in the subdivision control mesh and the parameter location of the intersection is then computed as discussed in Section 5.3.1. The deformed surface is asynchronously computed by the simulation web service and, once the result is available, updated in the visualization to interactively reflect the user input. The grabbed surface point therefore exactly follows the users hand movement, while the rest of the surface deforms according to its material properties and other constraints which e.g. hold it in place. Applying the constraint on the limit surface uses the `limit_comb` API endpoint described in Section 5.2 and allows the user to not only change the position of the surface but also to control the tangent plane at the grabbed surface point.

However, as explained in Section 3.1.5 multiple DOF of the surface need to be constrained to prevent any rigid body motion. As a simple intuitive metaphor to fix virtual objects in place, all parts of an object touching the floor are automatically constrained to the floor at that position.

For additional feedback, vibrations of the controller are used to let the user *feel* the deformation. The more deformation the interactions cause, the stronger the vibrations of the controller.

In addition to direct user input also interaction of the soft-body deformation with traditional rigid body physics has been implemented. This lets users deform virtual surfaces by e.g. throwing rigid objects on them. Once a collision with a rigid body object is detected, the parameter location of the collision point can be computed similar to how it is done for the user interaction. But instead of constraints, forces derived from the colliding objects are applied to the surface. Each surface will respond to collisions according to its material properties, improving realism.

5.5.3 *Interactive Collaborative Web-based Tools*

A big advantage of the HTTP-based API described in Section 4.5.4 is that it can be accessed from any client that can use the HTTP protocol. This also includes web browsers, which use HTTP natively to transfer data from web servers. Together with other features of modern web browsers like fast JavaScript implementations, WebGL [62] and more recently WebVR [99], they can be used to create interactive visualizations of virtual worlds or virtual objects. Users can easily access such visualizations on the web without installing any special software besides a web browser. Using the presented deformation web service now also enables users to interactively, and possible collaboratively, deform objects in a web browser, as shown in Figure 5.5. Providing simulation tools for web-based applications has many use cases, for example in games, product customization or engineering.

One possible use case is for example collaborative design and evaluation. To collaboratively evaluate designed objects, plugins for modeling or CAD software can send the subdivision geometry directly from the modeling application to the simulation server. A web-based application allows to share access to these models with other users

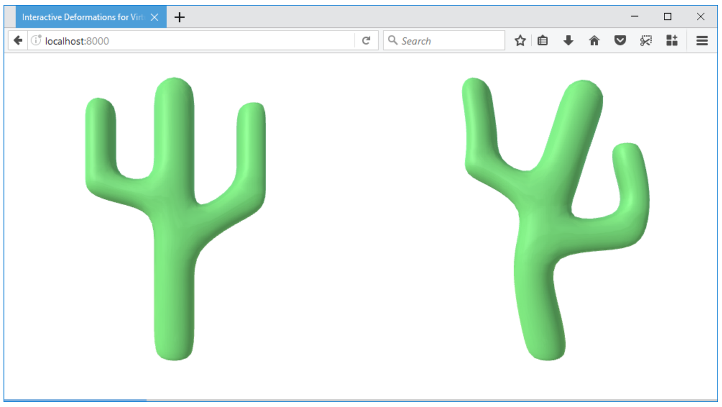


Figure 5.5: The simulation server can also be accessed from web applications running in a browser. On the left, a cactus model is shown in its initial state. The deformed surface, after applying some constraints, is shown on the right. Scripts running in the browser control the simulation parameters based on user interaction and efficiently render the surfaces in real time using WebGL.

and to interactively perform simulations. This enables users to discuss design ideas and to simulate and analyse structural properties. This use case is equally applicable to designing virtual worlds as it is to rapid prototyping of objects that are manufactured for the real world.

Experiments by Graf and Stork [43] have shown that VR techniques can help with the evaluation of analysis results for CAE. The presented simulation service can be used to seamlessly connect design and analysis in VR for such use cases. To do so, different interfaces can be combined, as the only requirement for collaboration is that all users access the same simulation server. Therefore, a user can, for example, control simulation parameters with a web-based interface on a mobile device, while the simulation results are visualized for evaluation in a VR environment. Meanwhile, a designer can create and edit geometry in a CAD application and seamlessly upload new geometry to the simulation server for other users to simulate.

This type of application makes use of all of the additional APIs defined in Section 4.5.4.2. Depending on the application, also more APIs can be defined on the server, e.g. to compute stresses in the material of a simulated object intended for manufacturing or to compute an accurate non-linear simulation non-interactively for engineering analysis. By augmenting the interactive features of the presented API with engineering tools, the presented simulation service can therefore also serve as the basis for analysis-as-a-service, as described in Section 4.5.3.

5.6 PERFORMANCE

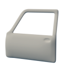
This section discusses several performance related topics of the presented simulation web service. All timing measurements were performed on a workstation with an Intel Core i7-3820 CPU running at 3.6 GHz and 12 GB of main memory.

5.6.1 *Precomputation*

The most time consuming part of computing a thin shell deformation is the derivation of the stiffness matrix. As noted in Section 5.1, this has to be done only once per object to compute linear simulation results and can be cached for subsequent uses of the model.

Table 5.1 lists the time needed to perform this initial precomputation for various models. Generally, the computation time depends on the number of faces in the subdivision control mesh, i.e. the number of elements in terms of FE. However, it also depends on the number of EVs in the control mesh. Regions of the Catmull-Clark subdivision surface containing EVs cannot be evaluated as cubic B-splines but require a more complex evaluation [91]. Usually, the precomputation takes between a few seconds and a few minutes, depending on the complexity of the subdivision control mesh.

Table 5.1: Time needed to precompute the stiffness matrix and to solve the system of equations for various models.

				
Model	Cactus	Barrel	Car door	Horse
Control Vertices	60	210	218	472
thereof EVs	32	8	24	64
Faces	58	208	218	470
Precomputation	14 s	7 s	20 s	82 s
Solving	2 ms	18 ms	18 ms	75 ms

Once the stiffness matrix has been either precomputed or loaded from a cache, a deformed version of the surface can be computed quickly, depending on the size of the system of equations, by applying forces and constraints and solving the resulting system of equations from Equation 3.11 or Equation 3.13.

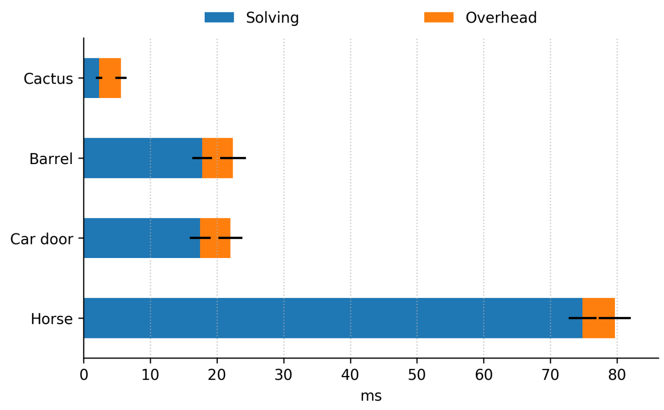


Figure 5.6: Average latency and standard deviation (visualized as black bars) for deforming various objects: The biggest part, solving the system of equations, is performed using a CPU-based sparse matrix solver for smaller meshes and using a GPU-based solver for the *Horse* model. The remaining overhead includes request processing and network latency on a LAN.

5.6.2 *Latency*

It is important for interactive applications to minimize the time between user interaction and visualization of the corresponding changes. This applies especially to VR applications. High latency limits immersion and can cause nausea for users of the VR application [2].

Figure 5.6 shows the total latency, from request until the result is available at the client, for deforming different objects. The stiffness matrix for each object was already precomputed before. The values show the average latency observed during an interactive modeling session creating approximately 300 deformations for each object using varying combinations of vertex constraints, Lagrange constraints and forces. The black error bars visualize the standard deviation of the measured timings. Most of the time is spent solving the linear system of equations on the server. This also includes building the deformed result by adding the computed deformations to the initial control points. The effect of additional Lagrange constraints, which increase the size of the system of equations, is small compared to the overall time to solve the system of equations. The remaining overhead includes processing the HTTP requests on the client and server and transferring the data over a local area network (LAN).

The systems of equations are either solved with a standard CPU-based sparse matrix solver or using a GPU-based solver based on the *cuSOLVER* CUDA library, depending on the number of DOF. Using the GPU-based solver has higher setup costs and therefore slower

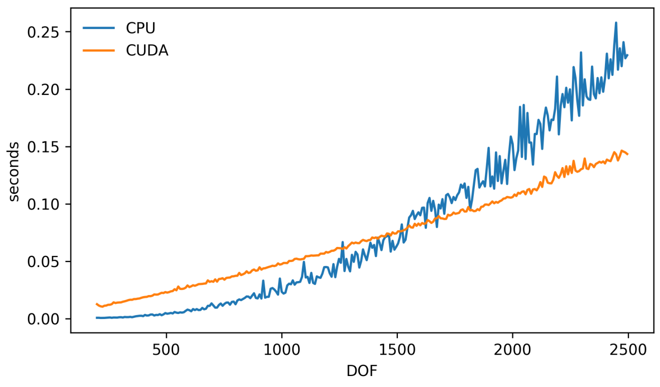


Figure 5.7: A GPU-based solver based on the *cuSOLVER* CUDA library improves performance for larger systems of equations, but has a higher overhead compared to a CPU-based solver. Therefore, the solver is dynamically chosen based on the DOF of the simulated mesh.

performance for solving the equations resulting from smaller meshes. A comparison of the two solvers for random systems of equations with varying DOF is shown in Figure 5.7. At around 1400 DOF, the GPU-based solver tends to be as fast as the CPU-based solver in most cases. For even larger systems of equations the GPU-based solver is significantly faster. Therefore, for larger meshes with more than approximately 1400 DOF, like the *Horse* model, the GPU-based solver is used, while for smaller meshes the CPU-based solver is the better choice.

The perceived latency for the user depends on the frame rate of the visualization. For desktop applications, typically rendering 60 frames per second (FPS), the deformed result for a moderately sized mesh will be visualized 1-2 frames after user interaction. This delay of up to 33 ms is acceptable and usually not noticed by the user. The *Horse* mesh has a latency of 5 frames at 60 FPS, or 83 ms, which is already noticeable but still allows interactive deformation.

5.6.3 *Bandwidth*

One of the challenges for synchronizing soft-body deformations over a network is the required bandwidth. In addition to object position, velocity, etc., which also need to be synchronized in rigid-body physics common in multiplayer games, soft-body deformations also need to synchronize each vertex of the mesh.

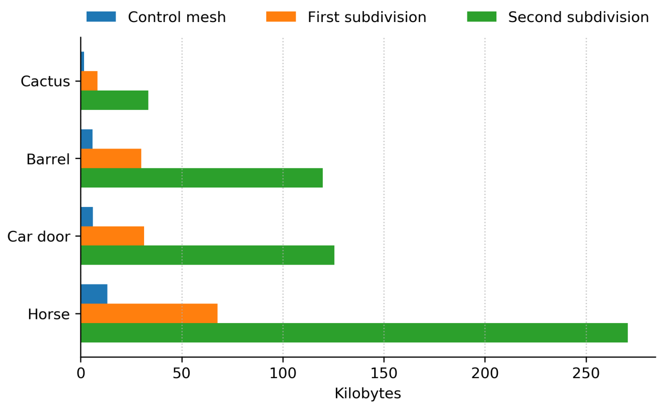


Figure 5.8: Size of initial mesh data in kilobytes. The subdivision control mesh requires significantly less data than the triangulated dense meshes used for rendering. By taking advantage of isogeometric analysis, the control mesh can be directly used to compute deformations.

For rendering, the subdivision control mesh is typically subdivided at least once or twice before triangulation to get a smooth looking surface. By taking advantage of IGA based on subdivision surfaces, only the subdivision control mesh is sent to the server when uploading new geometry. This requires considerably less data compared to purely mesh based approaches, where the complete mesh that is visualized is required to compute a deformation result.

As a baseline, Figure 5.8 compares the required data in kilobytes (10^3 bytes) for transferring the initial control mesh and the triangulations of the first and second subdivision level of that mesh. This assumes that vertices are defined using three single precision floating-point values and n-sided faces are defined by n 32-bit unsigned integer indices. Using the subdivision control mesh, instead of the denser meshes used for rendering, saves a significant amount of data.

When transferring the resulting deformed mesh data back to the client, only the vertices of the subdivision control mesh have to be transmitted, as the topology does not change. This also results in similar savings compared to a dense triangulated mesh, as shown in Figure 5.9 where the required bandwidth for 60 deformation updates per second is compared for the different representations. Therefore, the network overhead of transferring the result, included in the timings in Figure 5.6, is usually low compared to the time needed to solve the system of equations.

For all mesh-based deformation techniques, including our subdivision-based approach, many existing methods for 3D mesh compression

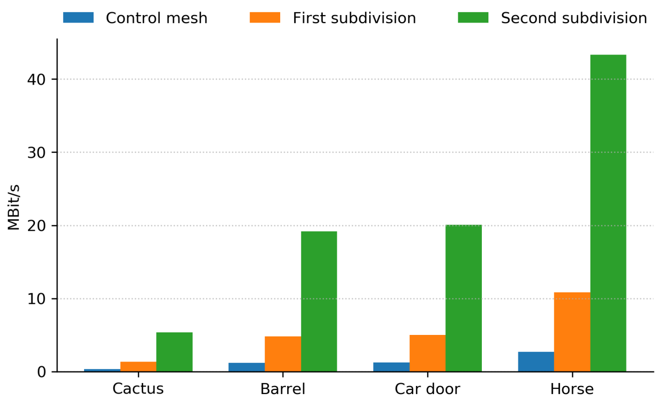


Figure 5.9: The required bandwidth to transfer 60 deformation updates per second varies significantly between the subdivision control mesh and the triangulated dense meshes used for rendering. Transferring just the subdivision control vertices for each deformation result enables using the simulation web service also over slower network connections.

could be used to further reduce the required bandwidth, as summarized in [67] and [61].

5.7 SUMMARY

This chapter extended the client-server architecture from Section 4.5.4 to provide a simulation web service to interactively compute and synchronize deformations of moderately complex subdivision surfaces.

Users can directly interact with the subdivision limit surface, while only the control mesh needs to be synchronized. By using subdivision surfaces, different levels of detail of a surface can be simulated. For moderately complex surfaces, the IGA simulation is shown to be fast enough for interactive use, thus satisfying Research question 4. This was achieved by precomputing and caching the stiffness matrix for the simulation on the server.

The web service architecture has many different use cases, e.g. realistic deformations for interactive virtual worlds, interactive physics-based product customization or collaborative analysis of designs. Using the presented architecture, different types of clients, like web applications, mobile devices or CAD applications, can seamlessly interact with the same geometry. This can also be used to enhance the design workflow, similar to the tools discussed in Chapter 4, e.g. designers can explore and modify surfaces in VR, or perform analysis tasks in VR by directly providing forces and constraints.

Part III

DISCUSSION AND OUTLOOK

"Even though the future seems far away, it is actually beginning right now."

Mattie Stepanek

6 CONCLUSIONS

The IGA framework presented in this thesis successfully combines the advantages of NURBS and subdivision surfaces for analysis. Subdivision surfaces are common in the entertainment industry, but, due to their advantages over NURBS, are becoming increasingly important in CAD. Unlike NURBS, using subdivision any shape can be represented with a single surface. Therefore, the framework presented in this thesis supports arbitrary topology surfaces without the need of preprocessing the CAD model to ensure continuity across the surface, as is required when employing NURBS patches for analysis. This is a mayor advantage in the isogeometric analysis of freeform surfaces.

6.1 CONTRIBUTIONS

Throughout this thesis, a number of research questions, stated in Section 1.2, have been discussed. To answer the research questions, a number of contributions advancing the state-of-the-art were presented in this thesis.

RESEARCH QUESTION 1 asked whether it is possible to combine the advantages of freeform surfaces and the precise control of NURBS in an IGA setting. In Section 2.2 the NURBS compatible subdivision scheme has been identified as an ideal basis to answer this question positively and in Chapter 3 the isogeometric concept was successfully extended to this geometric representation.

The first contribution was to extend the evaluation method of Stam [91] to the NURBS compatible subdivision scheme of Cashman [19] in Section 3.1.3.2. This enabled the implementation of the presented IGA framework based on NURBS compatible subdivision surfaces. To the author's knowledge, this is the first IGA implementation for subdivision surfaces with support for watertight, arbitrary topology meshes with non-uniform, rational and higher degree basis functions.

The problem of diverging derivatives around extraordinary vertices using the method of Stam [91] was not fully solved. While for many practical applications the remaining errors are not relevant, two techniques were presented to improve the evaluation accuracy for subdivision surfaces:

- a patching algorithm adapted from Peters [69] in Section 3.1.3.1, splitting the surface into regular patches and thus avoiding EVs entirely, and
- a simple adaptive integration method in Section 3.1.4, improving accuracy around EVs using Stam's evaluation approach.

The patch-based evaluation scheme does not suffer from the problem of diverging derivatives, but only approximates curved geometry. It is therefore a viable approach for flat surfaces, but is problematic in the analysis of freeform surfaces, which were the focus of this research. Nevertheless, using the patching technique, the commonly assumed drawback of subdivision based IGA not passing the patch test has been proven wrong. It has been shown in Section 3.2.3 that, using this technique, subdivision surfaces can pass the patch test exactly up to machine precision.

Extending the state of the art in IGA to the NURBS compatible subdivision scheme enables precise control in regular regions of the defined surface and its properties like tangents and boundaries. This eliminates the need for the various workarounds used in other subdivision based approaches [7, 25, 103] to e.g. set boundary constraints or handle symmetries. For improved accuracy, a rational representation can be used to exactly define conic sections without artifacts, ensuring reliable analysis. The analysis performance in terms of convergence is equivalent to NURBS, which, using higher degrees, can be significantly better than for low degree subdivision surfaces.

The application of IGA not only for engineering analysis, but also for the design process itself, as discussed in Research question 2, has been covered in Chapter 4. It has been shown that IGA can indeed be useful for the design. One use case is to guide the design towards a structurally stable or material efficient shape by visualizing different analysis results directly on the design or by automatically optimizing the shape. Another use case is to build physics-based design tools using IGA to extend the means available to a designer in order to modify the shape. The designer can interact with the simulation by setting constraints and forces, which can be done with intuitive drag and drop or sketch-based interfaces. Prototypes for these use cases have been integrated into the standard 3D modeling applications Blender and Maya.

The integration of IGA into other applications, topic of Research question 3, has been discussed in Chapter 4 and Chapter 5. Writing a plugin containing the IGA framework for each design software, like Blender or Maya, is possible, but does not scale to arbitrary software to reach a wide range of users. A more promising approach is to offer IGA as a service, which allows to access isogeometric computations from a wide range of different applications and devices.

In Chapter 5 the service oriented approach developed in Chapter 4 was used to address Research question 4. The analysis web-service has laid the foundation for a successful integration of IGA into an interactive VR environment, while traditionally, IGA was only used for non-interactive offline simulations. Using the web-service, soft-body deformations are computed in response to user interaction with objects in the VR environment. To speed up deformation during run-time some computations were precomputed and cached on the server. With this approach realistic soft-body deformations of moderately complex meshes can be computed at interactive rates.

6.2 applications

The unified IGA platform based on NURBS compatible subdivision surfaces was successfully applied to multiple areas of application:

- thin shell analysis,
- freeform surface design, and
- interactive simulations.

This flexibility allows for many different use cases.

First, the tight integration of design and analysis can be beneficial for product development. Using IGA in the early design phases, designers can quickly iterate on the design with quick feedback on structural properties from the analysis. The analysis could also be used to optimize the design with respect to certain structural criteria.

The IGA framework can also be used as the foundation for several physics-based modeling tools to create natural deformations of shapes. This simplifies for example the animation process for complex objects, where moving control points manually is time consuming. Besides the time aspect, even creating a naturally looking deformation manually is challenging in itself. This use case can also be extended to interactive applications, where deformations are computed in real time to improve realism in virtual worlds. However, the current implementation has some limitations regarding the size of meshes for real time deformations.

Finally, the presented IGA framework can be used to provide an analysis-as-a-service platform, which can be used for many different use cases. Providing easy access to IGA enables integration of analysis features into a wide range of different applications and devices. For example, interactive multi-user engineering and entertainment applications can use the service to compute realistic deformations. Other examples are applications for customized user designs, which e.g. can use the service to automatically verify the design of a user, even on mobile devices or within web applications.

6.3 LIMITATIONS AND FUTURE WORK

While the presented IGA framework can already be used for many applications, as discussed in the previous section, it could be extended and improved in a number of ways.

One limitation for both surface design and interactive simulations is that only a linear thin shell simulation can be performed interactively. The linear simulation results in physically accurate deformations as long as the overall deformations are small. For larger deformations this approach also yields plausible results in many cases, but may lead to distortion artifacts as demonstrated in [64]. An accurate simulation of larger deformations necessitates a non-linear [26] simulation, which requires repeated updating of the stiffness matrix, which is too time consuming for interactive use cases. Further, the topology of the subdivision surface used for deformation cannot change as this also requires repeated computation of the stiffness matrix.

A possible extension of the IGA framework for design and animation is to extend the thin shell implementation to perform a dynamic simulation of the shell behavior, e.g. as used by Clyde *et al.* [27], who describe their IGA implementation of thin shell dynamics using Catmull-Clark surfaces in the supplementary materials of their article. In contrast to the static simulation presented in this thesis, this would allow to simulate thin deformable objects in virtual worlds over time. However, a dynamic simulation is currently too slow for interactive use as it requires repeated evaluation of the stiffness matrix.

Another limitation is the speed of the computation. For complex subdivision meshes with many control points the computation is too slow for interactive deformation. Even neglecting the long precomputation time to setup the stiffness matrix, solving the system of equation alone is today still too time consuming on an average computer. The *Horse* example from Section 5.6 can still be deformed interactively, as shown in Table 5.1 and Figure 5.6, but solving the system of equations for meshes larger than this example is too slow. A more efficient approach to solving the equations on the GPU, e.g. as presented by Weber *et al.* [104], might enable deformation of larger models at interactive rates.

Another technique to improve performance is to use a coarse initial mesh and only refine regions of interest, where a more accurate result is needed. Therefore, an interesting research direction is to extend the NURBS compatible subdivision scheme to support local refinement. For subdivision surfaces, hierarchical subdivision techniques can be used to enable local refinement for analysis without changing the mesh topology, e.g. as demonstrated by Wei *et al.* [105].

For more accurate and more efficient analysis, the numerical integration could be further improved. Wawrzinek and Polthier [102] recently demonstrated how to exactly evaluate derivatives of Catmull-Clark surfaces near EVs. Extending this work to the NURBS compatible subdivision scheme would allow for a more accurate analysis and faster convergence. To improve performance and accuracy, Barendrecht *et al.* [8] presented a technique to reduce the number of quadrature points required for numerical integration by grouping several quad faces of the subdivision control mesh for integration.

The existing IGA framework could also be used to optimize the surface topology, similar to the work of Seo *et al.* [89]. During topology optimization geometry is removed in the design in a way which does not compromise e.g. the structural stability of the shape. Topology optimization is employed, for example, to save material in the manufacturing process of an object, or to make an object lighter. Two aspects in product design which become increasingly important.

Finally, an extension to volumetric representations would allow the simulation of more types of objects, not only thin shells. There is already existing work for IGA using Catmull-Clark solids by Burckhart *et al.* [17]. More recently Altenhofen *et al.* [3] demonstrated how a design process based on Catmull-Clark can be tightly integrated with a volumetric simulation by creating the volumetric information for classical FEM already during the design. It would be interesting to extend these approaches to IGA based on the hypothetical *NURBS compatible subdivision solids*, where the biggest challenge is to define the rules for this new volumetric subdivision scheme.

Part IV

APPENDIX

"An investment in knowledge always pays the best interest."

Benjamin Franklin

A TRANSFERRED DATA

The exact data sent and received for each API endpoint listed in Section 4.5.4 can be considered an implementation detail. Depending on the use case, different data formats are possible. For example, compressed binary data could be used to save bandwidth, or JSON encoded structures for ease of use in web-based applications. In the following, the binary format used for comparisons in Figure 5.6 and Figure 5.8 is shown in detail for the relevant core API endpoints from Section 4.5.4.1. The datatypes are abbreviated as `u32` for an unsigned 32 bit integer and `f32` for a 32 bit single precision floating-point number.

Using `POST /geometries` to upload new geometry, the subdivision control mesh is transmitted as raw binary data with a small header. The header contains the number of vertices n, the number of faces m, and the material parameters Young's Modulus E, Poisson Ratio ν and thickness t. Following the header, the control points and quad face indices are transmitted:

Header	u32 n	u32 m	f32 E	f32 ν	f32 t
Vertices	f32 x_1	f32 y_1	f32 z_1		
	...	...	...		
	f32 x_n	f32 y_n	f32 z_n		
Faces	u32 a_1	u32 b_1	u32 c_1	u32 d_1	
	...	...	...	...	
	u32 a_m	u32 b_m	u32 c_m	u32 d_m	

The response from the server is the unique `geom_id` transmitted as a sequence of 16 bytes.

Computing a new deformation with `POST /geometries/{geom_id}/results` requires the applied forces and constraints. Following the definitions in Section 3.1.5, clients can send a combination of v vertex displacement constraints, l Lagrange constraints and f force vectors:

u32 v	*vertex constraints data*
u32 l	*Lagrange constraints data*
u32 f	*force data*

The *vertex constraints data* consists of a sequence of dx_i, dy_i, dz_i displacement constraint values and a sequence of $mask_i$ and $index_i$ values:

Constraints	f32 dx_1	f32 dy_1	f32 dz_1
	...	...	...
	f32 dx_v	f32 dy_v	f32 dz_v
Masks & indices	u32 $mask_1$	u32 $index_1$	
	...	...	
	u32 $mask_v$	u32 $index_v$	

Constraint i is applied to control point $index_i$ of the control mesh, according to $mask_i$. The mask can be used to only enforce the constraint for certain coordinates, e.g. only for the x and z coordinates of the control point.

The *Lagrange constraints data* contains p control point indices c_i, $3p$ coefficients cx_i, cy_i, cz_i for all DOF of these control points, and 3 constraint values cvx, cvy, cvz for each of the l Lagrange constraints:

#Coefficients	u32 p_1		
Indices	u32 $c_{1,1}$	...	u32 c_{1,p_1}
	f32 $cx_{1,1}$	f32 $cy_{1,1}$	f32 $cz_{1,1}$
Coefficients	...	...	...
	f32 cx_{1,p_1}	f32 cy_{1,p_1}	f32 cz_{1,p_1}
Values	f32 cvx_1	f32 cvy_1	f32 cvz_1
	...	...	...
#Coefficients	u32 p_l		
Indices	u32 $c_{l,1}$	...	u32 c_{l,p_l}
	f32 $cx_{l,1}$	f32 $cy_{l,1}$	f32 $cz_{l,1}$
Coefficients	...	...	...
	f32 cx_{l,p_l}	f32 cy_{l,p_l}	f32 cz_{l,p_l}
Values	f32 cvx_l	f32 cvy_l	f32 cvz_l

Each of these Lagrange constraints adds 3 rows and columns to the system of equations, one for each component x, y, z of the affected control points, as described in Section 3.1.5.

Finally, *force data* is a sequence of three floats fx_i, fy_i, fz_i for the forces and a sequence of control point indices fp_i, to which the forces are applied:

Forces	f32 fx_1	f32 fy_1	f32 fz_1
	...	...	...
	f32 fx_f	f32 fy_f	f32 fz_f
Indices	u32 fp_1	...	u32 fp_f

After the deformation is computed, the server returns a monotonically increasing `res_id` and the updated control point positions for the deformed subdivision surface:

Result ID	u32 `res_id`		
Positions	f32 x_1	f32 y_1	f32 z_1
	...	...	...
	f32 x_n	f32 y_n	f32 z_n

The client uses the updated control points together with the unchanged face indices to visualize the deformed subdivision limit surface.

B PUBLICATIONS

Research presented in this thesis has been published in the following articles and conference papers:

A. Riffnaller-Schiefer, U. H. Augsdörfer, and D. W. Fellner, "Isogeometric analysis for modelling and design," in *EG 2015 - Short Papers*, B. Bickel and T. Ritschel, Eds., The Eurographics Association, 2015, pp. 17–20

A. Riffnaller-Schiefer, U. H. Augsdörfer, and D. W. Fellner, "Isogeometric shell analysis with NURBS compatible subdivision surfaces," *Applied Mathematics and Computation*, vol. 272, Part 1, pp. 139 –147, 2016, Subdivision, Geometric and Algebraic Methods, Isogeometric Analysis and Refinability, ISSN: 0096-3003

U. H. Augsdörfer and A. Riffnaller-Schiefer, "On the convergence of modeling and simulation," *IEEE Computer Graphics and Applications*, vol. 37, no. 4, pp. 8–13, 2017, ISSN: 0272-1716

A. Riffnaller-Schiefer, U. H. Augsdörfer, and D. W. Fellner, "Interactive physics-based deformation for virtual worlds," in *2017 International Conference on Cyberworlds (CW)*, 2017, pp. 88–95

C. Schinko, A. Riffnaller-Schiefer, U. Krispel, E. Eggeling, and T. Ullrich, "State-of-the-art overview on 3D model representations and transformations in the context of computer-aided design," *International Journal on Advances in Software*, vol. 10, p. 446, 2017, ISSN: 1942-2628

A. Riffnaller-Schiefer, U. H. Augsdörfer, and D. W. Fellner, "Physics-based deformation of subdivision surfaces for shared virtual worlds," *Computers & Graphics*, vol. 71, pp. 66 –76, 2018, ISSN: 0097-8493

Additionally, the following previous publication is also related to concepts applied in this thesis:

A. Schiefer, R. Berndt, T. Ullrich, V. Settgast, and D. W. Fellner, "Service-oriented scene graph manipulation," in *Proceedings of the*

15th International Conference on Web 3D Technology, ser. Web3D '10, Los Angeles, California: ACM, 2010, pp. 55–62, ISBN: 978-1-4503-0209-8

BIBLIOGRAPHY

[1] Aberdeen Group, *Simulation driven design benchmark report: Getting it right the first time*, Accessed 2018-12-20, Oct. 2006. [Online]. Available: http://www.reden.nl/bestanden/Aberdeen_Simulation_Driven_Design.pdf.

[2] R. S. Allison, L. R. Harris, M. Jenkin, U. Jasiobedzka, and J. E. Zacher, "Tolerance of temporal delay in virtual environments," in *Virtual Reality, 2001. Proceedings. IEEE*, IEEE, 2001, pp. 247–254.

[3] C. Altenhofen, F. Schuwirth, A. Stork, and D. Fellner, "Volumetric subdivision for consistent implicit mesh generation," *Computers & Graphics*, vol. 69, pp. 68 –79, 2017, ISSN: 0097-8493.

[4] U. H. Augsdörfer and A. Riffnaller-Schiefer, "On the convergence of modeling and simulation," *IEEE Computer Graphics and Applications*, vol. 37, no. 4, pp. 8–13, 2017, ISSN: 0272-1716.

[5] U. Augsdörfer, N. Dodgson, and M. Sabin, "Artifact analysis on B-splines, box-splines and other surfaces defined by quadrilateral polyhedra," *Computer Aided Geometric Design*, vol. 28, pp. 177–197, 2011.

[6] ——, "Artifact analysis on triangular box-splines and subdivision surfaces defined by triangular polyhedra," *Computer Aided Geometric Design*, vol. 28, pp. 198–211, 2011.

[7] P. J. Barendrecht, "Isogeometric analysis with subdivision surfaces," Master's thesis, Eindhoven University of Technology: Eindhoven, The Netherlands, 2013.

[8] P. J. Barendrecht, M. Bartoň, and J. Kosinka, "Efficient quadrature rules for subdivision surfaces in isogeometric analysis," *Computer Methods in Applied Mechanics and Engineering*, vol. 340, pp. 1 –23, 2018, ISSN: 0045-7825.

[9] Y. Bazilevs, V. Calo, J. Cottrell, J. Evans, T. Hughes, S. Lipton, M. Scott, and T. Sederberg, "Isogeometric analysis using T-splines," *Computer Methods in Applied Mechanics and Engineering*, vol. 199, pp. 229–263, 2010.

[10] L. Beirão da Veiga, A. Buffa, G. Sangalli, and R. Vázquez, "Analysis-suitable T-splines of arbitrary degree: Definition, linear independence and approximation properties," *Mathematical Models and Methods in Applied Sciences*, vol. 23, no. 11, pp. 1979–2003, 2013.

[11] T. Belytschko, H. Stolarski, W. K. Liu, N. Carpenter, and J. S. Ong, “Stress projection for membrane and shear locking in shell finite elements,” *Computer Methods in Applied Mechanics and Engineering*, vol. 51, no. 1-3, pp. 221 –258, 1985, ISSN: 0045-7825.

[12] C. Benthin, S. Boulos, D. Lacewell, and I. Wald, “Packet-based ray tracing of Catmull-Clark subdivision surfaces,” SCI Institute, University of Utah, Tech. Rep., 2007.

[13] H. Biermann, A. Levin, and D. Zorin, “Piecewise smooth subdivision surfaces with normal control,” in *SIGGRAPH 2000 Conference Proceedings: Computer Graphics Annual Conference Series*, Box1, 2000, pp. 113–120.

[14] W. Boehm, “Inserting new knots into b-spline curves,” *Computer-Aided Design*, vol. 12, no. 4, pp. 199 –201, 1980, ISSN: 0010-4485.

[15] C. de Boor, “On calculating with b-splines,” *Journal of Approximation Theory*, vol. 6, no. 1, pp. 50 –62, 1972, ISSN: 0021-9045.

[16] M. Botsch and O. Sorkine, “On linear variational surface deformation methods,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 1, pp. 213–230, Jan. 2008, ISSN: 1077-2626.

[17] D. Burckhart, B. Hamannand, and G. Umlauf, “Iso-geometric finite element analysis based on Catmull-Clark subdivision solids,” *Computer Graphics Forum*, vol. 29, no. 5, pp. 1575–1584, 2010.

[18] T. J. Cashman, U. H. Augsdörfer, N. A. Dodgson, and M. A. Sabin, “NURBS with Extraordinary Points: High-degree, Non-uniform, Rational Subdivision Schemes,” *ACM Transactions on Graphics*, vol. 28, no. 3, 46:1–46:9, Jul. 2009, ISSN: 0730-0301.

[19] T. J. Cashman, “NURBS-compatible subdivision surfaces,” University of Cambridge, Computer Laboratory, Tech. Rep. UCAM-CL-TR-773, Mar. 2010.

[20] H. Casquero, L. Liu, Y. Zhang, A. Reali, and H. Gomez, “Isogeometric collocation using analysis-suitable t-splines of arbitrary degree,” *Computer Methods in Applied Mechanics and Engineering*, vol. 301, pp. 164 –186, 2016, ISSN: 0045-7825.

[21] H. Casquero, L. Liu, Y. Zhang, A. Reali, J. Kiendl, and H. Gomez, “Arbitrary-degree t-splines for isogeometric analysis of fully nonlinear kirchhoff–love shells,” *Computer-Aided Design*, vol. 82, pp. 140 –153, 2017, Isogeometric Design and Analysis, ISSN: 0010-4485.

[22] E. Catmull and J. Clark, “Recursively generated B-spline surfaces on arbitrary topological meshes.,” *Computer Aided Design*, vol. 10, no. 6, pp. 183–188, 1978.

[23] G. Chaikin, "An algorithm for high speed curve generation," *Computer Graphics and Image Processing*, vol. 3, pp. 346–349, 1974.

[24] C. Chan, C. Anitescu, and T. Rabczuk, "Isogeometric analysis with strong multipatch C1-coupling," *Computer Aided Geometric Design*, vol. 62, pp. 294 –310, 2018, ISSN: 0167-8396.

[25] F. Cirak, M. Ortiz, and P. Schröder, "Subdivision surfaces: A new paradigm for thin-shell finite element analysis.," *Int. J. Numer. Meth. Eng.*, vol. 47, no. 12, pp. 2039–2072, 2000.

[26] F. Cirak and M. Ortiz, "Fully C1-conforming subdivision elements for finite deformation thin-shell analysis," *International Journal for Numerical Methods in Engineering*, vol. 51, no. 7, pp. 813–833, 2001.

[27] D. Clyde, J. Teran, and R. Tamstorf, "Modeling and data-driven parameter estimation for woven fabrics," in *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation*, ser. SCA '17, Los Angeles, California: ACM, 2017, 17:1–17:11, ISBN: 978-1-4503-5091-4.

[28] E. Cohen, T. Martin, R. Kirby, T. Lyche, and R. Riesenfeld, "Analysis-aware modeling: Understanding quality considerations in modeling for geometric analysis.," *Comput. Methods Appl. Mech. Engrg.*, vol. 199, pp. 334–356, 2010.

[29] M. T. Cook and A. Agah, "A survey of sketch-based 3-D modeling techniques," *Interacting with Computers*, vol. 21, no. 3, pp. 201 –211, 2009, ISSN: 0953-5438.

[30] J. A. Cottrell, T. J. Hughes, and Y. Bazilevs, *Isogeometric analysis: Toward integration of cad and fea*. John Wiley & Sons, Ltd, 2009, ISBN: 978-0-470-74873-2.

[31] M. G. Cox, "The numerical evaluation of b-splines," *IMA Journal of Applied Mathematics*, vol. 10, no. 2, pp. 134–149, 1972.

[32] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti, "Surround-screen projection-based virtual reality: The design and implementation of the cave," in *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '93, Anaheim, CA: ACM, 1993, pp. 135–142, ISBN: 0-89791-601-8.

[33] E. Dahlström, P. Dengler, A. Grasso, C. Lilley, C. McCormack, D. Schepers, J. Watt, J. Ferraiolo, J Fujisawa, and D. Jackson, "Scalable vector graphics (svg) 1.1," *World Wide Web Consortium Recommendation*, 2011.

[34] T. DeRose, M. Kass, and T. Truong, "Subdivision surfaces in character animation.," in *SIGGRAPH 1998 Conference Proceedings: Computer Graphics Annual Conference Series*, Box2, 1998, pp. 85–94.

[35] J. Deng, F. Chen, X. Li, C. Hu, W. Tong, Z. Yang, and Y. Feng, "Polynomial splines over hierarchical t-meshes," *Graphical Models*, vol. 70, no. 4, pp. 76 –86, 2008, ISSN: 1524-0703.

[36] D. Doo, "A subdivision algorithm for smoothing down irregularly shaped polyhedrons.," in *Proc. Int'l Conf. on Interactive Techniques in Computer Aided Design, IEEE Computer Soc.*, 1978, pp. 157–165.

[37] D. Doo and M. Sabin, "Behaviour of recursive division surfaces near extraordinary points.," *Computer Aided Design*, vol. 10, no. 6, pp. 177–181, 1978.

[38] G. Farin, *Curves and surfaces for CAGD*. Morgan Kaufmann, Oct. 22, 2001, ISBN: 9781558607378.

[39] C. Felippa, *Lecture notes on introduction to finite element methods*, Accessed 2018-09-18, 2018. [Online]. Available: http://www.colorado.edu/engineering/CAS/courses.d/IFEM.d/.

[40] J. Fish and T. Belytschko, *A first course in finite elements*. John Wiley & Sons, Ltd, 2007, ISBN: 9780470510858.

[41] C. Giannelli, B. Jüttler, and H. Speleers, "Thb-splines: The truncated basis for hierarchical splines," *Computer Aided Geometric Design*, vol. 29, no. 7, pp. 485 –498, 2012, Geometric Modeling and Processing 2012, ISSN: 0167-8396.

[42] C. Giannelli, B. Jüttler, S. K. Kleiss, A. Mantzaflaris, B. Simeon, and J. Špeh, "Thb-splines: An effective mathematical technology for adaptive refinement in geometric design and isogeometric analysis," *Computer Methods in Applied Mechanics and Engineering*, vol. 299, pp. 337 –365, 2016, ISSN: 0045-7825.

[43] H. Graf and A. Stork, "CAE/VR integration – a qualitative assessment of advanced visualization for interactive conceptual simulations (ICS) in industrial use," in *Virtual, Augmented and Mixed Reality: Applications in Health, Cultural Heritage, and Industry*, J. Y. Chen and G. Fragomeni, Eds., Cham: Springer International Publishing, 2018, pp. 260–271, ISBN: 978-3-319-91584-5.

[44] S. Green, "Multilevel, subdivision-based, thin shell finite elements: Development and an application to red blood cell modeling," PhD thesis, University of Washington, 2003.

[45] M. Halstead, M. Kass, and T. DeRose, "Efficient, fair interpolation using catmull-clark surfaces.," in *Computer Graphics Proceedings, Annual Conference Series, 1993. ACM SIGGRAPH*, box2, 1993, pp. 35–44.

[46] M. Hereld, I. R. Judson, and R. L. Stevens, "Introduction to building projection-based tiled display systems," *IEEE Computer Graphics and Applications*, vol. 20, no. 4, pp. 22–28, 2000.

[47] H. Hoppe, T. DeRose, T. Duchamp, M. Halstead, H. Jin, J. McDonald, J. Schweitzer, and W. Stützle, "Piecewise smooth surface reconstruction," *Computer Graphics (SIGGRAPH)*, vol. 1994, pp. 295–302, 1994.

[48] T. Hughes, J. Cottrell, and Y. Bazilevs, "Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement," *Comp. Methods Appl. Mech. Engrg.*, vol. 194, pp. 4135–4195, 2005.

[49] IGES/PDES Organization, *Initial graphics exchange specification - IGES 5.3*, 1996.

[50] B. Jüttler, A. Mantzaflaris, R. Perl, and M. Rumpf, "On numerical integration in isogeometric subdivision methods for PDEs on surfaces," *Computer Methods in Applied Mechanics and Engineering*, 2016, ISSN: 0045-7825.

[51] H. Kang, J. Xu, F. Chen, and J. Deng, "A new basis for pht-splines," *Graphical Models*, vol. 82, pp. 149 –159, 2015, ISSN: 1524-0703.

[52] J. Kiendl, K.-U. Bletzinger, J. Linhard, and R. Wüchner, "Isogeometric shell analysis with Kirchhoff-Love elements," *Computer Methods Appl. Engrg*, vol. 198, pp. 3902–3914, 2009.

[53] J. Kiendl, Y. Bazilevs, M.-C. Hsu, R. Wüchner, and K.-U. Bletzinger, "The bending strip method for isogeometric analysis of Kirchhoff-Love structures comprised of multiple patches," *Computer Methods Appl. Engrg*, vol. 199, pp. 2403–2416, 2010.

[54] G. R. Kirchhoff, *Über die gleichungen des gleichgewichtes eines elastischen körpers bei nicht unendlich kleinen verschiebungen seiner theile*. Wien, 1852, pp. 1 v. ; Sitzungsber. d. math.-naturw. Cl. d. Kaiserl. Akad. d. Wiss. [B.IX, T. 762].

[55] F. Komposch, "Physical modelling tool for Autodesk Maya 2016," Bachelor Thesis, Graz University of Technology, 2017.

[56] S.-J. Lee, "Solution of poisson equation using isogeometric formulation," *Architectural research*, vol. 13, no. 1, pp. 17–24, 2011.

[57] X. Li, J. Zheng, T. W. Sederberg, T. J. R. Hughes, and M. A. Scott, "On linear independence of T-spline blending functions," *Computer Aided Geometric Design*, vol. 29, no. 1, pp. 63 –76, 2012, ISSN: 0167-8396.

[58] Q. Long, P. Burkhard Bornemann, and F. Cirak, "Shear-flexible subdivision shells," *International Journal for Numerical Methods in Engineering*, vol. 90, no. 13, pp. 1549–1577, 2012, ISSN: 1097-0207.

[59] A. E. H. Love, "The small free vibrations and deformation of a thin elastic shell," *Philosophical Transactions of the Royal Society of London. A*, vol. 179, pp. 491–546, 1888.

[60] R. H. Macneal and R. L. Harder, "A proposed standard set of problems to test finite element accuracy," *Finite Elements in Analysis and Design*, vol. 1, no. 1, pp. 3 –20, 1985, ISSN: 0168-874X.

[61] A. Maglo, G. Lavoué, F. Dupont, and C. Hudelot, "3d mesh compression: Survey, comparisons, and emerging trends," *ACM Comput. Surv.*, vol. 47, no. 3, 44:1–44:41, Feb. 2015, ISSN: 0360-0300.

[62] C. Marrin, *WebGL specification*, Khronos WebGL Working Group, 2011.

[63] B. Marussig and T. J. R. Hughes, "A review of trimming in isogeometric analysis: Challenges, data exchange and simulation aspects," *Archives of Computational Methods in Engineering*, 2017, ISSN: 1886-1784.

[64] A. Nealen, M. Müller, R. Keiser, E. Boxerman, and M. Carlson, "Physically based deformable models in computer graphics," *Computer Graphics Forum*, vol. 25, no. 4, pp. 809–836, 2006, ISSN: 1467-8659.

[65] T. Nguyen, K. Karčiauskas, and J. Peters, "A comparative study of several classical, discrete differential and isogeometric methods for solving poisson's equation on the disk," *Axioms*, vol. 3, no. 2, pp. 280–299, 2014, ISSN: 2075-1680.

[66] L. Olsen, F. F. Samavati, M. C. Sousa, and J. A. Jorge, "Sketch-based modeling: A survey," *Computers & Graphics*, vol. 33, no. 1, pp. 85 –103, 2009, ISSN: 0097-8493.

[67] J. Peng, C.-S. Kim, and C.-C. J. Kuo, "Technologies for 3d mesh compression: A survey," *Journal of Visual Communication and Image Representation*, vol. 16, no. 6, pp. 688 –733, 2005, ISSN: 1047-3203.

[68] J. Peters and U. Reif, "Analysis of algorithms generalizing b-spline subdivision," *SIAM J of Numerical Analysis*, vol. 35, pp. 728–748, 1998.

[69] J. Peters, "Patching Catmull-Clark meshes," in *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '00, New York, NY, USA: ACM Press/Addison-Wesley Publishing Co., 2000, pp. 255–258, ISBN: 1-58113-208-5.

[70] R. Riesenfeld, "On chaikin's algorithm," *CGIP*, vol. 4, pp. 304–310, 1975.

[71] A. Riffnaller-Schiefer, U. H. Augsdörfer, and D. W. Fellner, "Isogeometric shell analysis with NURBS compatible subdivision surfaces," *Applied Mathematics and Computation*, vol. 272, Part 1, pp. 139 –147, 2016, Subdivision, Geometric and Algebraic Methods, Isogeometric Analysis and Refinability, ISSN: 0096-3003.

[72] ——, "Interactive physics-based deformation for virtual worlds," in *2017 International Conference on Cyberworlds (CW)*, 2017, pp. 88–95.

[73] ——, "Physics-based deformation of subdivision surfaces for shared virtual worlds," *Computers & Graphics*, vol. 71, pp. 66 –76, 2018, ISSN: 0097-8493.

[74] A. Riffnaller-Schiefer, U. H. Augsdörfer, and D. W. Fellner, "Isogeometric analysis for modelling and design," in *EG 2015 - Short Papers*, B. Bickel and T. Ritschel, Eds., The Eurographics Association, 2015, pp. 17–20.

[75] M. A. Sabin, "Spline finite elements," PhD thesis, University of Leeds, UK, 1997, p. 169.

[76] M. Sabin, "Eigenanalysis and artifacts of subdivision curves and surfaces.," in *Tutorials on Multiresolution in Geometric Modelling*, A. Iske, E. Quak, and M. S. Floater, Eds., Springer, 2002, ch. 4, pp. 69–92.

[77] M. Sabin, U. Augsdörfer, and N. Dodgson, "Artifacts in box-spline surfaces.," in *IMA Conference on the Mathematics of Surfaces XI, 11th IMA International Conference, Loughborough, UK, September 5-7, 2005, Proceedings*, R. R. Martin, H. E. Bez, and M. A. Sabin, Eds., ser. Lecture Notes in Computer Science, vol. 3604, Springer, 2005, pp. 350–363, ISBN: 3-540-28225-4.

[78] M. Sabin and L. Barthe, "Artifacts in recursive subdivision surfaces.," in *Curve and Surface Fitting: St Malo 2002*, A. Cohen, J. L. Merrien, and L. L. Schumaker, Eds., box4: Nashboro Press, Brentwood, TN, 2003, pp. 353–362.

[79] A. Saxena and B. Sahay, *Computer aided engineering design*. Springer, Apr. 1, 2005, 416 pp., ISBN: 978-1-4020-2555-6.

[80] A. Schiefer, R. Berndt, T. Ullrich, V. Settgast, and D. W. Fellner, "Service-oriented scene graph manipulation," in *Proceedings of the 15th International Conference on Web 3D Technology*, ser. Web3D '10, Los Angeles, California: ACM, 2010, pp. 55–62, ISBN: 978-1-4503-0209-8.

[81] C. Schinko, A. Riffnaller-Schiefer, U. Krispel, E. Eggeling, and T. Ullrich, "State-of-the-art overview on 3D model representations and transformations in the context of computer-aided design," *International Journal on Advances in Software*, vol. 10, p. 446, 2017, ISSN: 1942-2628.

[82] J. E. Schweitzer, "Analysis and application of subdivision surfaces," PhD thesis, University of Washington, 1996, ISBN: 0-591-11119-5.

[83] M. Scott, R. Simpson, J. Evans, S. Lipton, S. Bordas, T. Hughes, and T. Sederberg, "Isogeometric boundary element analysis using unstructured T-splines," *Computer Methods in Applied Mechanics and Engineering*, vol. 254, pp. 197 –221, 2013, ISSN: 0045-7825.

[84] M. A. Scott, M. J. Borden, C. V. Verhoosel, T. W. Sederberg, and T. J. R. Hughes, "Isogeometric finite element data structures based on Bézier extraction of T-splines," *Int. J. Numer. Meth. Eng.*, vol. 88, no. 2, pp. 126–156, 2011, ISSN: 1097-0207.

[85] M. A. Scott, "T-splines as a design-through-analysis technology," PhD thesis, THE UNIVERSITY OF TEXAS AT AUSTIN, 2011.

[86] T. Sederberg, J. Zheng, A. Bakenov, and A. Nasri, "T-splines and t-nurccs," *ACM Transactions on Graphics*, vol. 22, no. 3, pp. 477–484, Jul. 2003, ISSN: 0730-0301.

[87] T. Sederberg, D. Cardon, G. Finnigan, N. North, J. Zheng, and T. Lyche, "T-spline simplification and local refinement.," *ACM Transactions on Graphics*, vol. 23, no. 3, pp. 276–283, 2004.

[88] T. W. Sederberg, G. T. Finnigan, X. Li, H. Lin, and H. Ipson, "Watertight trimmed NURBS," *ACM Trans. Graph.*, vol. 27, no. 3, 79:1–79:8, Aug. 2008, ISSN: 0730-0301.

[89] Y.-D. Seo, H.-J. Kim, and S.-K. Youn, "Shape optimization and its extension to topological design based on isogeometric analysis," *International Journal of Solids and Structures*, vol. 47, no. 11-12, pp. 1618 –1640, 2010, ISSN: 0020-7683.

[90] W. Slaughter, *The linearized theory of elasticity*. Birkhäuser Boston, 2012, ISBN: 9781461200932.

[91] J. Stam, "Exact evaluation of Catmull-Clark subdivision surfaces at arbitrary parameter values.," in *Proceedings of SIGGRAPH 1998*, box5, 1998, pp. 395–404.

[92] J. Stam, "Evaluation of Loop subdivision surfaces," in *SIGGRAPH Course Notes*, 1999.

[93] ——, "On subdivision schemes generalizing uniform b-spline surfaces of arbitrary degree," *Computer Aided Geometric Design*, vol. 18, no. 5, pp. 383–396, 2001.

[94] J. Stoer and R. Bulirsch, *Introduction to numerical analysis*, 3rd ed. Springer, 2002.

[95] I. Stroud, *Boundary representation modelling techniques*. Springer Science & Business Media, 2006.

[96] R. L. Taylor and S. Govindjee, "Solution of clamped rectangular plate problems," *Communications in Numerical Methods in Engineering*, vol. 20, no. 10, pp. 757–765, 2004, ISSN: 1099-0887.

[97] S. Timoshenko and S. Woinowsky-Krieger, *Theory of plates and shells*, ser. Engineering societies monographs. McGraw-Hill, 1959.

[98] E. Ventsel and T. Krauthammer, *Thin plates and shells: Theory: Analysis, and applications*. CRC press, 2001.

[99] V Vukicevic, B Jones, K Gilbert, and C. Wiemeersch, *WebVR*, World Wide Web Consortium, 2016.

[100] W. Wang, Y. Zhang, M. A. Scott, and T. J. R. Hughes, "Converting an unstructured quadrilateral mesh to a standard t-spline surface," *Computational Mechanics*, vol. 48, no. 4, pp. 477–498, 2011, ISSN: 1432-0924.

[101] W. Wang, Y. Zhang, G. Xu, and T. J. R. Hughes, "Converting an unstructured quadrilateral/hexahedral mesh to a rational t-spline," *Computational Mechanics*, vol. 50, no. 1, pp. 65–84, 2012, ISSN: 1432-0924.

[102] A. Wawrzinek and K. Polthier, "Integration of generalized B-spline functions on Catmull-Clark surfaces at singularities," *Computer-Aided Design*, vol. 78, pp. 60 –70, 2016, SPM 2016, ISSN: 0010-4485.

[103] A. Wawrzinek, K. Hildebrandt, and K. Polthier, "Koiter's thin shells on Catmull-Clark limit surfaces," in *Proceedings of the Vision, Modeling, and Visualization Workshop 2011*, 2011, pp. 113–120.

[104] D. Weber, J. Bender, M. Schnoes, A. Stork, and D. Fellner, "Efficient GPU data structures and methods to solve sparse linear systems in dynamics applications," *Computer Graphics Forum*, vol. 32, no. 1, pp. 16–26, 2013, ISSN: 1467-8659.

[105] X. Wei, Y. Zhang, T. J. R. Hughes, and M. A. Scott, "Truncated hierarchical Catmull-Clark subdivision with local refinement," *Computer Methods in Applied Mechanics and Engineering*, vol. 291, pp. 1 –20, 2015, ISSN: 0045-7825.

[106] O. C. Zienkiewicz and R. L. Taylor, *The finite element method: Solid mechanics*. Butterworth-Heinemann, 2000, vol. 2.

[107] D. Zorin and P. Schröder, "A unified framework for primal/dual quadrilateral subdivision schemes," *CAGD*, vol. 18, pp. 429–454, 2001.