MASTER THESIS

# Isogeometric Analysis with Catmull-Clark Subdivision Surfaces

Ayhan Demirel

Chair of Structural Analysis and Dynamics
Prof. Dr.-Ing. habil. Roland Wüchner
Technical University of Munich

# Isogeometric Analysis with Catmull-Clark Subdivision Surfaces

## Master's Thesis

Submitted to the Department of Civil, Geo and Environmental Engineering
in partial fulfillment of the requirements for the degree of
M.Sc.
at the Technical University of Munich.


**Supervised by** M.Sc. Bastian Devresse
M.Sc. Ricky Aristio
Prof. Dr.-Ing. habil. Roland Wüchner
Chair of Structural Analysis and Dynamics

**Submitted by** Ayhan Demirel
Am Oberwiesenfeld 9
80809 München
Student number: 03768086


**Submitted on** Munich, 30/12/2024

# Abstract

This thesis focuses on the implementation of Catmull-Clark subdivision rules for isogeometric analysis applied to thin shells. The formulation of the NURBS-based Kirchhoff-Love shell element for isogeometric analysis was utilized. Modeling complex geometries with arbitrary topologies may face limitations when using NURBS surfaces due to their nature as tensor products of two NURBS curves. Sometimes, CAD models are very complex and consist of multiple NURBS patches that may not be well joined during the design phase. It is crucial to analyze these models carefully to ensure that any unmatched patches are addressed properly and that the geometries are watertight. Furthermore, adaptive mesh refinement techniques cannot be applied to NURBS because local refinement is not feasible. Starting from the control mesh of a surface, the subdivision process creates smooth surfaces through a limiting procedure of repeated refinements. At the limit, the Catmull-Clark surfaces are represented by bi-cubic B-Spline curves. Due to their ability to effectively represent objects with arbitrary topologies, the subdivision surfaces have gained importance in geometric modeling and structural analysis. Extended subdivision methods also make it possible to model both smooth objects and those with boundaries and sharp edge features. The extended Catmull-Clark subdivision algorithm was implemented in Python and adapted for Kratos Multiphysics. The discussion focused on the challenges associated with applying boundary support conditions, and a proposed solution for these conditions was tested. Some benchmark geometries utilizing regular elements were analyzed, and convergence analyses were conducted for both subdivision-based and NURBS-based approaches to perform comparison. Finally, the current achievements and challenges related to the evaluation of geometries with irregular elements were examined.

## Keywords

CAD; FEM; Isogeometric Analysis; Kirchhoff-Love Shell Theory; Catmull-Clark Subdivision

# Kurzfassung

Diese Arbeit konzentriert sich auf die Implementierung der Catmull-Clark-Subdivision-Regeln für die isogeometrische Analyse, angewandt auf dünne Schalen. Dabei wurde die Formulierung des NURBS-basierten Kirchhoff-Love-Schalenelements für die isogeometrische Analyse genutzt. Das Modellieren komplexer Geometrien mit beliebigen Topologien kann beim Einsatz von NURBS-Flächen aufgrund ihrer Eigenschaft als Tensorprodukte zweier NURBS-Kurven auf Einschränkungen stoßen. CAD-Modelle sind oft sehr komplex und bestehen aus mehreren NURBS-Patches, die während der Entwurfsphase möglicherweise nicht gut miteinander verbunden sind. Es ist entscheidend, diese Modelle sorgfältig zu analysieren, um sicherzustellen, dass alle nicht übereinstimmenden Patches ordnungsgemäß behandelt werden und die Geometrien wasserdicht sind. Darüber hinaus können adaptive Netzverfeinerungstechniken nicht auf NURBS angewendet werden, da lokale Verfeinerungen nicht möglich sind. Ausgehend vom Kontrollnetz einer Fläche erzeugt der Subdivisionsprozess durch ein Grenzverfahren wiederholter Verfeinerungen glatte Flächen. Im Grenzfall werden die Catmull-Clark-Flächen durch bi-kubische B-Spline-Kurven dargestellt. Aufgrund ihrer Fähigkeit, Objekte mit beliebigen Topologien effektiv darzustellen, haben Subdivisionsflächen in der geometrischen Modellierung und Strukturanalyse an Bedeutung gewonnen. Erweiterte Subdivisionsmethoden ermöglichen es außerdem, sowohl glatte Objekte als auch solche mit Rändern und scharfen Kanten zu modellieren. Der erweiterte Catmull-Clark-Subdivision-Algorithmus wurde in Python implementiert und für Kratos Multiphysics angepasst. Die Diskussion konzentrierte sich auf die Herausforderungen im Zusammenhang mit der Anwendung von Randstützbedingungen, und eine vorgeschlagene Lösung für diese Bedingungen wurde getestet. Einige Benchmark-Geometrien, die reguläre Elemente verwenden, wurden analysiert, und Konvergenzanalysen wurden sowohl für Subdivisions-basierte als auch für NURBS-basierte Ansätze durchgeführt, um Vergleiche anzustellen. Abschließend wurden die aktuellen Errungenschaften und Herausforderungen im Zusammenhang mit der Bewertung von Geometrien mit irregulären Elementen untersucht.

## Schlüsselwörter

CAD; FEM; Isogeometrische Analyse; Kirchhoff-Love-Schalentheorie; Catmull-Clark Subdivision

# Table of Contents

# 1. Introduction

In nature, all objects around us are described as three-dimensional bodies. However, they are not always defined as such in engineering applications but are described in reduced dimensions. For instance, an object with a much larger dimension than the other two is usually modeled as a one-dimensional model. Similarly, if an object has two significantly larger dimensions than the other dimension, it is modeled as a two-dimensional model. Some examples of one-dimensional models are trusses and beams, and two-dimensional models are plates, membranes, and shells. A shell refers to a thin-walled structure that can have any curvature within a three-dimensional space, and membranes and plates are special cases of shells. The former is called a plane shell, which primarily carries in-plane forces. In contrast, the latter is called a plane shell, which carries out-of-plane bending loads, typically with negligible thickness [1].

As the dimensional complexity of the models increases, the mathematical complexity of the models also grows. For instance, plates are more complex than trusses and beams, and shells are more complex than plates. Interestingly, describing a shell model is more challenging than a fully three-dimensional continuum model due to the fact that the general continuum mechanics equations can be directly applied without concerning the shape of a three-dimensional solid, whereas a shell requires a precise mathematical description for its geometry as well as specific properties such as curvature [1]. That's why shell theories are still in development today, and their importance comes from the widespread presence of shell structures in nature and technology. This is because such structures can carry loads through their shapes, enabling highly efficient designs for material savings and weight reduction. The curvature of such structures enables them to carry transverse loads via tension and compression while minimizing bending moments, which results in efficient material use. Today, shells are extensively used in industries such as aerospace, automotive, civil engineering, and architecture [1].

The origins of the shell theory come from the first plate theory developed by Gustav R. Kirchhoff in 1850 [2], also called "the classical plate theory". This theory still remains the foundation of various plate calculation tables used today in civil engineering. In 1888, August E.H. Love built upon Kirchhoff's assumption to develop a shell theory [3], today known as Kirchhoff-Love theory. In the shell theory, another widely used theory is the Reissner-Mindlin theory, which also accounts for transverse shear deformation, unlike the Kirchhoff-Love theory [4]. The transverse shear deformation is significant for thick shells, which satisfy the slenderness condition of $R/t < 20$, but insignificant for thin shells. The Reissner-Mindlin theory remains dominant in FEA (Finite Element Analysis) while the Kirchhoff-Love theory is widely used for the practical use of thin shells [1].

In 2005, Hughes et al. [5, 6] introduced the term "IGA" to bridge the gap between the CAD

(Computer Aided Design) and CAE (Computer Aided Engineering), enabling both disciplines to use the same geometric description. In FEA, the structural domain is divided into a finite number of small, basic geometries, with elements defined by a set of nodal points and shape functions associated with them. To achieve greater geometric flexibility and higher continuity between the elements, NURBS were introduced as basis functions for the analysis, leading to what is known as IGA. In the CAD world, NURBS are widely used and meet the essential criteria of basis functions to perform analysis without any geometry conversion [1]. Therefore, NURBS-based IGA has been effectively applied to study solids, shells, fluids, fluid-structure interaction, turbulence, and structural optimization [1]. A shell element formulation based on Kirchhoff-Love shell theory was introduced by J. Kiendl [1, 7], utilizing NURBS as the basis function. It has been demonstrated that using NURBS ensures the necessary $C^1$ continuity among the elements for Kirchhoff-Love shells. The use of NURBS makes it possible for easy implementation of Kirchhoff-Love theory, unlike standard FEA that utilizes high-order polynomials.

Subdivision surfaces and curves have become increasingly important in geometric modeling applications because they can effectively represent objects with arbitrary topologies. This flexibility makes it easier to design, render, and manipulate these objects [8]. Subdivision surfaces enable modeling smooth objects and those with boundaries and sharp features through extended subdivision rules [8]. Representing objects with non-planar topologies or sharp features is impossible without applying curve trimming for B-Splines or NURBS, which are prominent representations in the CAD world [8]. Nevertheless, the piecewise polynomial (or rational) structure of B-Splines and NURBS simplifies the analysis of their representation. On the other hand, the subdivision surfaces are defined as the limit surface results from continuously refining a 3D control mesh through multiple iterations. The obtained limit surfaces are directly influenced by the applied subdivision scheme, and these schemes can be divided into two categories in the mathematical geometric modeling literature: interpolating schemes and approximating schemes. This thesis focuses on the use of the Catmull-Clark subdivision scheme [9], which is an approximating subdivision method that produces a smooth limit surface through iterative refinement. The scheme works by generating quadrilateral meshes at each subdivision step.

In literature, the first application that uses subdivision surfaces to analyze Kirchhoff-Love shells with FEA was introduced by Cirak et al.[10]. The research conducted by Cirak et al. utilized the subdivision algorithm through Loop's scheme, an approximating method that generates triangular meshes at each subdivision step. However, it was noted that this approach could also be implemented using the Catmull-Clark subdivision scheme [10]. Recent studies have also focused on applying the Catmull-Clark subdivision to IGA, highlighting the growing significance of subdivision surfaces in architectural design and geometric modeling programs supported by common CAD packages [11]. For instance, the investigation of Catmull-Clark subdivision solids based volumetric IGA was studied by Hamann et al. [12]. The truncated hierarchical Catmull-Clark subdivision method developed by Wei et al. [13] allows for local

refinement of geometry and facilitates the generalization of arbitrary topology for truncated hierarchical B-splines. Other studies focused on solving linear and non-linear PDEs on planes based on extended subdivision rules were studied in [14, 15].

This thesis aims to adapt Catmull-Clark subdivision rules for isogeometric Kirchhoff-Love shell analysis within the Kratos Multiphysics [16, 17, 18] framework by implementing the subdivision algorithm and testing it on benchmark examples. During this work, the isogeometric Kirchhoff-Love shell element formulation introduced by J. Kiendl [1, 7] is used, which is already implemented in "$IgaApplication$" of Kratos Multiphysics. As noted in [1], the developed shell element formulation is not exclusive to NURBS, but is generally applicable to displacement-based Kirchhoff-Love shell theory, allowing its use with subdivision-based IGA. The thesis is organized as follow:

- $Section\ 2$ provides an overview of the basics of geometry, including the mathematical descriptions of curves and surfaces, geometric and parametric continuity, and the differential geometry of surfaces.
- $Section\ 3$ reviews the structural mechanics of shells, emphasizing the fundamentals of continuum mechanics and the Kirchhoff-Love shell theory.
- $Section\ 4$ provides a summary of the isogeometric analysis and the NURBS-based Kirchhoff-Love shell element formulation as presented by J. Kiendl [1].
- $Section\ 5$ provides information about subdivision surfaces by first introducing the concept of subdivision surfaces in general, then detailing the Catmull-Clark subdivision algorithm, and finally presenting the extended Catmull-Clark subdivision algorithm.
- $Section\ 6$ outlines the implementation details of the subdivision algorithm in Kratos Multiphysics and explains how the analyses are conducted. It also discusses issues related to the implementation of boundary conditions and the methods used to resolve these issues and provides an update on the progress made with irregular elements.
- $Section\ 7$ presents the results from various benchmark examples.
- $Section\ 8$ provides the conclusion of the work and suggestions for the future implementation as an outlook.

# 2. Fundamentals of Geometry

This section briefly describes the mathematical description of the curves and surfaces. The formula for the mathematical description of free-form curves and surfaces is presented first. Then, the fundamental formulations of differential geometry of the surfaces are presented, which is the basis for shell kinematics.

## 2.1. Mathematical Representation of Curves and Surfaces

In mathematics, there are different ways of describing curves and surfaces. In this section, the explicit, implicit, and parametric description with the focus on curves is presented. These representations have some advantages and disadvantages over them, and they can describe derivatives, continuities, and geometrical properties differently from each other for analysis purposes [1].

### 2.1.1. Explicit, Implicit, and Parametric Representation of Curves and Surfaces

The simplest way of describing a curve is the explicit representation, which is given in the formula: $y = f(x)$ for curves and $z = f(x, y)$ for surfaces. As can be seen from these formulas, the coordinates depend on each other functionally. Although explicit representation enables easy calculation of derivatives and geometric properties like curvature, it is the most limited representation type. The reason is the explicit representation allows a minimal amount of curve description because each point in the $x$-coordinate only takes one $y$-value. Additionally, this representation depends on the axis, meaning that a quadratic interpolant through three points differs for each coordinate system. That's why this representation is rarely used in CAD nowadays [1].

The implicit representation of curves is given in the form of $f(x, y) = 0$, and for surfaces: $f(x, y, z) = 0$. With this representation, it is possible for each $x$-value to have more than one $y$-value, which enables drawing curves like circles. In this representation method, it is still easy to determine whether a point lies in a curve. However, it may be more complicated to find the intersection points of two curves. Although this representation allows more curve description, it is still limited. Both explicit and implicit representations encounter difficulties in providing an exact analytic description of curves and surfaces [1].

The parametric description of curves, which is the most convenient representation for free-form geometries, is given in the form of an explicit representation of coordinates $x$,$y$, and $z$ to an independent parameter, usually denoted as $t$. For the surfaces, there are two independent parameters. Compared to the other two representations, this gives the most flexible

and variable geometric descriptions, which also allows the representation of space curves as well. Since the independent parameter can be defined over an interval $a \leq t \leq b$, the curve has the property of having a start and end point, which is not the case for the other two representations. Nevertheless, this representation has a drawback, which is that finding intersections of a point and the curve or of two curves is usually difficult. The examples of the three representation types are given in Figure 2.1.

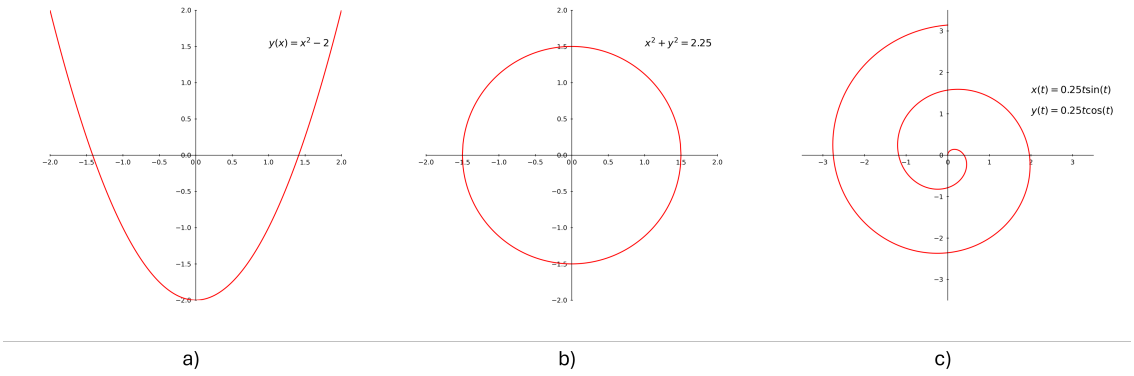


**Figure 2.1** Curve representations: a) Explicit b) Implicit c) Parametric

## 2.2. Overview of Bézier, B-Spline, and NURBS Curves and Surfaces

B-Splines originated from Bézier curves, and NURBS from B-Splines. That's why Bézier curves are introduced in this chapter first. Then, B-Spline is explained in more detail since it contains most of the definitions from NURBS. Finally, NURBS is explained in more detail.

### 2.2.1. Bézier Curves

In the curve description, data points are fitted with either interpolating polynomials or approximation curves. The initial can represent each data point on the curve; however, it may create oscillations. A Bézier curve approximates the data points or control points, creating a non-oscillating smooth curve inside the control polygon, which is obtained by the linear connection of the control points. The mathematical expression of Bézier curves is given as:

$$\mathbf{C}(\xi) = \sum_{i=1}^{n} B_{i,p}(\xi)\mathbf{P}_i \tag{2.1}$$

where the number of control points and the Bernstein polynomials of polynomial degree $p$ are denoted as $n$, $B_{i,p}(\xi)$ respectively. The polynomial degree in the Bernstein polynomials depends on the number of control points: $p = n-1$. The Bernstein polynomials are expressed as [19]:

$$B_{i,p}(\xi) = \frac{n!}{i!(n-i)!}\xi^i(1-\xi)^{n-i} \tag{2.2}$$

where $\xi$ is defined as $\xi \in [0, 1]$.

Since the polynomial degree depends on the number of control points, increasing the control point would result in an increase of the polynomial degree, which leads to deviation of the approximation curve from the control polygon. Another drawback of Bézier curve is that any change on the control points influences the whole curve, which doesn't allow local changes on the curve.

### 2.2.2. B-Spline Curves and Surfaces

To overcome the limitations of Bézier curve, B-Spline curves can be used. As in the Bézier curves, B-Spline curves are also defined by a linear combination of control points and basis functions within a parametric space. The basis functions are called Basis-Splines, or in short B-Splines, and they are defined over intervals in the parametric space having specific conditions between the intervals. In B-Spline curves, the number of intervals is flexible, which allows the selection of polynomial degrees independent of the number of control points. Hence, approximation of a large data set can be done with lower polynomial degrees [1].

The parametric space for B-Splines is determined by a knot vector $\Xi = [\xi_1, \xi_2, \xi_3, ..., \xi_{n+p+1}]$, which consists of parametric coordinates $\xi_i$ arranged in non-descending order. A knot vector partitions the parametric space into sections, and it is referred to as uniform when the knots are evenly spaced. Within each knot span, a B-Spline basis function is $C^{\infty}$ continuous, while at a single knot, it has $C^{p-1}$ continuity. Multiple knot is the case when a knot occurs multiple times, and a knot with multiplicity $k$ has continuity of $C^{p-k}$, meaning that increasing the multiplicity of a knot reduces its continuity [1].

Open knot vector is the case when the first and last knot has a multiplicity of $p + 1$, which means the first and last control points of a B-Spline are interpolated such that the curve is tangential to the control points at these ends [20]. Noticing that starting and end points have to be specified in CAD applications, which open knot vector allows this and becomes a standard in such applications.

The B-Spline basis functions are given in the form of $Cox - deBoor$ recursion formula as for $p = 0$:

$$N_{i,0}(\xi) = \begin{cases} 1, & \xi_i \leqslant \xi < \xi_{i+1} \\ 0, & otherwise \end{cases} \tag{2.3}$$

For $p \geqslant 1$:

$$N_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi) \tag{2.4}$$

A B-Spline curve having a polynomial degree $p$, and its first derivative are mathematically

expressed as:

$$\mathbf{C}(\xi) = \sum_{i=1}^{n} N_{i,p}(\xi)\mathbf{P}_i \tag{2.5}$$

$$\mathbf{C}'(\xi) = \sum_{i=1}^{n} N'_{i,p}(\xi)\mathbf{P}_i \tag{2.6}$$

To sum up, the important characteristics of B-Spline curves are [1]:

- Curve is located inside the convex hull of the control points.
- Control points are generally approximated.
- Each control point influences a maximum of p + 1 sections.
- A special case of a B-Spline curve having only one knot interval is Bézier curve.

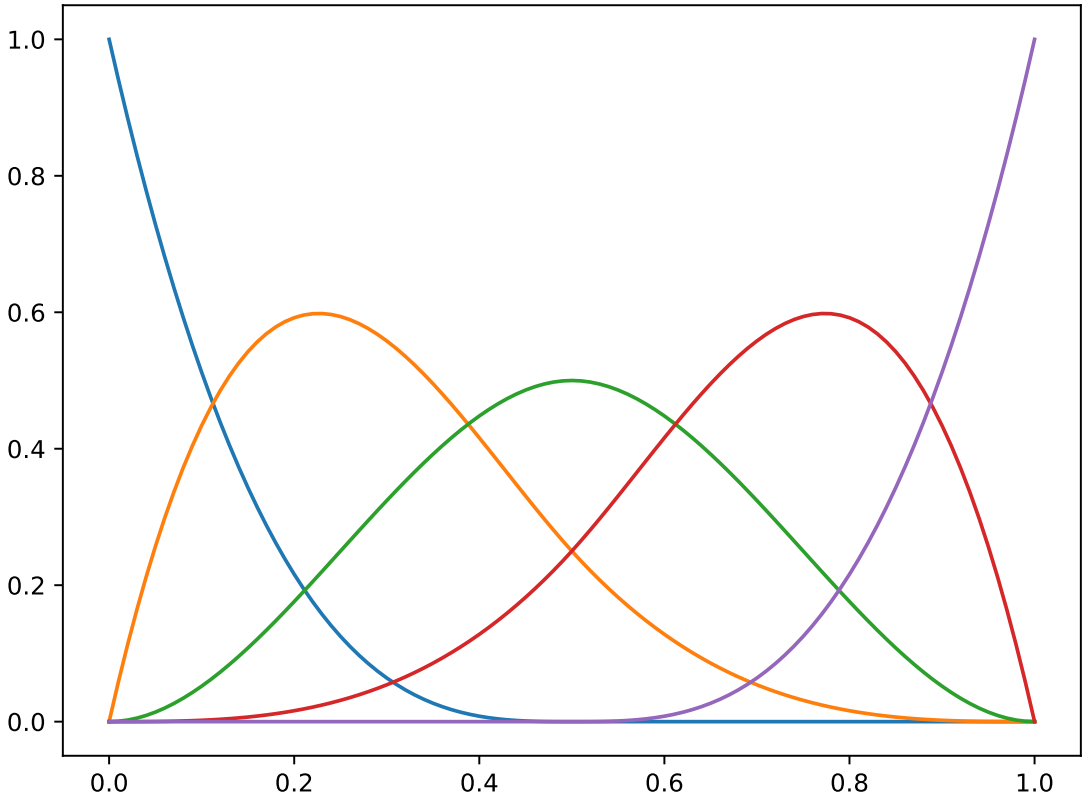


**Figure 2.2** Cubic B-Spline basis function with open knot vector $\Xi = [0, 0, 0, 0, 0.5, 1, 1, 1, 1]$

A B-Spline surface is calculated through the tensor product of B-Spline basis functions across two parametric dimensions, $\xi$ and $\eta$. The surface is defined by a grid of $n \times m$ control points, two knot vectors $\Xi$ and $H$, and two polynomial degrees $p$ and $q$. The corresponding basis functions are $N_{i,p}(\xi)$ and $M_{j,q}(\eta)$.The surface is represented by the following expression[1]:

$$\mathbf{S}(\xi, \eta) = \sum_{i=1}^{n} \sum_{j=1}^{m} N_{i,p}(\xi) M_{j,q}(\eta) \mathbf{P}_{i,j} \tag{2.7}$$

Similarly, a B-Spline solid can be obtained in parametric dimensions, $\xi, \eta$ and $\zeta$ as [1]:

$$\mathbf{B}(\xi, \eta, \zeta) = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{l} N_{i,p}(\xi) M_{j,q}(\eta) L_{k,r}(\zeta) \mathbf{P}_{i,j,k} \tag{2.8}$$

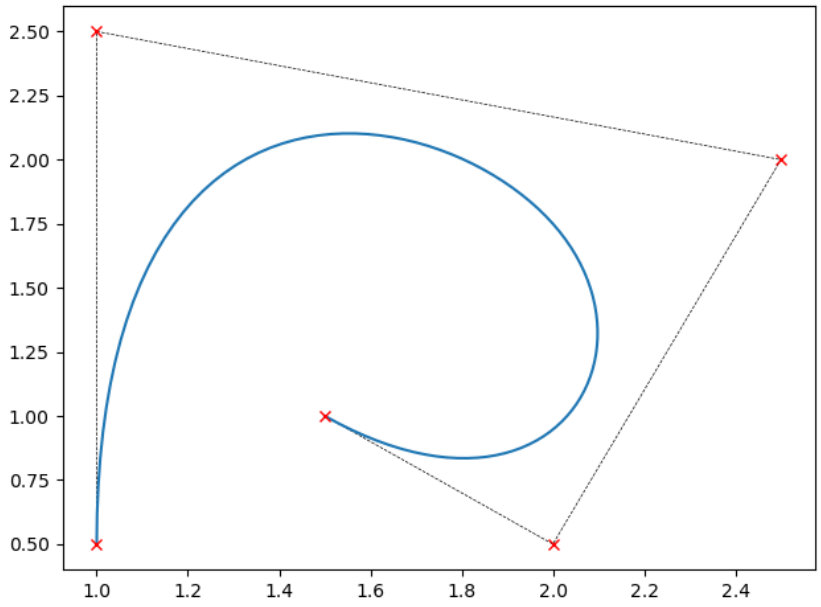


**Figure 2.3** B-Spline Curve

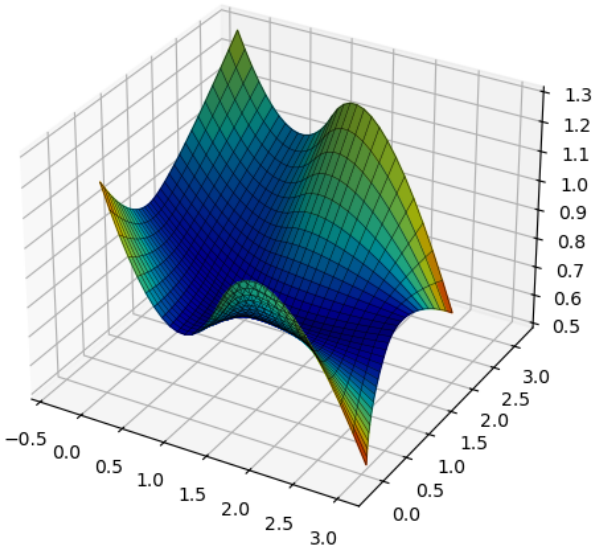


**Figure 2.4** B-Spline Surface

### 2.2.3. NURBS Curves and Surfaces

The difference between B-Splines and NURBS comes from the fact that basis functions in the initial are piecewise polynomials, although they are piecewise rational polynomials for the latter. In NURBS, the control points are represented with an additional weight, $w_i$, assigned to each of them. The mathematical expression of a NURBS curve is given as:

$$\mathbf{C}(\xi) = \frac{\sum_{i=1}^{n} N_{i,p}(\xi) w_i \mathbf{P}_i}{\sum_{i=1}^{n} N_{i,p}(\xi) w_i} \tag{2.9}$$

and basis function for NURBS can be defined as:

$$R_{i,p}(\xi) = \frac{N_{i,p}(\xi) w_i}{\sum_{i=1}^{n} N_{i,p}(\xi) w_i} \tag{2.10}$$

such that the NURBS curve formula can be rewritten as:

$$\mathbf{C}(\xi) = \sum_{i=1}^{n} R_{i,p}(\xi) \mathbf{P}_i \tag{2.11}$$

A special case of the NURBS having the same weight for all control points is what we know as the B-Splines. Thus, all the properties of the B-Spline are also valid for the NURBS. The importance of the weights comes from the fact that a curve can be located close to a control point by increasing the corresponding weight. In addition, conical sections, such as circles and ellipses, can be exactly represented with the NURBS basis functions, which allows the representation of smooth free-form geometries, sharp edges, kinks, spheres, cylinders, etc.[1]. This is why NURBS is the most common geometrical representation method in CAD modeling today.

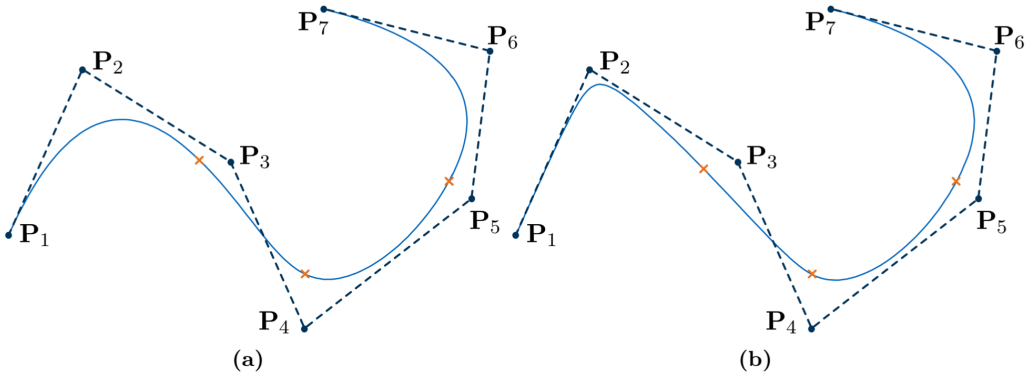


**Figure 2.5** Representation of a: a) B-Spline curve. b) NURBS curve with increased weight at $P_2$. (Taken from Wüchner et al. [21])

As in the definition of B-Spline surfaces and solids, NURBS surfaces and solids can be represented similarly in the equations 2.12 and 2.14 with the basis functions defined in the equa-

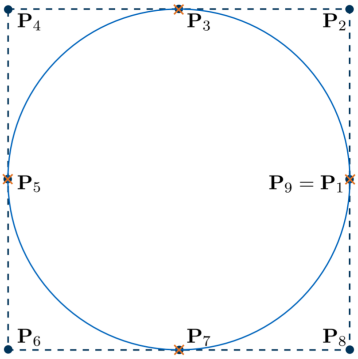


**Figure 2.6** Representation of a circle with NURBS (Taken from Wüchner et al. [21])

tions 2.13 and 2.15 respectively as:

$$\mathbf{S}(\xi,\eta) = \sum_{i=1}^{n}\sum_{j=1}^{m} R_{i,j}^{p,q}(\xi,\eta)\mathbf{P}_{i,j} \tag{2.12}$$

$$R_{i,j}^{p,q}(\xi,\eta) = \frac{N_{i,p}(\xi)M_{j,q}(\eta)w_{i,j}}{\sum_{i=1}^{n}\sum_{j=1}^{m} N_{i,p}(\xi)M_{j,q}(\eta)w_{i,j}} \tag{2.13}$$

$$\mathbf{B}(\xi,\eta,\zeta) = \sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l} R_{i,j,k}^{p,q,r}(\xi,\eta,\zeta)\mathbf{P}_{i,j,k} \tag{2.14}$$

$$R_{i,j,k}^{p,q,r}(\xi,\eta,\zeta) = \frac{N_{i,p}(\xi)M_{j,q}(\eta)L_{k,r}(\zeta)w_{i,j,k}}{\sum_{i=1}^{n}\sum_{j=1}^{m}\sum_{k=1}^{l} N_{i,p}(\xi)M_{j,q}(\eta)L_{k,r}(\zeta)w_{i,j,k}} \tag{2.15}$$

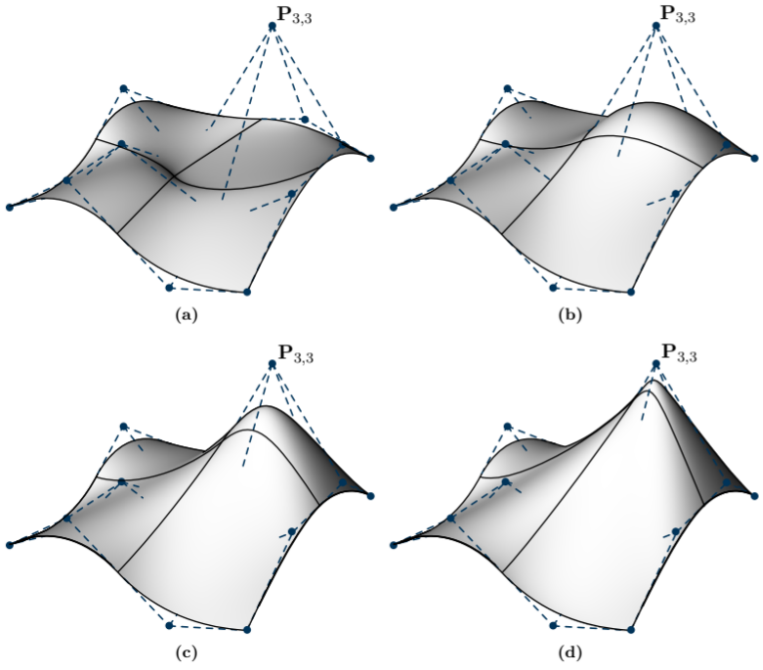


**Figure 2.7** Representation of a NURBS surface with different weights at $P_{3,3}$ (Taken from Wüchner et al. [21])

The two primary mesh refinement methods for a NURBS curve or surface are knot insertion and order elevation. These techniques result in an improvement in the design space of the geometry with the addition of control points. The knot insertion is a method in which new knots are inserted to divide the knot span into smaller sections, which reduces the continuity by one at the location where the new knot is inserted. Also, an additional control point is introduced for each newly added knot. The order elevation is a method in which the polynomial degree of the basis functions is increased without changing the number of knot intervals, resulting in the repetition of the existing knots to maintain the same continuity at those points. As a result of these methods, the geometry or the parametrization is not altered [1].

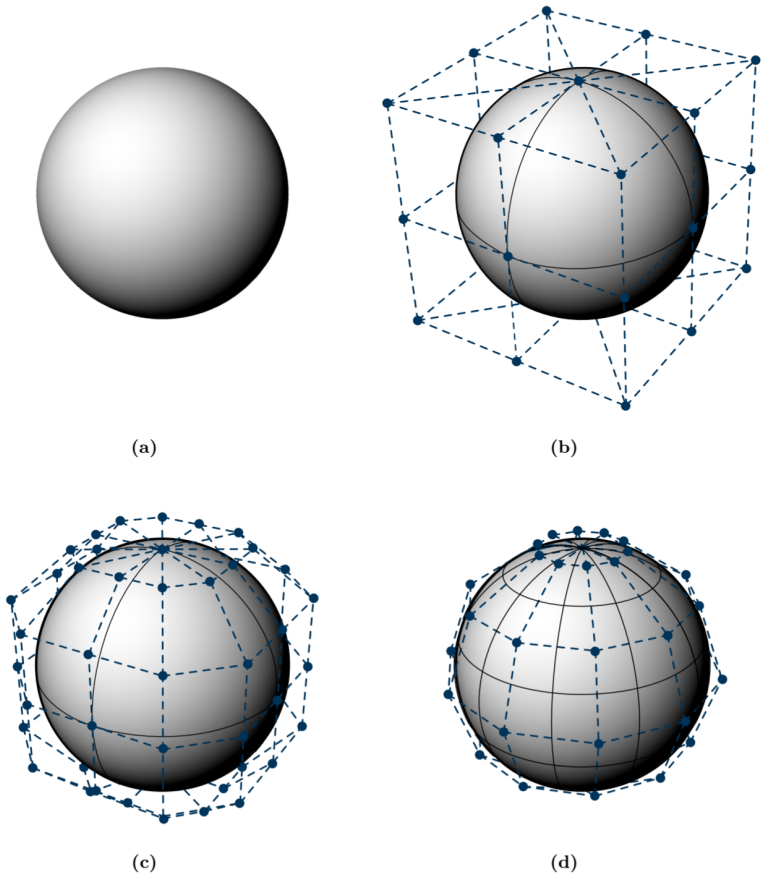


**Figure 2.8** Refinement of a NURBS solid: a) Sphere. b) Initial NURBS representation. c) Order elevation. d) Knot insertion. (Taken from Wüchner et al. [21])

## 2.3. Geometric and Parametric Continuity

There are two types of continuity for curves and surfaces defined by parametric representation, namely geometric and parametric continuity. The parametric and geometric continuity represents the same thing for the zeroth order continuity, i.e. $G^0 = C^0$, but this is not the case for the order equal or larger than 1. In this case, the parametric continuity $C^k$ also represents the geometric continuity $G^k$, but this is not the case for vice versa[1].

For two curves, $\mathbf{C}^1(\xi)$ and $\mathbf{C}^2(\xi)$ and $0 \leq \xi \leq 1$, joining from the end points such that $\mathbf{C}^1(1) = \mathbf{C}^2(0)$, the parametric continuity at the first order, $C^1$, is defined as:

$$\frac{\partial \mathbf{C}^1(1)}{\partial \xi} = \frac{\partial \mathbf{C}^2(0)}{\partial \xi} \tag{2.16}$$

such that both curves have the same first derivative at the joint point. In this case, the tangent vectors at the joint are parallel and possess the same magnitude. For the geometric continuity at the first level, $G^1$, the tangent vectors do not need to have the same magnitude, but at least they have to be parallel such that [22]:

$$\frac{\partial \mathbf{C}^1(1)}{\partial \xi} = c \cdot \frac{\partial \mathbf{C}^2(0)}{\partial \xi} \tag{2.17}$$

where c is any scalar multiplier.

For the surfaces, the conditions on the equations 2.16 and 2.17 must be satisfied for both parametric directions $\xi$ and $\eta$ to have parametric and geometric continuity along a common edge respectively [7].

## 2.4. Fundamentals of Differential Geometry of Surfaces

This section aims to review the fundamental aspects of differential geometry of surfaces, which further form the foundation for the structural shell model explained in Chapter 3.

In three-dimensional space, each point of a curve or surface can be mathematically expressed with their position vector $\mathbf{x}$ as:

$$\mathbf{x} = x^1\mathbf{e}_1 + x^2\mathbf{e}_2 + x^3\mathbf{e}_3 = x^i\mathbf{e}_i \tag{2.18}$$

where the global Cartesian base vectors and their coordinates are denoted as $\mathbf{e}_i$ and $x^i$ respectively. In this equation, the Einstein summation convention is used. Noticing that the indices take the Latin numbers for the case {1,2,3} and the Greek letters for {1,2} [1].

It is beneficial to use curvilinear coordinate systems and local bases to describe free-form geometries, particularly for surfaces. Two fundamental bases in differential geometry are the covariant basis $\mathbf{g}_i$ and the contravariant basis $\mathbf{g}^i$. A position vector $\mathbf{x}$ can be rewritten by using curvilinear coordinates with the corresponding contravariant coordinates $\theta_i$ and covariant coordinates $\theta_i$ as:

$$\mathbf{x} = \theta^i\mathbf{g}_i = \theta_i\mathbf{g}^i \tag{2.19}$$

where the covariant base vectors are defined as the partial derivatives of the position vector

$\mathbf{x}$ with respect to the corresponding contravariant coordinates $\theta^i$:

$$\mathbf{g}_i = \frac{\partial \mathbf{x}}{\partial \theta^i} = \mathbf{x}_{,i} \tag{2.20}$$

In addition, the covariant and contravariant base vectors are dependent on each other with the following condition:

$$\mathbf{g}_i \cdot \mathbf{g}^j = \delta_i^j = \begin{cases} 0 & \text{if } i \neq j \\ 1 & \text{if } i = j \end{cases} \tag{2.21}$$

A surface is a two-dimensional geometry that can be parametrically defined, with each point being defined by two curvilinear coordinates $(\theta_1, \theta_2)$. For the surfaces, the first two base vectors in covariant coordinates $\mathbf{g}_\alpha$ can be computed as in the equation 2.20 while the third covariant base vector can be computed from:

$$\mathbf{g}_3 = \frac{\mathbf{g}_1 \times \mathbf{g}_2}{|\mathbf{g}_1 \times \mathbf{g}_2|} \tag{2.22}$$

such that it is a normalized vector orthogonal to $\mathbf{g_1}$ and $\mathbf{g_2}$. In addition, the contravariant and covariant of the third base vectors are equal:

$$\mathbf{g}^3 = \mathbf{g}_3 \tag{2.23}$$

due to the fact that the contravariant base vectors $\mathbf{g}^\alpha$ reside in the tangential plane formed by the covariant base vectors $\mathbf{g}_\alpha$.

A local Cartesian basis, which is an orthogonal and normalized basis having an arbitrary orientation, can be established by using the base vectors $\mathbf{g}_1$ and $\mathbf{g}_2$. The definition of such a basis is given as follows:

$$\mathbf{e}_1 = \frac{\mathbf{g}_1}{\|\mathbf{g}_1\|} \tag{2.24}$$

$$\mathbf{e}_2 = \frac{\mathbf{g}_2 - (\mathbf{g}_2 \cdot \mathbf{e}_1)\mathbf{e}_1}{\|\mathbf{g}_2 - (\mathbf{g}_2 \cdot \mathbf{e}_1)\mathbf{e}_1\|} \tag{2.25}$$

$$\mathbf{e}_3 = \mathbf{g}_3 \tag{2.26}$$

The metric tensor $\mathbf{g}$ plays a vital role in defining the geometry of surfaces; it is sometimes referred to as the identity tensor. It can be characterized in both covariant and contravariant bases as follows:

$$\mathbf{g} = g^{\alpha\beta}\mathbf{g}_\alpha \otimes \mathbf{g}_\beta = g_{\alpha\beta}\mathbf{g}^\alpha \otimes \mathbf{g}^\beta \tag{2.27}$$

where the covariant metric coefficients $g_{\alpha\beta}$ can be computed by scalar product of covariant base vectors as [23]:

$$g_{\alpha\beta} = \mathbf{g}_\alpha \cdot \mathbf{g}_\beta \tag{2.28}$$

The equation mentioned above is referred to as the *first form of surfaces*, which holds essen-

tial details concerning the geometry of the surface, including the lengths of the base vectors and the angle between them. In order to get the contravariant metric coefficients $g^{\alpha\beta}$, one can take the inverse of the covariant coefficient matrix:

$$\left[g^{\alpha\beta}\right] = \left[g_{\alpha\beta}\right]^{-1} \tag{2.29}$$

In addition, one can obtain covariant base vectors from the covariant metric coefficients and contravariant base vectors as follow:

$$\mathbf{g}^{\alpha} = g^{\alpha\beta}\mathbf{g}_{\alpha} \tag{2.30}$$

Similarly:

$$\mathbf{g}_{\alpha} = g_{\alpha\beta}\mathbf{g}^{\alpha} \tag{2.31}$$

The curvature information of the surface geometries is defined by the *second fundamental form of surfaces*, and the curvature tensor coefficient can be expressed as [23]:

$$b_{\alpha\beta} = -\mathbf{g}_{\alpha} \cdot \mathbf{g}_{3,\beta} = -\mathbf{g}_{\beta} \cdot \mathbf{g}_{3,\alpha} = \mathbf{g}_{\alpha,\beta} \cdot \mathbf{g}_{3} \tag{2.32}$$

# 3. Fundamentals of Shell Structural Mechanics

## 3.1. Continuum Mechanics: An Overview

This section aims to summarize the fundamentals of continuum mechanics and derive some important kinematics equations for shell structures using the differential geometry quantities defined in Section 2.4. In the following parts, small strains with large displacements are assumed with the Lagrangian description.

### 3.1.1. Kinematics

In continuum mechanics, the deformation of a body is defined by kinematics, and it is crucial to differentiate between the reference (undeformed) and actual (deformed) states of the body on a material point [1]. The quantities in the reference state are denoted in upper case, and lower case is used for the actual state.

In material point, the deformation $\mathbf{u}$ is denoted with the position vectors in the actual and reference configuration states as:

$$\mathbf{u} = \mathbf{x} - \mathbf{X} \tag{3.1}$$

A deformation gradient $\mathbf{F}$ is used to map a differential line element from the reference state $d\mathbf{X}$ to a line element in the actual state $d\mathbf{x}$ as in the following equation [24]:

$$d\mathbf{x} = \mathbf{F} \cdot d\mathbf{X} \tag{3.2}$$

This deformation gradient can be expressed using the base vectors in both reference and actual state as follows [25]:

$$\begin{aligned} \mathbf{F} &= \mathbf{g}_i \otimes \mathbf{G}^i & \mathbf{F}^T &= \mathbf{G}^i \otimes \mathbf{g}_i \\ \mathbf{F}^{-1} &= \mathbf{G}_i \otimes \mathbf{g}^i & \mathbf{F}^{-T} &= \mathbf{g}^i \otimes \mathbf{G}_i \end{aligned} \tag{3.3}$$

and the mapping between deformed and undeformed base vectors can be done with the deformation gradient [1]:

$$\begin{aligned} \mathbf{g}_i &= \mathbf{F} \cdot \mathbf{G}_i & \mathbf{G}_i &= \mathbf{F}^{-1} \cdot \mathbf{g}_i \\ \mathbf{g}^i &= \mathbf{F}^{-T} \cdot \mathbf{G}^i & \mathbf{G}^i &= \mathbf{F}^T \cdot \mathbf{g}^i \end{aligned} \tag{3.4}$$

The overall deformation of a body with the rigid body motion is expressed with the deformation gradient, which is not directly suitable for strain measurements. Hence, one needs to use different strain measurements, i.e. using the Green-Lagrange strain tensor $\mathbf{E}$. This tensor is

appropriate for strains caused by large deformations due to its ability to describe the nonlinear relations between deformation and strains. The Green-Lagrange strain tensor is expressed in terms of the deformation gradient and the identity tensor $\mathbf{I}$ as:

$$\begin{aligned}\mathbf{E} = \frac{1}{2}(\mathbf{F}^T \cdot \mathbf{F} - \mathbf{I}) &= E_{ij}\mathbf{G}^i \otimes \mathbf{G}^j \\ &= \frac{1}{2}(g_{ij} - G_{ij})\mathbf{G}^i \otimes \mathbf{G}^j\end{aligned} \tag{3.5}$$

where the Green-Lagrange strain coefficients $E_{ij}$ are defined by the reference and actual states as:

$$E_{ij} = \frac{1}{2}(g_{ij} - G_{ij}) \tag{3.6}$$

corresponding to the undeformed configuration of the contravariant basis $\mathbf{G}^i \otimes \mathbf{G}^j$ [1].

### 3.1.2. Constitutive Equations

The relationship between stress and strain, as described by material laws, is expressed through constitutive equations. In the same way, as for the strain tensor, there are various definitions of the stress tensor. For instance, the second Piola-Kirchhoff (PK2) stress tensor $\mathbf{S}$ is the conjugate of the Green-Lagrange strain tensor $E$ in terms of energy [25], which is derived from the strain energy $W^{int}$ as:

$$\mathbf{S} = \frac{\partial W^{int}}{\partial \mathbf{E}} \tag{3.7}$$

A material tensor $\mathbf{C}$, or an elasticity tensor, is a fourth-order tensor, which relates the stress and strain tensors as:

$$\mathbf{C} = \frac{\partial \mathbf{S}}{\partial \mathbf{E}} = \frac{\partial^2 W^{int}}{\partial \mathbf{E}^2} \tag{3.8}$$

The stress and strain can be linearly related by using a St.Venant-Kirchhoff material model, which holds the following assumptions:

$$\mathbf{S} = \mathbf{C} : \mathbf{E} \tag{3.9}$$

$$S^{ij} = C^{ijkl}E_{kl} \tag{3.10}$$

$$\mathbf{S} = S^{ij}\mathbf{G}_i \otimes \mathbf{G}_j \tag{3.11}$$

An isotropic elastic material can be characterized by just two independent parameters. However, various conventions are used depending on the application. In engineering, Young's modulus $E$ and Poisson's ratio $\nu$ are most often used, while in mathematical contexts, the Lamé parameters $\lambda$ and $\mu$ are more common. The Lamé constants are related to the Young's modulus $E$ and the Poisson's ratio $\nu$ as follows:

$$\lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} \quad \mu = \frac{E}{2(1+\nu)} \tag{3.12}$$

Although the energetically conjugate of the Green-Lagrange strain tensor is the second Piola-Kirchhoff stress tensor $\mathbf{S}$, it does not indicate the physical stress state, which is actually defined by the Cauchy stress tensor $\boldsymbol{\sigma}$. The relationship between the Cauchy and the PK2 stress tensor is given as follows:

$$\boldsymbol{\sigma} = (det\mathbf{F})^{-1} \cdot \mathbf{F} \cdot \mathbf{S} \cdot \mathbf{F}^{T} \tag{3.13}$$

$$\mathbf{S} = det\mathbf{F} \cdot \mathbf{F}^{-1} \cdot \boldsymbol{\sigma} \cdot \mathbf{F}^{-T} \tag{3.14}$$

The first Piola-Kirchhoff (PK1) stress tensor $\mathbf{P}$ is another stress tensor used commonly, and it is expressed as:

$$\mathbf{P} = det\mathbf{F} \cdot \boldsymbol{\sigma} \cdot \mathbf{F}^{-T} = \mathbf{F} \cdot \mathbf{S} \tag{3.15}$$

### 3.1.3. Equilibrium Equations

The balance between internal and external forces is represented by the equilibrium equations. The mathematical expression of the equilibrium condition in the reference state is given as:

$$div\mathbf{P} + \rho_0\mathbf{B} = div(\mathbf{F} \cdot \mathbf{S}) + \rho_0\mathbf{B} = \mathbf{0} \tag{3.16}$$

where the $\rho_0$ and $\mathbf{B}$ represent the density and the body force vector in the reference state, respectively [1].

The strong form of the boundary value problem, in this case, is defined with equations 3.5 3.9 and 3.16 having appropriate boundary conditions. Numerical methods, such as FEM (Finite Element Method), are utilized to address the strong form of equations in the context of most three-dimensional problems because feasible solutions may not exist. In FEM, the governing equations and boundary conditions are satisfied in an integral sense rather than at every point, which makes the resulting equilibrium equation the weak form of the problem. During the development of a NURBS-based Kirchhoff-Love shell element, the Principle of Virtual Work, specifically the Principle of Virtual Displacements, was taken as a basis in [1]. According to this principle, the total virtual work done by the internal and external forces when an infinitesimal virtual displacement is applied to the system, which is the sum of internal and external virtual works, equals zero if the system is under equilibrium [26]:

$$\delta W = \delta W_{int} + \delta W_{ext} = 0 \tag{3.17}$$

where the internal and external virtual works are expressed as:

$$\delta W_{int} = -\int_{\Omega} \delta\mathbf{E} : \mathbf{S}\, d\Omega \tag{3.18}$$

$$\delta W_{ext} = \int_{\Gamma} \mathbf{T} \cdot \delta\mathbf{u}\, d\Gamma + \int_{\Omega} \rho\mathbf{B} \cdot \delta\mathbf{u}\, d\Omega \tag{3.19}$$

In these equations, the domain, the domain boundary, and the body vector forces are represented with $\Omega$, $\Gamma$, and $\mathbf{T}$ respectively [1].

## 3.2. The Fundamentals of Kirchhoff-Love Shell Theory

The Kirchhoff-Love shell theory uses what is known as a "direct approach," which means that it considers the shell as a two-dimensional surface right from the beginning rather than being developed from three-dimensional continuum mechanics. The model integrates kinematic assumptions to represent three-dimensional behavior, characterizing the thickness of the shell by a vector field, known as the director, on the middle surface [1].

Normal stresses and strains through the direction of thickness are not considered in the shell theories that developed from the direct approach. In the Kirchoff-Love shell theory, the strain is distributed linearly through the thickness of the shell according to the assumption that cross sections remain straight during deformations. In addition, the the theory assumes that the cross section perpendicular to the middle surface remains perpendicular even in the deformed state, which means the director always remains perpendicular to the middle surface. Hence, the shell can be expressed completely by its middle surface. The assumption of the perpendicularity of the cross section and the middle surface means that the transverse shear strains are neglected, which is an important assumption for thin structures. The validity of this assumption is determined by the shell slenderness, such that if the shell slenderness is $R/t > 20$, the assumption is valid, where $R$ is the radius of curvature and $t$ is the shell thickness.[1].

Due to the assumption of neglecting the transversal normal and transversal shear strains, the equation 3.5 can be written only with the in-plane strain coefficients as:

$$\mathbf{E} = E_{\alpha\beta}\mathbf{G}^{\alpha} \otimes \mathbf{G}^{\beta} \tag{3.20}$$

where the strain coefficients become:

$$E_{\alpha\beta} = \frac{1}{2}(g_{\alpha\beta} - G_{\alpha\beta}) \tag{3.21}$$

The middle surface of the shell is denoted by $\mathbf{x}(\theta^3 = 0)$ where the $t$ is the shell thickness within the thickness coordination ranging from $(-0.5t \leq \theta^3 \leq 0.5t)$. Thus, the base vectors of the middle surface $\mathbf{a}_i$ are obtained via the following relations:

$$\mathbf{a}_{\alpha} = \mathbf{x}_{,\alpha}(\theta^3 = 0) \tag{3.22}$$

$$\mathbf{a}_3 = \frac{\mathbf{a}_1 \times \mathbf{a}_2}{|\mathbf{a}_1 \times \mathbf{a}_2|} \tag{3.23}$$

The metric and curvature coefficients of the shell middle surface are given based on the

equations 2.28 and 2.32 as:

$$a_{\alpha\beta} = \mathbf{a}_\alpha \cdot \mathbf{a}_\beta \tag{3.24}$$

$$b_{\alpha\beta} = -\mathbf{a}_\alpha \cdot \mathbf{a}_{3,\beta} = -\mathbf{a}_\beta \cdot \mathbf{a}_{3,\alpha} = \mathbf{a}_{\alpha,\beta} \cdot \mathbf{a}_3 \tag{3.25}$$

Thus, the position vector $\mathbf{x}$ within the shell continuum becomes:

$$\mathbf{x} = \theta^\alpha \mathbf{a}_\alpha + \theta^3 \mathbf{a}_3 \tag{3.26}$$

and the base vectors $\mathbf{g}_a$ and the metric coefficients $g_{\alpha\beta}$ are expressed as:

$$\mathbf{g}_\alpha = \mathbf{a}_\alpha + \theta^3 \mathbf{a}_{3,\alpha} \tag{3.27}$$

$$\begin{aligned} g_{\alpha\beta} &= (\mathbf{a}_\alpha + \theta^3 \mathbf{a}_{3,\alpha}) \cdot (\mathbf{a}_\beta + \theta^3 \mathbf{a}_{3,\beta}) \\ &= a_{\alpha\beta} - 2\theta^3 b_{\alpha\beta} + (\theta^3)^2 \mathbf{a}_{3,\alpha} \cdot \mathbf{a}_{3,\beta} \end{aligned} \tag{3.28}$$

The quadratic terms with $\theta^3$ in the equation 3.28 can be neglected for thin and moderately thick shells according to [24]:

$$g_{\alpha\beta} = a_{\alpha\beta} - 2\theta^3 b_{\alpha\beta} \tag{3.29}$$

Hence, the equation 3.21 can be rewritten as:

$$E_{\alpha\beta} = \frac{1}{2}(a_{\alpha\beta} - A_{\alpha\beta}) + \theta^3 (B_{\alpha\beta} - b_{\alpha\beta}) \tag{3.30}$$

The strains in the shell continuum are expressed in terms of the metric and curvature coefficients of the middle surface according to the equation 3.30. The strains can be expressed with a constant term and a linear term, in which the constant terms represent the strains within the middle surface according to the membrane action. As a result, the membrane strains $\varepsilon_{\alpha\beta}$ are defined as [1]:

$$\varepsilon_{\alpha\beta} = \frac{1}{2}(a_{\alpha\beta} - A_{\alpha\beta}) \tag{3.31}$$

The effect of the bending represents the change in curvature, and it is denoted by the linear part of the equation, which is symmetric with respect to the middle surface. This change of curvature can be expressed as:

$$\kappa_{\alpha\beta} = B_{\alpha\beta} - b_{\alpha\beta} \tag{3.32}$$

As a result, the equation 3.30 can be rewritten as:

$$E_{\alpha\beta} = \varepsilon_{\alpha\beta} + \theta^3 \kappa_{\alpha\beta} \tag{3.33}$$

The stress applied to the shell can be separated into membrane and bending actions as in the case of strains, which gives the resultant stresses of normal forces $\mathbf{n}$ and bending moments $\mathbf{m}$. These stress resultants can be formulated as follows due to a linear stress distribution

through thickness [1]:

$$S^{\alpha\beta} = C^{\alpha\beta\gamma\delta} E_{\gamma\delta} \tag{3.34}$$

$$n^{\alpha\beta} = \int_{-t/2}^{t/2} S^{\alpha\beta}\, d\theta^3 = C^{\alpha\beta\gamma\delta} \varepsilon_{\gamma\delta} \cdot t \tag{3.35}$$

$$m^{\alpha\beta} = \int_{-t/2}^{t/2} S^{\alpha\beta} \cdot \theta^3\, d\theta^3 = C^{\alpha\beta\gamma\delta} \kappa_{\gamma\delta} \cdot \frac{t^3}{12} \tag{3.36}$$

There exists three independent stress and strain coefficients, $S^{11}$, $S^{22}$, $S^{12}$, $E^{11}$, $E^{22}$, and $E^{12}$, respectively due to the symmetry of both the stress and strain tensors. Thus, the constitutive relation can be rewritten in Voigt notation as [1]:

$$\begin{bmatrix} S^{11} \\ S^{22} \\ S^{12} \end{bmatrix} = \tilde{\mathbf{D}} \cdot \begin{bmatrix} E_{11} \\ E_{22} \\ 2E_{12} \end{bmatrix} \tag{3.37}$$

where $\tilde{\mathbf{D}}$ denotes the material matrix. In general, physical parameters like Young's modulus $E$ are used to obtain the material matrix. In addition, the stress and strain values in the equation 3.39 must be converted into a local Coordinate system because the quantities in this equation are represented in normalized units. Thus, the strain coefficients can be transformed into a local Coordinate basis, where the upper bar indicates this basis as:

$$\bar{E}_{\gamma\delta} = E_{\alpha\beta} \left(\mathbf{E}_\gamma \cdot \mathbf{G}^\alpha\right) \left(\mathbf{G}^\beta \cdot \mathbf{E}_\delta\right) \tag{3.38}$$

where $\mathbf{E}_\gamma$ and $\mathbf{E}_\delta$ are the local Cartesian vectors obtained from equations 2.24 and 2.25. These vectors are from the reference configuration [1].

In the following equations, an upper bar notation ($\bar{}$) represents the components on a local Cartesian basis. In the following equation, the PK2 stress coefficients $\bar{S}^{\alpha\beta}$ are calculated with a material matrix $\mathbf{D}$ using physical components:

$$\begin{bmatrix} \bar{S}^{11} \\ \bar{S}^{22} \\ \bar{S}^{12} \end{bmatrix} = \tilde{\mathbf{D}} \cdot \begin{bmatrix} \bar{E}_{11} \\ \bar{E}_{22} \\ 2\bar{E}_{12} \end{bmatrix} \tag{3.39}$$

For an isotropic material and an orthotropic material, the material matrices are defined as follows:

$$\mathbf{D}^{iso} = \frac{E}{1-\nu^2} \begin{bmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix} \tag{3.40}$$

$$\mathbf{D}^{ort} = \begin{bmatrix} \frac{E_1}{1-\nu_{12}\nu_{21}} & \frac{\nu_{21}E_1}{1-\nu_{12}\nu_{21}} & 0 \\ \frac{\nu_{12}E_2}{1-\nu_{12}\nu_{21}} & \frac{E_2}{1-\nu_{12}\nu_{21}} & 0 \\ 0 & 0 & G_{12} \end{bmatrix} \tag{3.41}$$

where $E$ denotes Young's modulus and $\nu$ denotes Poisson's ratio. Noticing that there are different Young moduli and Poisson's ratios $\nu_{12}$ and $\nu_{21}$ exist such that the $\nu_{21}E_1 = \nu_{12}E_2$ is satisfied to ensure the material matrix symmetry for the orthotropic material [1].

The tensors of normal and bending stress resultants are also symmetric, and one can calculate the coefficients of them with material matrix and Voigt notation as [1]:

$$\begin{bmatrix} \bar{n}^{11} \\ \bar{n}^{22} \\ \bar{n}^{12} \end{bmatrix} = t \cdot \mathbf{D} \cdot \begin{bmatrix} \bar{\varepsilon}_{11} \\ \bar{\varepsilon}_{22} \\ 2\bar{\varepsilon}_{12} \end{bmatrix} \tag{3.42}$$

$$\begin{bmatrix} \bar{m}^{11} \\ \bar{m}^{22} \\ \bar{m}^{12} \end{bmatrix} = \frac{t^3}{12} \cdot \mathbf{D} \cdot \begin{bmatrix} \bar{\kappa}_{11} \\ \bar{\kappa}_{22} \\ 2\bar{\kappa}_{12} \end{bmatrix} \tag{3.43}$$

Now, the internal virtual work can be expressed in terms of the normal forces and bending moments as:

$$\delta W_{int} = -\int_{\Omega} (\mathbf{S} : \delta\mathbf{E})\, d\Omega = -\int_{A} (\mathbf{n} : \delta\boldsymbol{\varepsilon} + \mathbf{m} : \delta\boldsymbol{\kappa})\, dA \tag{3.44}$$

where $dA$ denotes the middle surface differential area in the reference state. The partial differential equations for the Kirchhoff-Love shell can be represented as in equation 3.44 in the weak form [1].

# 4. Isogeometric Analysis and The NURBS-Based Kirchhoff-Love Shell Element Formulation

## 4.1. Motivation to Isogeometric Analysis

As mentioned in the Introduction, Hughes et al. [5, 6] introduced the term "isogeometric analysis", which is an improvement of "isoparametric analysis" such that the same mathematical description is used with the analysis model and the geometric model. The isoparametric concept means that the initial geometry and the unknown solution field, such as displacement, are described by the same functions [27]. The accurate treatment of rigid body motion is based on the isoparametric concept. In CAD, techniques like spline-functions or SubD (Subdivision Surfaces) are commonly used during geometric modeling. However, low-order basis functions such as linear Lagrange polynomials are used in FEA. Thus, a conversion between these two representations is necessary if a geometry designed in a CAD environment is to be analyzed with FEA. However, a model conversion process with meshing is needed for the analysis of the geometry, which causes the loss of geometric information. In FEA, approximate geometries are used as a result of the meshing process, and the quality of the mesh has an important effect on the analysis result. However, minor geometric flaws may have a decisive effect on the overall structure, which requires the use of an exact geometry, that is, as observed in the buckling of thin shells [1].

The IGA aims to avoid the meshing process, which has an important time effect on some applications, and to adopt the use of the same geometric description in both the analysis of geometry and the solution field. By this, only one model can be used for both design and analysis [1].

## 4.2. Isogeometric Analysis with NURBS

In IGA, the basis functions can comprise any function applied in CAD, provided that they fulfill the essential criteria of the basis function, such as the partition of unity and the maintenance of linear independence. NURBS are widely used in the CAD world, and the essential criteria of the basis functions mentioned above are also satisfied. The T-Splines [28, 29, 30, 31, 32], and the subdivision surfaces [33, 10] are alternatives of NURBS in isogeometric analysis as basis functions. Isogeometric analysis with subdivision surfaces will be investigated and compared further with the NURBS based isogeometric analysis in this thesis.

Elements are used for computations in IGA, as in FEA, and there exist two ways of defining a NURBS element. The first method is to treat the entire patch as a single NURBS element.

The second method, which was also used in [1] to derive NURBS-based Kirchhoff-Love shell element, is to define NURBS elements based on the knot spans of the knot vectors. This means that the domain is divided into several NURBS patches, and each patch is also further divided into elements by the knot vectors to act as a subdomain [1].

As stated in Chapter 2, a NURBS patch is constructed within a parametric domain, which is segmented into intervals with knot vectors, and these segments are referred to as elements. Since B-Spline basis functions behave as polynomials within each knot interval, it is possible to use Gauss quadrature for integration over the element level. However, the use of Gauss quadrature provides only an approximation solution for the integration of NURBS basis functions because the NURBS basis functions are not pure polynomials but are rational polynomials. Nevertheless, the application of Gauss quadrature to NURBS elements has been explored in the literature [5] and validated, as well as in the benchmark examples in [1].

Similar to FEA, a set of nodes and their corresponding basis functions are required to define a NURBS element, where the nodes are the control points of NURBS, and the degrees of freedoms and boundary conditions are applied to them. The NURBS-based Kirchhoff-Love shell element formulation [1] is based on displacements. Thus, the degrees of freedom of the element formulation are the displacements of control points.

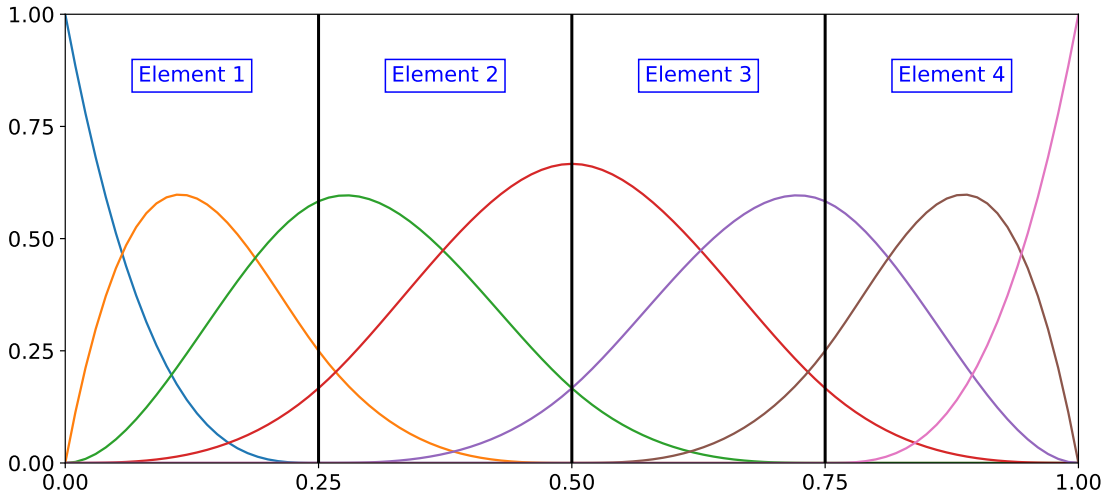


**Figure 4.1** Isogeometric elements and the extension of basis function over them (Adapted from J. Kiendl [1])

According to the definition of the elements provided above, the basis functions are extended to a series of elements as in Figure 4.1, which is a crucial difference between FEM and IGA. For IGA, this difference provides a higher continuity of shape functions within the boundaries of elements. The formulation of the Kirchhoff-Love shell element is based on the high-order continuities between the NURBS elements. However, these elements are not independent from each other due to the interconnection between them. That is why the description of a single NURBS element without a complete NURBS patch is not possible [1]. Nevertheless, the approach for the elements in NURBS-based IGA is the same as the classical FEA for

the implementation purposes, i.e., the stiffness matrix is calculated within an element and assembled into a global stiffness matrix at the end [1].

In the following part, the mesh refinement methods for a NURBS element are explained. For analysis purposes, the knot insertion (h-refinement) and order elevation (p-refinement) methods are the methods for the mesh refinement, which were explained in Section 2.2.3. In the refinement of NURBS, the geometry remains unchanged. This ensures that the geometry is accurately represented, allowing the refinement process to continue without referring back to the original model. This is a crucial distinction from classical FEA [1]. For the NURBS, the knot insertion can be done by arbitrary insertion of the knots, however, which enables local refinement for NURBS curves. However, this may result in an extension of a knot in $\xi$-direction if the inserted knot is in $\eta$-direction and vice versa for the NURBS surfaces. This phenomenon is shown in Figure 4.2. The unrefined parametric space and the corresponding physical model are shown in (a) and (b), respectively. Similarly, the refined parametric space and the corresponding physical model are shown in (c) and (d). The additional knots are extended throughout the entire patch in the specified directions, as illustrated in Figure 4.2. However, this indicates that pure local refinement for NURBS patches is not possible due to the fact that the NURBS surfaces are obtained by a tensor product [1, 34].

For the NURBS, the combination of knot refinement and order elevation is also possible, but it must be done in a proper sequence. If order elevation is applied before knot insertion, this results in an increase on continuity at this location. However, if knot insertion is applied first, then the continuity at this location remains the same since all continuities are preserved for the order elevation. The k-refinement describes the process of applying the order elevation first, and the knot insertion afterwards. This refinement is the desired case if lower continuities are not wanted [1, 6, 34].

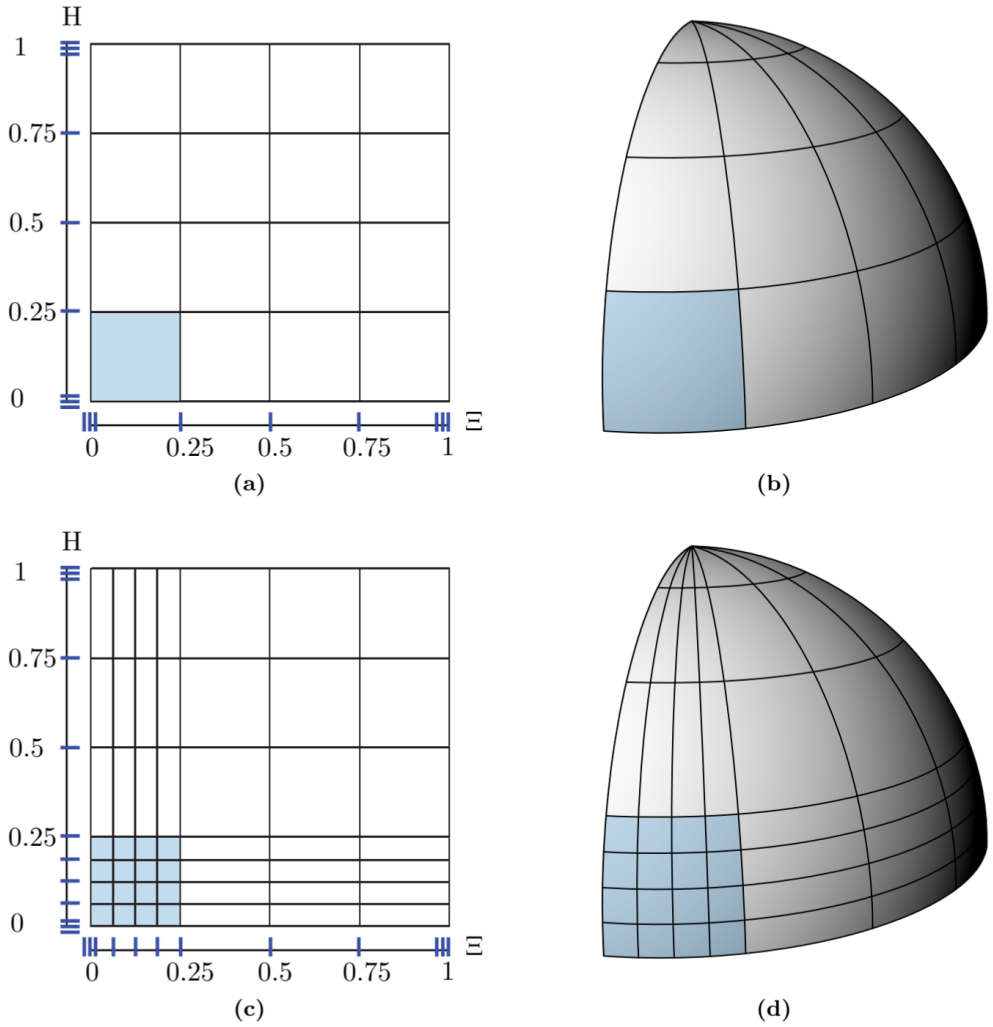


**Figure 4.2** Local refinement of a NURBS element. a) Unrefined parametric space. b) Unrefined physical model. c) Refined parametric space. d) Refined physical model. (Taken from Wüchner et al. [21])

## 4.3. The NURBS-Based Kirchhoff-Love Shell Element Formulation

This section summarized the formulation of NURBS-based Kirchhoff-Love shell element developed by J. Kiendl in [1, 7] to further use to perform isogeometric analysis with both NURBS elements and SubD elements within Kratos Multiphysics environment [16, 17, 18]. The fundamentals of Kirchhoff-Love shell theory and the weak formulation of differential equations were already given in Section 3.2. The detailed derivation of these equations for a discretized system is available in [1], which nodal displacements are the variational variables. It is important to note that the formulation in this section is not exclusive to NURBS but is generally applicable to a displacement-based Kirchhoff-Love shell theory [1]. That is why it can be further used for IGA with SubD.

For any arbitrary variation of the displacement variables $\delta u_r$, the following virtual work equi-

librium condition must be satisfied:

$$\delta W = \frac{\partial W}{\partial u_r}\delta u_r = 0 \tag{4.1}$$

$$\frac{\partial W}{\partial u_r} = 0 \tag{4.2}$$

The Newton-Raphson method is used to solve the nonlinear equation system in equation 4.2 by linearizing it.

$$\frac{\partial W}{\partial u_r} + \frac{\partial^2 W}{\partial u_r \partial u_s}\Delta u_s = 0 \tag{4.3}$$

The sum of internal and external virtual work gives the virtual work, as depicted in equation 3.17. The internal work was also given in 3.18, however, it is rewritten here for the following derivations as:

$$\delta W_{int} = -\int_A (\mathbf{n} : \delta\boldsymbol{\varepsilon} + \mathbf{m} : \delta\boldsymbol{\kappa})\ \mathrm{d}A \tag{4.4}$$

The residual force vector $\mathbf{R}$ is defined as the first derivative of virtual work w.r.t. a displacement variable:

$$R_r = \left(\frac{\partial W^{int}}{\partial u_r} + \frac{\partial W^{ext}}{\partial u_r}\right) = F_r^{int} + F_r^{ext} \tag{4.5}$$

where $\mathbf{F_r^{ext}}$ and $\mathbf{F_r^{int}}$ are the vectors of external nodal loads and internal nodal forces, respectively. The vector of internal nodal forces is given as:

$$F_r^{int} = -\int_A \left(\mathbf{n} : \frac{\partial\boldsymbol{\varepsilon}}{\partial u_r} + \mathbf{m} : \frac{\partial\boldsymbol{\kappa}}{\partial u_r}\right) dA \tag{4.6}$$

The stiffness matrix $\mathbf{K}$ is obtained by taking the second derivative of the virtual work, and it can be split into internal and external virtual works as:

$$K_{rs} = -\left(\frac{\partial^2 W^{int}}{\partial u_r \partial u_s} + \frac{\partial^2 W^{ext}}{\partial u_r \partial u_s}\right) = K_{rs}^{int} + K_{rs}^{ext} \tag{4.7}$$

The derivative of the external loads w.r.t. the displacement variables give the stiffness matrix $\mathbf{K}^{ext}$, and it is taken into account for the displacement-dependent load's case [1, 8]. The derivation of the above term for internal virtual work w.r.t. to the displacement variables gives the internal stiffness matrix $\mathbf{K}^{i}nt$:

$$K_{rs}^{int} = \int_A \left(\frac{\partial\mathbf{n}}{\partial u_s} : \frac{\partial\boldsymbol{\varepsilon}}{\partial u_r} + \mathbf{n} : \frac{\partial^2\boldsymbol{\varepsilon}}{\partial u_r \partial u_s} + \frac{\partial\mathbf{m}}{\partial u_s} : \frac{\partial\boldsymbol{\kappa}}{\partial u_r} + \mathbf{m} : \frac{\partial^2\boldsymbol{\kappa}}{\partial u_r \partial u_s}\right) dA \tag{4.8}$$

In equation 4.8, the membrane stiffness is represented by the first two terms, while the bending stiffness is accounted for by the last two terms. Finally, the equation system depicted below is obtained by inserting the equations 4.5 and 4.8 into equation 4.3 [1]:

$$\mathbf{K}\Delta\mathbf{u} = \mathbf{R} \tag{4.9}$$

The first derivatives of the covariant base vectors w.r.t to the displacement variables $u_r$ for a discretized system having $n$ nodes with the discretized nodal displacement vectors $\hat{\mathbf{u}}_i$, $i = 1, 2, ..., n$, and their corresponding shape functions $N^i$ is given as [7]:

$$\frac{\partial \mathbf{g}_\alpha}{\partial u_r} = \frac{\partial(\mathbf{X}_{,\alpha} + \mathbf{u}_{,\alpha})}{\partial u_r} = \frac{\partial \mathbf{u}_{,\alpha}}{\partial u_r} = \sum_{i=1}^{n} N^i_{,\alpha} \frac{\partial \hat{\mathbf{u}}_i}{\partial u_r}. \tag{4.10}$$

The derivatives of $\boldsymbol{\varepsilon}$, $\boldsymbol{\kappa}$, $\mathbf{n}$ and $\mathbf{m}$ w.r.t. to the displacement variables $u_r$ can be derived from the equation 4.10, and their detailed derivation are available in [1]. Using NURBS as basis functions allows for the calculation of curvatures at any point within the structure without needing additional information. In contrast, rotation-free FEA employing linear shape functions requires evaluating a patch of elements surrounding the element in question to compute curvatures [35, 36]. Thus, the mentioned theory can be directly applied when NURBS are used as shape functions [7].

# 5. Fundamentals of Subdivision Surfaces

## 5.1. Motivation to Subdivision Surfaces

This chapter aims to introduce subdivision surfaces (SubD) and explore how subdivision schemes can be utilized for isogeometric analysis (IGA) as an alternative to using NURBS as basis functions. Modeling complex geometries with arbitrary topologies might have limitations when using NURBS surfaces, as they are derived from the tensor products of two NURBS curves. Sometimes, the CAD models are very complex and made up by joining multiple NURBS patches, which can often be poorly joined during the design phase. One needs to take care while analyzing these models such that any unmatched patches must be handled carefully to ensure that the geometries are watertight. In addition, the adaptive mesh refinement techniques cannot be applied to NURBS because it is not possible to refine them locally [37].

Starting from an initial mesh, which is also referred to as the control mesh of the surface, the subdivision process creates smooth surfaces through a limiting procedure of repeated refinements. In general, the subdivision process has two steps. The first step is refining the mesh, followed by computing the new nodal coordinates. These new nodal coordinates are linear functions of the coarser mesh's nodal coordinates. These computations are local for the relevant schemes, meaning they only involve the nodal coordinates of the coarser mesh within a limited, finite topological area. As a result, highly efficient implementations are obtained [33]. The production of smooth surfaces in the limit depends on a proper subdivision scheme design using appropriate weights. Subdivision techniques that produce limit surfaces with a square-integrable curvature tensor are valuable for both geometric modeling applications and thin-shell analysis [33].

The subdivision schemes in the mathematical geometric modeling literature can be divided into two groups:

- **Interpolating schemes:** In this scheme, the nodal positions of new vertices are computed as we transition from a coarser to a finer mesh, while the nodal positions of the coarser mesh remain fixed. Hence, the limit surface is represented by the nodal positions of the initial mesh, along with any nodes generated during the subdivision process. In literature, the interpolating schemes with quadrilateral surfaces were introduced by Kobbelt et al. [38], and the triangular meshes were also studied by Dyn et al. [39] and Zorin et al. [40]. However, these schemes are not suitable for the thin-shell analysis because curvatures of the limit surfaces do not exist, although they have $C^1$ continuity [33].
- **Approximating schemes:** In these methods, the computation of nodal positions of the new vertices that have been added and the vertices coming from the coarser mesh, which

means they already have nodal positions, are performed. As a result, the nodal positions of the original mesh do not represent samples of the final surface [33]. Loop [41] introduced the approximating scheme using triangular meshes, and the schemes using quadrilateral meshes were introduced by Catmull-Clark [9] and Doo-Sabin [42]. This thesis focuses on the Catmull-Clark subdivision scheme, a technique that produces surfaces with $C^2$ continuity throughout, except at points associated with extraordinary vertices. According to the findings of Peters and Reif [43], these extraordinary vertices exhibit $C^1$ continuity instead. In addition, the surfaces have square-integrable principal curvatures, which enables the use of this scheme for thin-shell analysis [33].

The first application uses the subdivision surfaces for solving the Kirchhoff-Love shell formulation was implemented by Cirak et al. [33]. This study focused on the use of Loop's subdivision scheme, but it was also mentioned that the Catmull-Clark subdivision scheme would be a promising alternative due to the relatively better performance of quadrilateral elements for very coarse meshes in the finite-element computations [33]. In this thesis, the Catmull-Clark subdivision scheme will be adapted for isogeometric Kirchhoff-Love shell analysis within the Kratos Multiphysics framework [16, 17, 18].

In the following sections, the details about the Catmull-Clark subdivision algorithm, as well as Stam's exact evaluation method for subdivision surfaces [44] will be presented.

## 5.2. Catmull-Clark Subdivision Surfaces

Catmull-Clark subdivision surfaces are the limit surfaces obtained through successive subdivisions of a control mesh, which are considered uniform bi-cubic B-splines at the limit. To better understand the Catmull-Clark subdivision algorithm for surfaces, it is helpful to first discuss the Lane-Riesenfeld subdivision algorithm for curves.

### 5.2.1. Lane-Riesenfeld Algorithm

Starting from an initial control polygon, the Lane-Riesenfeld algorithm successively refines a curve after a number of subdivision steps, which results in a B-spline curve. A special case of this subdivision algorithm is presented in Figure 5.1. The control point $\mathbf{P}^i_{2j}$ in the $i^{th}$ refinement level can be obtained by the following expression:

$$\mathbf{P}^i_{2j} = \frac{1}{2}\mathbf{P}^{i-1}_j + \frac{1}{2}\mathbf{P}^{i-1}_{j+1} \tag{5.1}$$

where the $\mathbf{P}^i_{2j}$ is the mid-point of the 'edge points' $\mathbf{P}^{i-1}_j$ and $\mathbf{P}^{i-1}_{j+1}$. The expression for the computation of control point $\mathbf{P}^i_{2j+1}$ is given as:

$$\mathbf{P}^i_{2j+1} = \frac{1}{8}\mathbf{P}^{i-1}_j + \frac{3}{4}\mathbf{P}^{i-1}_{j+1} + \frac{1}{8}\mathbf{P}^{i-1}_{j+2} \tag{5.2}$$

which is also called a 'vertex point'. To compute this point, the connection between the points $\mathbf{P}^{i}_{2j} - \mathbf{P}^{i-1}_{j+1}$ and $\mathbf{P}^{i-1}_{j+1} - \mathbf{P}^{i}_{2j+2}$ is needed. Then, the vertex point $\mathbf{P}^{i}_{2j+1}$ is the mid-point of this connecting line, which is also associated with the upper level of refinement [37].

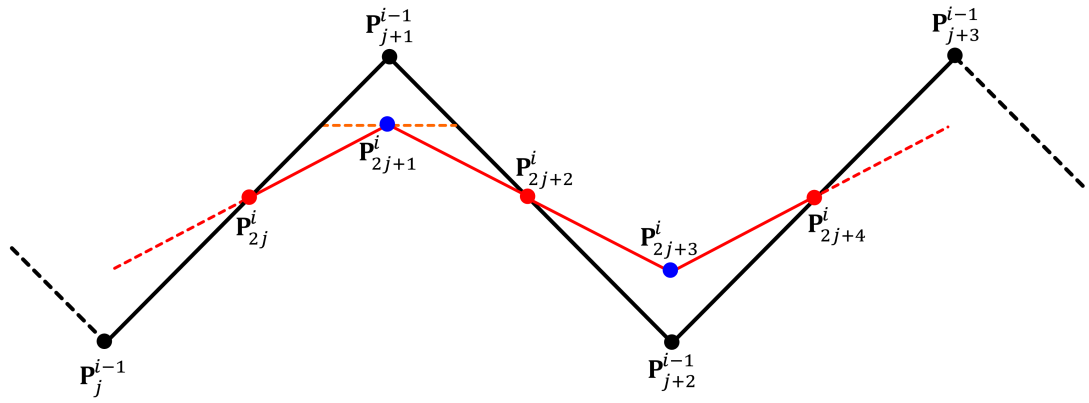


**Figure 5.1** Computation of new control points with the Lane-Riesenfeld algorithm (Adapted from Liu et al. [37])

### 5.2.2. Catmull-Clark Subdivision Algorithm for Surfaces

The algorithm for subdividing surfaces resembles that of curves. Each face in the original control mesh is divided into four new faces. For a closed surface, both the number of faces and control vertices are doubled. The new control points can be classified as 'face points', 'edge points', and 'vertex points', as in the one-dimensional curves. Based on the Figure 5.2, the face points, edge points, and the vertex points in the $i^{th}$ level of refinement are computed with the following expressions:

For face points:

$$\mathbf{P}_{2j,2k} = \frac{1}{4}\left[\mathbf{P}^{i-1}_{j,k} + \mathbf{P}^{i-1}_{j,k+1} + \mathbf{P}^{i-1}_{j+1,k} + \mathbf{P}^{i-1}_{j+1,k+1}\right] \tag{5.3}$$

For edge points case 1:

$$\mathbf{P}_{2j,2k+1} = \frac{1}{16}\left[\mathbf{P}^{i-1}_{j,k} + \mathbf{P}^{i-1}_{j,k+1} + 6\mathbf{P}^{i-1}_{j+1,k} + 6\mathbf{P}^{i-1}_{j+1,k+1} + \mathbf{P}^{i-1}_{j+2,k} + \mathbf{P}^{i-1}_{j+2,k+1}\right] \tag{5.4}$$

For edge points case 2:

$$\mathbf{P}_{2j+1,2k} = \frac{1}{16}\left[\mathbf{P}^{i-1}_{j,k} + 6\mathbf{P}^{i-1}_{j,k+1} + \mathbf{P}^{i-1}_{j,k+2} + \mathbf{P}^{i-1}_{j+1,k} + 6\mathbf{P}^{i-1}_{j+1,k+1} + \mathbf{P}^{i-1}_{j+1,k+2}\right] \tag{5.5}$$

For vertex points:

$$\begin{aligned}\mathbf{P}^{i}_{2j+1,2k+1} = \frac{1}{64}\Big[ & \mathbf{P}^{i-1}_{j,k} + 6\mathbf{P}^{i-1}_{j,k+1} + \mathbf{P}^{i-1}_{j,k+2} + 6\mathbf{P}^{i-1}_{j+1,k} \\ & + 36\mathbf{P}^{i-1}_{j+1,k+1} + 6\mathbf{P}^{i-1}_{j+1,k+2} + \mathbf{P}^{i-1}_{j+2,k} + 6\mathbf{P}^{i-1}_{j+2,k+1} + \mathbf{P}^{i-1}_{j+2,k+2}\Big]\end{aligned} \tag{5.6}$$

In this context, $j$ and $k$ refer to the indices of the control points in the orthogonal direction.

The formulas indicate that the weights associated with the control points for the surfaces are, in fact, the tensor product of the weights specified in the Lane-Riesenfeld algorithms, as given in equations 5.1 and 5.2.

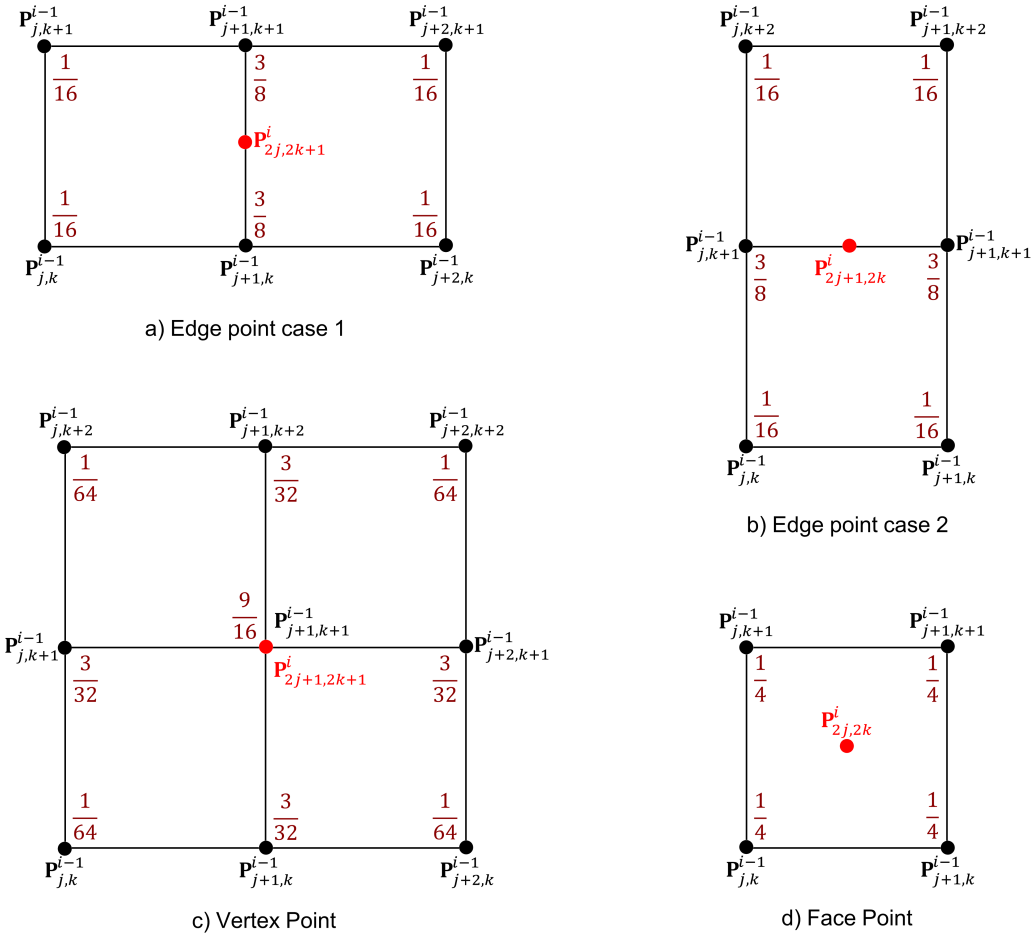


**Figure 5.2** Subdivision masks for different types of control points (Adapted from Liu et al. [37])

Having these formulas, the new control points can be evaluated for the $i^{th}$ level of refinement with the following expression:

$$\boldsymbol{P}^{i} = \boldsymbol{S}\boldsymbol{P}^{i-1} \tag{5.7}$$

where $\boldsymbol{S}$ is the subdivision operator. The subdivision operator $\boldsymbol{S}$ is a matrix consisting of the weights associated with each control point in $\boldsymbol{P}^{i-1}$. Weight distributions for different types of control points are illustrated in Figure 5.2, often referred to as subdivision masks for regular vertices. A vertex with a valence of 4 is termed a "regular vertex," while a vertex with a valence different from 4 is called an "extraordinary vertex." The term "regular valence" specifically indicates a valence of 4, whereas "irregular valence" denotes any valence that is not equal to 4. The weight distributions for an extraordinary vertex are given in Figure 5.3:

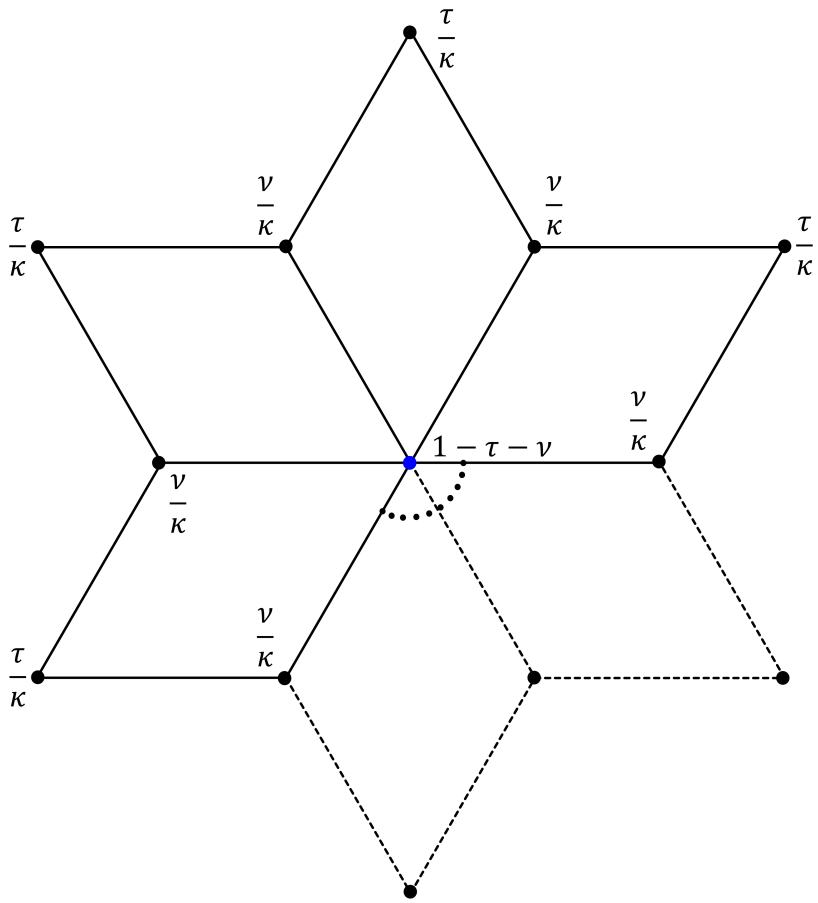


**Figure 5.3** Distribution of weights for an extraordinary point with valence $\kappa$ (Adapted from Liu et al. [37])

where the blue point is the extraordinary vertex, $\kappa$ is the number of valence of this extraordinary vertex, $\nu = 3/(2\kappa)$, and $\tau = 1/4\kappa$. After completing several subdivision steps, a smooth B-spline surface is achieved [37].

### 5.2.3. Interpolation and Evaluation of Curves Using Subdivision Algorithms

This section presents the methods for interpolating and evaluating curves using the Catmull-Clark subdivision algorithm. The curve illustrated in Figure 5.4 is generated using a subdivision algorithm, resembling a cubic B-Spline curve. This means that the interpolation of cubic B-Splines and their corresponding control points results in the limiting curve. To interpolate an element on the curve, four control points are necessary, including the neighboring control points. To evaluate a curve point within element 2, four control points $\boldsymbol{P}_1$, $\boldsymbol{P}_2$, $\boldsymbol{P}_3$, and $\boldsymbol{P}_4$ are necessary, as illustrated in Figure 5.4. This curve point can be evaluated with the following expression:

$$\boldsymbol{x}(\xi) = \sum_{j=1}^{4} N_j(\xi)\boldsymbol{P}_j \tag{5.8}$$

where the parametric coordinates within any element defined as $\xi \in [0, 1]$. The basis function for this element is given as:

$$
\begin{aligned}
N_1(\xi) &= \frac{1}{6}\left[1 - 3\xi + 3\xi^2 - \xi^3\right], \\
N_2(\xi) &= \frac{1}{6}\left[4 - 6\xi^2 + 3\xi^3\right], \\
N_3(\xi) &= \frac{1}{6}\left[1 + 3\xi + 3\xi^2 - 3\xi^3\right], \\
N_4(\xi) &= \frac{1}{6}\xi^3.
\end{aligned}
\tag{5.9}
$$

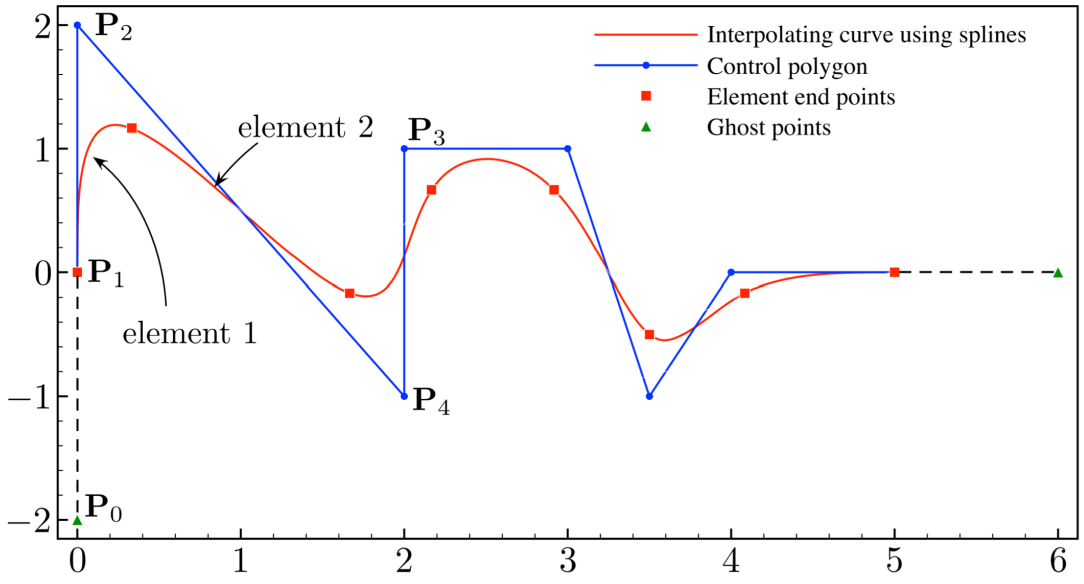


**Figure 5.4** Interpolation of a subdivision curve with B-Splines and corresponding control points (Taken from Liu et al. [37])

To evaluate element 1 in Figure 5.4, the point $\boldsymbol{P}_2$ can be mirrored to $\boldsymbol{P}_0$ as:

$$
\boldsymbol{P}_0 = 2\boldsymbol{P}_1 - \boldsymbol{P}_2 \tag{5.10}
$$

Using a basis function set as illustrated in Figure 5.5b, the curve point for the first element can be evaluated. However, additional ghost points would lead to additional degrees of freedom if the element is adopted for an analysis. Thus, Liu et al. [37] proposed a method to evaluate the curve point on the end elements that the point $\boldsymbol{P}_0$ is substituted into the interpolating equation 5.11 as:

$$
\boldsymbol{x}(\xi) = \sum_{j=0}^{3} N_{j+1}(\xi)\boldsymbol{P}_j = \sum_{k=1}^{3} N_k'(\xi)\boldsymbol{P}_k \tag{5.11}
$$

Having this equation, the curve points within the end elements can be evaluated only using

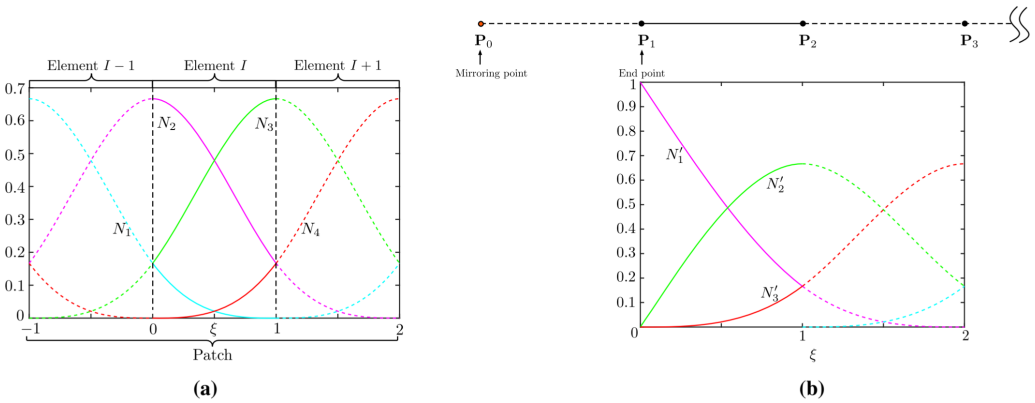


**Figure 5.5** Point mirroring: a) B-Splines for evaluating element $I$ as a Catmull-Clark curve. b) Construction of the mirroring point to evaluate end elements (Taken from Liu et al. [37])

three control points, and their corresponding modified basis functions are defined as:

$$\begin{aligned} N_1'(\xi) &= \frac{1}{6}\left[6 - 6\xi + \xi^3\right], \\ N_2'(\xi) &= \frac{1}{6}\left[6\xi - 2\xi^3\right], \\ N_3'(\xi) &= \frac{1}{6}\xi^3. \end{aligned} \tag{5.12}$$

In the following section, the interpolation and evaluation of Catmull-Clark surfaces having a regular patch are mentioned with these modified basis functions.

### 5.2.4. Interpolation and Evaluation of Catmull-Clark Surfaces with Regular Patch

As mentioned in Section 5.2.2, a regular vertex of a Catmull-Clark surface mesh has a valence of 4. Valence refers to the number of elements connected to a vertex. By incorporating extraordinary vertices within the subdivision surface, it becomes possible to manage arbitrary topologies. Catmull-Clark surfaces offer a straightforward solution for representing surfaces with complex geometries using a single mesh, while NURBS typically require the linking of multiple patches to achieve the same result [37].

Figure 5.6a demonstrates a subdivision surface element (dashed on the figure) with no extraordinary vertices. Evaluation of this element is possible with the formation of an element patch, which contains the element itself and the neighboring elements. In the case of a regular element patch, there are 9 elements with 16 control vertices. Thus, the evaluation of the surface point can be performed with the 16 basis functions corresponding to each control point:

$$\boldsymbol{x}(\boldsymbol{\xi}) = \sum_{j=0}^{15} N_j(\boldsymbol{\xi})\boldsymbol{P}_j \tag{5.13}$$

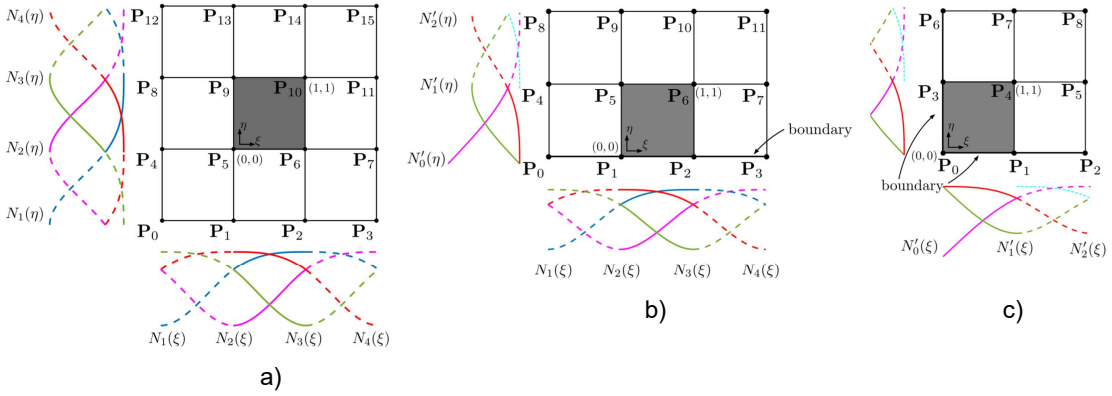


**Figure 5.6** Element patches for evaluation of a Catmull-Clark subdivision element: a) Regular element b) Element having one face at the boundary c) Element having two faces on the boundary (Taken from Liu et al. [37])

where the parametric coordinates of a Catmull-Clark subdivision element are represented as $\boldsymbol{\xi} := (\xi, \eta)$. A Catmull-Clark surface is generated by taking the tensor product of two Catmull-Clark curves, and the basis functions in equation 5.13 obtained by the tensor products of $N(\xi)$ and $N(\eta)$, which defined in the equation 5.9, as:

$$N_i(\boldsymbol{\xi}) = N_{i\%4}(\xi) N_{\lfloor i/4 \rfloor}(\eta), \quad i = 0, 1, \ldots, 15, \tag{5.14}$$

where the operators $\lfloor \bullet \rfloor$ and $\%$ denote the modulus operator and the remainder operator that gives the remainder of the integer division, respectively [37].

An element patch with a shaded subdivision surface element having an edge on the physical boundary is illustrated in Figure 5.6b. This subdivision element has 5 neighboring adjacent elements and 12 corresponding control vertices. Liu et al. [37] proposed a method for direct calculation of the surface element using the shape functions defined in the equation 5.12 on the boundary. The same method can be applied for the case in Figure 5.6c, where the subdivision element within the element patch has two boundary edges. In this case, the number of adjacent neighboring elements is 3, resulting in 9 corresponding control vertices. Additionally, all basis functions are derived from equation 5.12 [37].

### 5.2.5. Interpolation and Evaluation of Catmull-Clark Surfaces with Irregular Patch

Although the existence of extraordinary vertices in Catmull-Clark subdivision surfaces allows the modeling of complex geometries having arbitrary topologies, it also makes the evaluation of surfaces more difficult. In Figure 5.7a, an irregular patch of a Catmull-Clark subdivision element having an extraordinary vertex is illustrated. As can be seen from this figure, the elements are re-numbered as suggested in [44]. The one-level subdivision of this element generates four new sub-elements, along with their corresponding control points. As shown in Figure 5.7, the sub-elements $\Omega_1$, $\Omega_2$, and $\Omega_3$ form a regular patch, while the fourth element

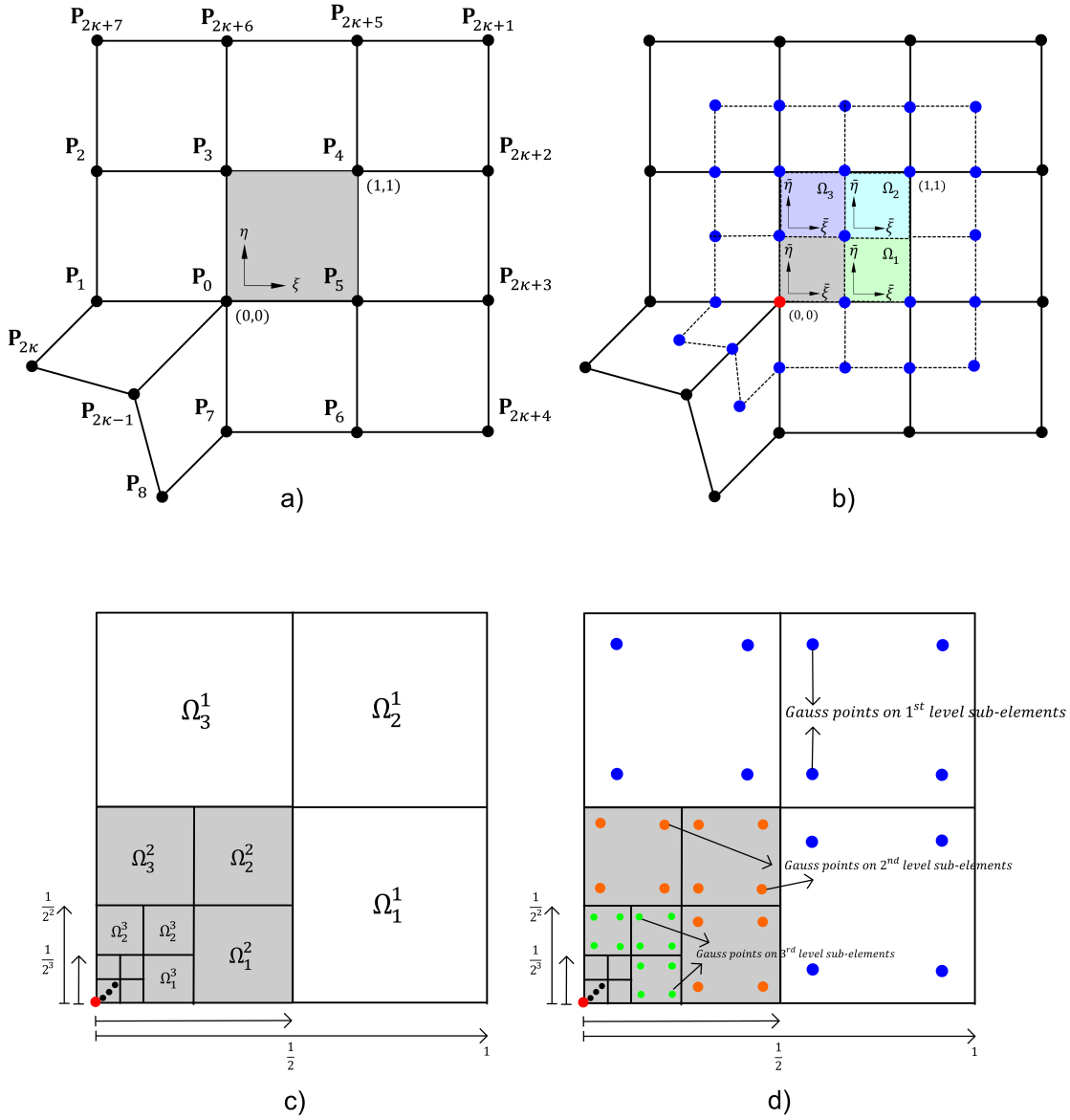


**Figure 5.7** a) An irregular patch from a Catmull-Clark subdivision element that includes an extraordinary vertex. b) One level of subdivision of the element resulting in three regular sub-elements and one irregular sub-element. c) Successive subdivision of the element continues until the evaluation point is located within a regular patch. d) Adaptive Gauss quadrature method the element with an extraordinary vertex. (Adapted from Liu et al. [37])

(in gray) still contains an extraordinary vertex. The iterative subdivision of elements in an irregular patch is necessary until the evaluation point falls within a sub-element that has a regular patch if the point falls into an irregular patch. Hence, the evaluation of the point is possible within the sub-element having a new set of control points $\boldsymbol{P}_{n,k}$ where $n$ is the required number of subdivisions, and the $k = 1, 2, 3$ is the index of sub-element as illustrated in Figure 5.7b. These new set of control points are obtained from:

$$\boldsymbol{P}_{n,k} = \mathbf{D}_k \mathbf{A} \bar{\mathbf{A}}^{n-1} \boldsymbol{P}_0 \tag{5.15}$$

where the $\mathbf{D}_k$ is the selection operator to pick control points for the sub-elements, $\mathbf{A}$ and $\bar{\mathbf{A}}$ are two types of the subdivision operators. This approach was first introduced by J. Stam [44], and the details of the equation are given in the following. The initial set of control points for the irregular patch in Figure 5.7a are given as:

$$\boldsymbol{P}_0 = \left\{\mathbf{P}_0^0, \mathbf{P}_1^0, ..., \mathbf{P}_{2\kappa+6}^0, \mathbf{P}_{2\kappa+7}^0\right\} \tag{5.16}$$

After completing one subdivision step, the new $2\kappa + 17$ control points are shown in Figure 5.7b, represented as blue dots:

$$\boldsymbol{P}_1 = \left\{\mathbf{P}_0^1, \mathbf{P}_1^1, ..., \mathbf{P}_{2\kappa+15}^1, \mathbf{P}_{2\kappa+16}^1\right\} \tag{5.17}$$

where the $\kappa$ is the number of valence. The first subdivision step is expressed with the subdivision operator $\mathbf{A}$ as:

$$\boldsymbol{P}_1 = \mathbf{A}\boldsymbol{P}_0 \tag{5.18}$$

where the subdivision operator $\mathbf{A}$ is defined as:

$$\mathbf{A} = \begin{bmatrix} \mathbf{S} & \mathbf{0} \\ \mathbf{S}_{11} & \mathbf{S}_{12} \\ \mathbf{S}_{21} & \mathbf{S}_{22} \end{bmatrix} \tag{5.19}$$

The terms given in the equation 5.19 are defined by J. Stam in [44], where the term $\mathbf{S}$ is the similar subdivision term in equation 5.7. The first term $\mathbf{S}$ is defined in [44] as:

$$\mathbf{S} = \begin{pmatrix} a_N & b_N & c_N & b_N & c_N & b_N & \cdots & b_N & c_N & b_N & c_N \\ d & d & e & e & 0 & 0 & \cdots & 0 & 0 & e & e \\ f & f & f & f & 0 & 0 & \cdots & 0 & 0 & 0 & 0 \\ d & e & e & d & e & e & \cdots & 0 & 0 & 0 & 0 \\ f & 0 & 0 & f & f & f & \cdots & 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \vdots \\ d & e & 0 & 0 & 0 & 0 & \cdots & e & e & d & e \\ f & f & 0 & 0 & 0 & 0 & \cdots & 0 & 0 & f & f \end{pmatrix} \tag{5.20}$$

where

$$a_N = 1 - \frac{7}{4N} \quad b_N = \frac{3}{2N^2} \quad c_N = \frac{1}{4N^2} \quad d = \frac{3}{8} \quad e = \frac{1}{16} \quad f = \frac{1}{4}$$

The terms $\mathbf{S}_{11}$ and $\mathbf{S}_{12}$ are defined as [44]:

$$\mathbf{S}_{11} = \begin{pmatrix} c & 0 & 0 & b & a & b & 0 & 0 & \mathbf{0} \\ e & 0 & 0 & e & d & d & 0 & 0 & \mathbf{0} \\ b & 0 & 0 & c & b & a & b & c & \mathbf{0} \\ e & 0 & 0 & 0 & 0 & d & d & e & \mathbf{0} \\ e & 0 & 0 & d & d & e & 0 & 0 & \mathbf{0} \\ b & c & b & a & b & c & 0 & 0 & \mathbf{0} \\ e & e & d & d & 0 & 0 & 0 & 0 & \mathbf{0} \end{pmatrix} \quad \mathbf{S}_{12} = \begin{pmatrix} c & b & c & 0 & b & c & 0 \\ 0 & e & e & 0 & 0 & 0 & 0 \\ 0 & c & b & c & 0 & 0 & 0 \\ 0 & 0 & e & e & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & e & e & 0 \\ 0 & 0 & 0 & 0 & c & b & c \\ 0 & 0 & 0 & 0 & 0 & e & e \end{pmatrix} \tag{5.21}$$

where

$$a = \frac{9}{16} \quad b = \frac{3}{32} \quad c = \frac{1}{64} \quad d = \frac{3}{8} \quad e = \frac{1}{16} \quad f = \frac{1}{4}$$

The terms $\mathbf{S}_{21}$ and $\mathbf{S}_{22}$ are defined as [44]:

$$\mathbf{S}_{21} = \begin{pmatrix} 0 & 0 & 0 & 0 & f & 0 & 0 & \mathbf{0} \\ 0 & 0 & 0 & 0 & d & e & 0 & \mathbf{0} \\ 0 & 0 & 0 & 0 & f & f & 0 & \mathbf{0} \\ 0 & 0 & 0 & 0 & e & d & e & \mathbf{0} \\ 0 & 0 & 0 & 0 & 0 & f & f & \mathbf{0} \\ 0 & 0 & 0 & e & d & 0 & 0 & \mathbf{0} \\ 0 & 0 & 0 & f & f & 0 & 0 & \mathbf{0} \\ 0 & 0 & e & d & e & 0 & 0 & \mathbf{0} \\ 0 & 0 & f & f & 0 & 0 & 0 & \mathbf{0} \end{pmatrix}, \quad \mathbf{S}_{22} = \begin{pmatrix} f & f & 0 & 0 & f & 0 & 0 \\ e & d & e & 0 & e & 0 & 0 \\ 0 & f & f & 0 & 0 & 0 & 0 \\ 0 & e & d & e & 0 & 0 & 0 \\ 0 & 0 & f & f & 0 & 0 & 0 \\ e & e & 0 & 0 & d & e & 0 \\ 0 & 0 & 0 & 0 & f & f & 0 \\ 0 & 0 & 0 & 0 & e & d & e \\ 0 & 0 & 0 & 0 & 0 & f & f \end{pmatrix}. \tag{5.22}$$

where

$$d = \frac{3}{8} \quad e = \frac{1}{16} \quad f = \frac{1}{4}$$

To evaluate the sub-elements $\Omega_1$, $\Omega_2$, and $\Omega_3$ in Figure 5.7b, it is necessary to determine $2\kappa + 8$ control points from the new $2\kappa + 17$ control point patch. The required control points from $\boldsymbol{P}_1$ is selected with the help of a selection operator $\mathbf{D}_k$ for the sub-element $\Omega_k$ where $k = 1, 2, 3$ as:

$$\boldsymbol{P}_{1,k} = \mathbf{D}_k \boldsymbol{P}_1 \tag{5.23}$$

The details about the selection operator are available in [44]. Having these relation, the cubic spline basis functions can be used to evaluate a surface point as:

$$\boldsymbol{x}(\boldsymbol{\xi}) = \sum_{j=0}^{15} N_j(\boldsymbol{\xi}) \mathbf{P}_j^{1,k} \tag{5.24}$$

For an irregular element, the number of required subdivisions for an evaluation point in parametric coordinates $\boldsymbol{\xi} = (\xi, \eta)$ is calculated by:

$$n = \lfloor \min\left(-\log_2(\xi), -\log_2(\eta)\right) + 1 \rfloor \tag{5.25}$$

After calculating the required number of subdivisions, the sub-element index $k$ is determined from:

$$k = \begin{cases} 1 & \text{if } \boldsymbol{\xi} \in \left[\frac{1}{2^n}, \frac{1}{2^{n-1}}\right] \times \left[0, \frac{1}{2^n}\right], \\ 2 & \text{if } \boldsymbol{\xi} \in \left[\frac{1}{2^n}, \frac{1}{2^{n-1}}\right] \times \left[\frac{1}{2^n}, \frac{1}{2^{n-1}}\right], \\ 3 & \text{if } \boldsymbol{\xi} \in \left[0, \frac{1}{2^n}\right] \times \left[\frac{1}{2^n}, \frac{1}{2^{n-1}}\right]. \end{cases} \tag{5.26}$$

After the $n^{th}$ refinement, the evaluation point $\boldsymbol{x}$ is positioned in the regular sub-element $k$. By using the selection element $\mathbf{D}_k$, one can obtain the patch for this element as:

$$\boldsymbol{P}_{n,k} = \mathbf{D}_k \boldsymbol{P}_n \tag{5.27}$$

The extended set $\boldsymbol{P}_n$ consists of $2\kappa + 17$ control points, and it is generated as a result of the subdivision of $\boldsymbol{P}^*_{n-1}$ as:

$$\boldsymbol{P_n} = \mathbf{A}\boldsymbol{P}^*_{n-1} \tag{5.28}$$

In the set $\boldsymbol{P}^*_{n-1}$, there are $2\kappa + 8$ control vertices, which are obtained through the refinement of the initial set $\boldsymbol{P}_0$ as:

$$\boldsymbol{P}^*_{n-1} = \bar{\mathbf{A}}^{n-1}\boldsymbol{P}_0 \tag{5.29}$$

where the $\bar{\mathbf{A}}$ denotes a subdivision operator that subdivides the patch to compute the new patch for the irregular element. The subdivision operator $\bar{\mathbf{A}}$ is defined by the terms $\mathbf{S}$, $\mathbf{S}_{11}$, and $\mathbf{S}_{12}$ as:

$$\bar{\mathbf{A}} = \begin{bmatrix} \mathbf{S} & \mathbf{0} \\ \mathbf{S}_{11} & \mathbf{S}_{12} \end{bmatrix} \tag{5.30}$$

After having the set of control points $\boldsymbol{P}_{n,k}$ defined in equation 5.15, which contains 16 control points, one can compute a surface point for an element having an extraordinary vertex as:

$$\boldsymbol{x}(\boldsymbol{\xi}) = \sum_{j=0}^{15} N_j(\bar{\boldsymbol{\xi}})\mathbf{P}_j^{n,k} \tag{5.31}$$

where the parametric coordinates of the evaluated point within the sub-element are denoted by $\bar{\boldsymbol{\xi}}$, and they can be mapped from $\boldsymbol{\xi}$ as:

$$\bar{\boldsymbol{\xi}} = \begin{cases} (2^n\xi - 1, 2^n\eta) & \text{if } k = 1 \\ (2^n\xi - 1, 2^n\eta - 1) & \text{if } k = 2 \\ (2^n\xi, 2^n\eta - 1) & \text{if } k = 3 \end{cases} \tag{5.32}$$

Equation 5.32 can be rewritten as:

$$\boldsymbol{x}(\boldsymbol{\xi}) = \sum_{j=0}^{2\kappa+7} \hat{N}_j(\boldsymbol{\xi})\mathbf{P}_j^0 \tag{5.33}$$

with the Catmull-Clark subdivision surfaces basis function $\hat{N}$. Let $\hat{\mathbf{N}}$ represent a set of $2\kappa + 8$ basis functions within an element that has an extraordinary vertex. Additionally, let $\mathbf{N}$ be defined as a set of 16 regular basis functions, as specified in Equation (7). The set $\hat{\mathbf{N}}$ can be calculated in vector form as follows:

$$\hat{\mathbf{N}}(\boldsymbol{\xi}) = [\mathbf{D}_k\mathbf{A}\bar{\mathbf{A}}^{\mathbf{n-1}}]^T\mathbf{N}(\bar{\boldsymbol{\xi}}) \tag{5.34}$$

The first derivative of the basis functions for Catmull-Clark subdivision surfaces, specifically for elements with extraordinary vertices, is calculated as:

$$\frac{\partial\hat{\mathbf{N}}(\boldsymbol{\xi})}{\partial\boldsymbol{\xi}} = \begin{bmatrix} \frac{\partial\hat{N}_0}{\partial\xi} & \frac{\partial\hat{N}_0}{\partial\eta} \\ \frac{\partial\hat{N}_1}{\partial\xi} & \frac{\partial\hat{N}_1}{\partial\eta} \\ \vdots & \vdots \\ \frac{\partial\hat{N}_{2k+7}}{\partial\xi} & \frac{\partial\hat{N}_{2k+7}}{\partial\eta} \end{bmatrix} \tag{5.35}$$

and it can be computed from the following relation:

$$\frac{\partial\hat{\mathbf{N}}(\boldsymbol{\xi})}{\partial\boldsymbol{\xi}} = [\mathbf{D}_k\mathbf{A}\bar{\mathbf{A}}^{n-1}]^T\frac{\partial\mathbf{N}(\bar{\boldsymbol{\xi}})}{\partial\bar{\boldsymbol{\xi}}}\frac{\partial\bar{\boldsymbol{\xi}}}{\partial\boldsymbol{\xi}} \tag{5.36}$$

where $\frac{\partial\bar{\boldsymbol{\xi}}}{\partial\boldsymbol{\xi}}$ represents a mapping matrix denoted as:

$$\frac{\partial\bar{\boldsymbol{\xi}}}{\partial\boldsymbol{\xi}} = \begin{bmatrix} 2^n & 0 \\ 0 & 2^n \end{bmatrix} \tag{5.37}$$

To calculate the basis functions $\hat{\mathbf{N}}$ at a point $\boldsymbol{x}$ in physical space, two mappings are needed. The first mapping translates physical space to the parametric space of an element with an irregular patch, expressed as $\boldsymbol{x} \rightarrow \boldsymbol{\xi}$. Due to the fact that the irregular patch lacks tensor-product characteristics, $n$ levels of subdivisions are required. Consequently, the point is then

mapped to the parametric domain of a sub-element, represented as $\boldsymbol{\xi} \rightarrow \bar{\boldsymbol{\xi}}$. This mapping is defined in equation 5.32. The number of required subdivisions $n$ approaches infinity when the parametric coordinate $\boldsymbol{\xi}$ becomes $(0, 0)$. Thus, the diagonal terms in the mapping matrix in equation 5.37 become positive infinity when the number of subdivisions also tends to infinity, resulting in not differentiable basis functions $\hat{\mathbf{N}}$ at $\boldsymbol{\xi} = \mathbf{0}$ [37]. This issue is termed the "singular configuration"[45] or "singular parametrization"[46, 47] problem in the literature [37].

## 5.3. Extended Catmull-Clark Subdivision Surfaces

To address the lack of smoothness at extraordinary boundary vertices and the presence of folds near concave corners in the original Catmull-Clark subdivision scheme, Biermann et al. [48] introduced extended Catmull-Clark subdivision schemes. The extended Catmull-Clark subdivision algorithm allows for defining subdivision rules specifically for boundary and crease cases. Before this extended algorithm, some ad hoc solutions were applied, which often led to unexpected results [48]. Properly addressing boundaries and creases is essential in the context of isogeometric analysis using subdivision surfaces. For this reason, the extended Catmull-Clark subdivision algorithm was implemented in this thesis. The following definitions are adapted from Oberbichler et al. [11] because the implementation of the subdivision algorithm was based on this study, which explains the formulation of the algorithm more clearly.

To accurately define the geometry, it is essential to consider the vertices $\mathbf{V}$, as well as the topology formed by the faces $\mathbf{F}$ and edges $\mathbf{E}$. Different tags are assigned to the vertices and edges in the following formulations to represent sharp features. For instance, an edge can be tagged as a crease to model sharp interior and boundary edges or a vertex can be tagged as a crease, dart, or corner. If an edge or vertex is not tagged, it is referred to as untagged or smooth [11]. The effects of various tags on the limit surfaces are shown in Figure 5.8, with tagged edges represented by thick lines and corner vertices depicted using $\odot$. A vertex is assigned a dart tag if connected to only one crease edge. A vertex connected to two or more crease edges can be classified as either a crease vertex or a corner vertex. Conversely, if a vertex is connected only to smooth edges, it is tagged as a smooth vertex [11].

The subdivision rule defined in the following is provided from [9, 48]. This subdivision process involves calculating new points for each new face $\mathbf{F}_i$, vertex $\mathbf{V}_i$, and edge $\mathbf{E}_i$ points. The subdivision point $\mathbf{f}_i$ of the face $\mathbf{F}_i$ is determined from the average of n adjacent vertices $\mathbf{V}_j$ as [11]:

$$\mathbf{f}_i = \frac{1}{n} \sum_{j=1}^{n} \mathbf{v}_j \tag{5.38}$$

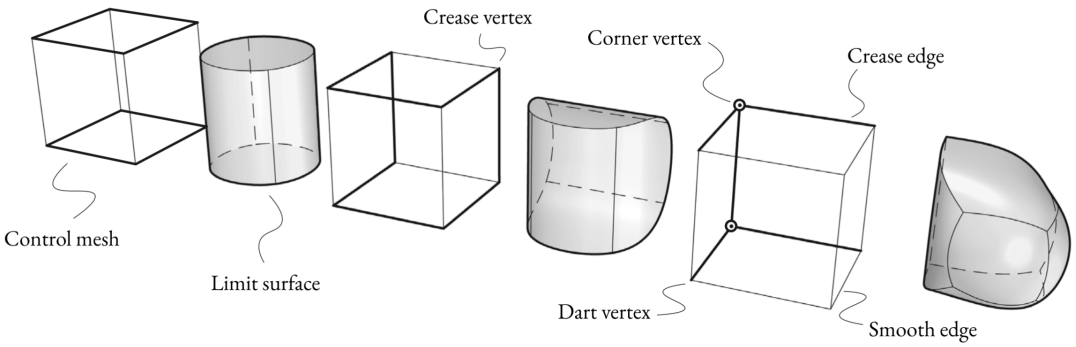


**Figure 5.8** Control meshes having different tags, and their resulting limit surfaces (Taken from Oberbichler et al. [11]).

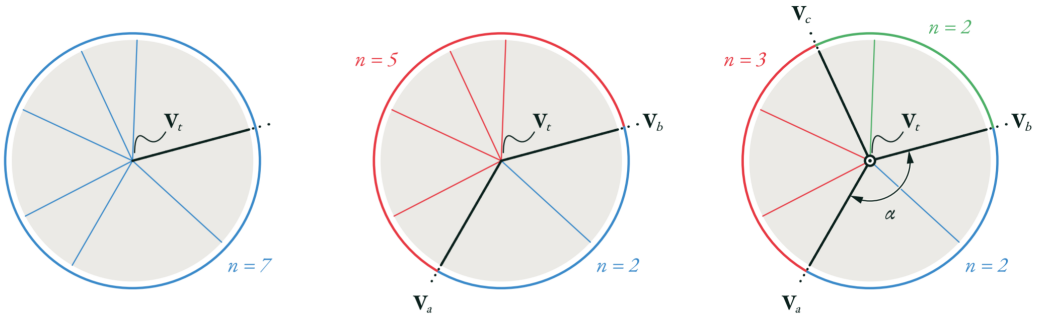


**Figure 5.9** Crease the edges that stretch across the sectors surrounding a marked vertex $V_t$. At a dart vertex (on the left), only one sector exists (red). A crease vertex (in the middle) generates two sectors (blue and red). A corner can be adjacent to any number of sectors (for instance, blue, red, green) (Taken from Oberbichler et al. [11]).

The subdivision point for the new vertex position $\mathbf{v}_i$ is determined according to the tag of the vertex $\mathbf{V}_i$. In the case of a crease tag, the two crease edges are connected to $\mathbf{V}_i$ with vertices $\mathbf{V}_a$ and $\mathbf{V}_b$. If the tag of the vertex $\mathbf{V}_i$ is smooth or dart, one needs to calculate the valence $n$, which is the number of adjacent faces, edges, or vertices in this case. The computation of the new vertex position $\mathbf{v}_i$ is given as follows [11]:

$$\mathbf{v}_i = \begin{cases} \mathbf{V}_i & \text{if } \mathbf{V}_i \text{ is a corner,} \\ \frac{3}{4}\mathbf{V}_i + \frac{1}{8}\mathbf{V}_a + \frac{1}{8}\mathbf{V}_b & \text{if } \mathbf{V}_i \text{ is a crease,} \\ \frac{n-2}{n}\mathbf{V}_i + \frac{1}{n^2}\sum_{j=1}^{n}\mathbf{V}_j + \frac{1}{n^2}\sum_{j=1}^{n}\mathbf{f}_j & \text{if } \mathbf{V}_i \text{ is smooth.} \end{cases} \tag{5.39}$$

To compute the new edge point $\mathbf{e}_i$, it is necessary to calculate the angle $\theta_i$ for each smooth edge $\mathbf{E}_i$ that is connected to a tagged vertex $(\mathbf{V}_t$. If the tag of $\mathbf{V}_t$ is dart, then $n$ represents the number of faces connected to that vertex. In other cases, the edge $\mathbf{E}_i$ is bounded by two crease edges that connect to $\mathbf{V}_i$ at vertices $\mathbf{V}_a$ and $\mathbf{V}_b$ [11]. Figure 5.9 illustrates the area between these two edges, where the number of faces in this region is denoted as $n$, and the

angle $\alpha$ denotes the smaller angle formed by the two radii within the area [11].

$$\theta_i = \begin{cases} \frac{2\pi}{n} & \text{if } \mathbf{V}_t \text{ is a dart,} \\ \frac{\pi}{n} & \text{if } \mathbf{V}_t \text{ is a crease,} \\ \frac{\alpha}{n} & \text{if } \mathbf{V}_t \text{ is a corner,} \end{cases} \tag{5.40}$$

where $\alpha = \angle(\mathbf{V}_a, \mathbf{V}_t, \mathbf{V}_b) = \arccos\left(\frac{\mathbf{V}_a - \mathbf{V}_t}{\|\mathbf{V}_a - \mathbf{V}_t\|_2} \cdot \frac{\mathbf{V}_b - \mathbf{V}_t}{\|\mathbf{V}_b - \mathbf{V}_t\|_2}\right)$

After obtaining the angle $\theta_i$, the subdivision point for the edge $\mathbf{e}_i$ is calculated based on three distinct cases. In these cases, it is important to consider both the tags of the vertices and the edges. If only one vertex of the edge is tagged, that vertex is referred to as $\mathbf{V}_t$, while the other, untagged vertex is denoted as $\mathbf{V}_s$, corresponding to the smooth vertex. In the other scenarios, the order of the vertices does not matter, and they are labeled as $\mathbf{V}_a$ and $\mathbf{V}_b$.

$$\mathbf{e}_i = \begin{cases} \frac{1}{2}\mathbf{V}_a + \frac{1}{2}\mathbf{V}_b & \text{if } \mathbf{E}_i \text{ is a crease,} \\ \frac{1}{4}\mathbf{V}_a + \frac{1}{4}\mathbf{V}_b + \frac{1}{4}\mathbf{f}_a + \frac{1}{4}\mathbf{f}_b & \text{if } \mathbf{V}_a \text{ and } \mathbf{V}_b \text{ are tagged or both are untagged,} \\ \frac{1+\cos\theta}{4}\mathbf{V}_t + \frac{1-\cos\theta}{4}\mathbf{V}_s + \frac{1}{4}\mathbf{f}_a + \frac{1}{4}\mathbf{f}_b & \text{otherwise.} \end{cases} \tag{5.41}$$

The subdivision points serve as the vertices of the updated mesh. For each face $\mathbf{f}_i$, a new smooth vertex is created, which has a valence corresponding to the number of vertices within $\mathbf{F}_i$. The tags for the new vertices $\mathbf{v}_i$ and $\mathbf{e}_i$ are derived from the initial vertices $\mathbf{V}_i$ and edges $\mathbf{E}_i$. For each edge $\mathbf{E}_i = \{\mathbf{V}_a, \mathbf{V}_b\}$, two new edges $\{\mathbf{v}_a, \mathbf{e}_i\}$ and $\{\mathbf{e}_i, \mathbf{v}_b\}$ are formed, preserving the original edge's tag and orientation. Each face $\mathbf{F}_i$ connects the subdivision point $\mathbf{f}_i$ to neighboring edge points $\mathbf{e}_j$, creating new smooth edges $\mathbf{f}_i$ and $\mathbf{e}_j$. These edges divide each face into $n$ smaller faces, and this entire process can be applied recursively. At the refinement level $k$, a vertex $\mathbf{V}_i^{(k)}$ on the control mesh corresponds to a vertex $\mathbf{V}_i^{(k+1)}$ on the refined mesh, ultimately converging to a limit point $\mathbf{V}_i^{(\infty)}$ on the limit surface [11].

# 6. Methodology

In this section, the implementation details of IGA with Catmull-Clark SubD surfaces within the Kratos Multiphysics [16, 17, 18] will be discussed. During the implementation, it was found that the modeling of the boundary conditions significantly influences the results, which will also be addressed in the Results section (Section 7). Specifically, correct implementation of clamped supports is essential for ensuring accurate behavior and achieving better convergence results. Green et al. [49] introduced the second-order accurate formulation of boundary conditions for subdivision-based finite element simulations of thin shells. However, the suggested methods couldn't be applied to this thesis due to some limitations. Instead, the clamped supports are approached differently, yielding better convergence rates and more accurate results despite some limitations. Lastly, the current progress of the solution methods for the geometries with irregular elements will be discussed.

## 6.1. Implementation of Isogeometric Analysis with Catmull-Clark Subdivision Surfaces within Kratos Multiphysics

### 6.1.1. Implementation of the Catmull-Clark Subdivision Algorithm and the Creation of Quadrature Points Geometry

The Catmull-Clark subdivision algorithms were developed using the Python programming language to make the programming procedure easier and the code more understandable. However, this choice has limitations, especially regarding computational performance. Previously, the subdivision algorithms defined in Section 5.2.2 were implemented. Then, Stam's fast evaluation technique [44] was also tested. However, the first issue with these two methods, and the most important one, is that they don't deal with the boundaries of the geometry such that they suffer from the lack of smoothness at the extraordinary boundary vertices. That's why, the extended Catmull-Clark subdivision algorithm [48] was implemented in Python based on the formulation from Oberbichler et al. [11]. The Python script that contains the subdivision algorithm is named $extended\_catmull\_clark.py$. In addition to the subdivision algorithm, this Python script also contains several functionalities, which are:

1. Calculation of the shape functions according to the definition provided by Liu et al. [37]. The shape function relations are given in the equations 5.9, 5.12, and 5.14. The regular element patch cases are defined in Figure 5.6. By using this information, the shape function values and their first and second derivatives can be calculated for the cases defined in Figure 5.6.
2. After the subdivision steps, it is important to set the element patches defined in Figure 5.6 with correct control point indices. That's why an algorithm that assigns the correct control point indices for each regular element patch is also included in the Python script.

3. Calculation of the Gauss-Legendre integration points with corresponding weights in the parametric space $[0, 1] \times [0, 1]$.
4. Local subdivision function that only subdivides the irregular elements and their first ring adjacent elements.
5. Identifying the control point indices where the boundary conditions are applied.

Having the Python script $extended\_catmull\_clark.py$, a main script to perform the isogeometric analysis with subdivision surfaces was developed. This script is based on the $kratos\_main\_iga.py$, which is available in the "$IgaApplication$" of Kratos Multiphysics. This main script defines a class called "$StructuralMechanicsAnalysis$", which is the main script of the "$StructuralMechanicsApplication$" within a class structure. The main script also extended with the integration of subdivision surfaces, and the implemented code works in the following steps:

1. Initializing the analysis stage by defining the modelers, extracting the data from the input files, assigning materials to model parts, creating the properties for the shell elements, including the load properties and output properties, and creating the submodel parts for the implementation of the boundary conditions.
2. Integration of the Catmull-Clark subdivision algorithm. Firstly, the control points of the geometry are obtained from the geometry input file. The control points are written manually in the script instead of taken from the geometry input file for the rectangular plate examples including the inner boundary layer for better clamping support effect. Then, the element relations with control points are stored in a list of lists called $faces$, which contains four control point indices with a given order. Then, the geometry is subdivided by a given subdivision level, and the new control points and element definitions, or faces, are obtained.
3. For the definition of the Scordelis-Lo roof geometry, an optimization algorithm was developed. The subdivision hierarchy cannot accurately model spherical and cylindrical shapes due to the limitations of bicubic B-Spline basis functions [49]. That's why the geometry is approximated very closely to the original geometry with the optimization algorithm. This optimization step is also suggested in the [49]. The subdivision step is completed within the optimization algorithm by a given subdivision level, and the new control points and faces are obtained.
4. For the subdivision function, called "$perform\_subdivision$", the inputs are control points, faces, corner control point indices, corner control point indices that are actually crease control point indices, and a number of subdivision levels. The corner control point indices and corner control point indices that are actually crease control point indices inputs are required for the correct subdivision behavior for the extended Catmull-Clark subdivision algorithm.
5. After performing the subdivision, the nodes are recreated in the model part to ensure the correct assignment of the degree of freedom.
6. The valences of each control point for the subdivided geometry are then calculated,

which is required to obtain the control point sets for each element patch defined in Figure 5.6. According to the case, the control points set contains 9, 12, or 16 control points.

7. Loop over each face, or element, starts. Firstly, the control point set of the element is accessed, and the corresponding control points are added to the sub-model parts. Then, the Gauss-Legendre integration points are calculated in the parametric domain.
8. After obtaining the integration points, loop over each integration point within the element starts. First, the shape functions and their first and second derivatives are obtained for each integration point. Then, quadrature point geometry is created. The quadrature points are required to perform the numerical integration.
9. To create a quadrature point geometry with the bicubic B-Spline shape functions, it is necessary to add the $CreateQuadraturePointsUtility$ in Kratos Multiphysics' core codes so that the isogeometric analysis with subdivision surfaces can be performed within the "$IgaApplication$" without adding new shell element formulation. The aim is to change only the shape functions that suit the subdivision surfaces and to create the quadrature points for integration points within each element of the subdivided geometry.
10. The required inputs to create the quadrature point geometry are:
    - Geometry,
    - Working space dimension,
    - Local space dimension,
    - Integration point,
    - Control point set indices for the element,
    - Shape functions, $N$,
    - Shape functions first derivative w.r.t $\xi$, $\frac{\partial N}{\partial \xi}$,
    - Shape functions first derivative w.r.t $\eta$, $\frac{\partial N}{\partial \eta}$,
    - Shape functions second derivative w.r.t $\xi$, $\frac{\partial^2 N}{\partial \xi^2}$,
    - Shape functions second derivative w.r.t $\xi$, and $\eta$, $\frac{\partial^2 N}{\partial \xi \partial \eta}$,
    - Shape functions second derivative w.r.t $\eta$, $\frac{\partial^2 N}{\partial \eta^2}$
11. After creating the quadrature points, a new element to a sub-model part is added. This sub-model part is within the IGA model part. To create a new element, the required inputs are the element name, which is "Shell3pElement" in our case, an assigned ID of the integration point, the quadrature point geometry, and the shell properties. Then, a new condition is added to another sub-model part within the same IGA model part. To create new conditions, the required inputs are the condition name, which is "LoadCondition" in this specific case, quadrature point geometry, and the load properties.
12. At the end of the first loop, the integration point ID is incremented by one, preparing it for the next iteration. Similarly, the element ID is incremented by one, indicating the creation of a new element in the next iteration. Finally, this process is completed for each face or element in the geometry and each integration point within the corresponding element.

The workflow of the implementation described above is summarized in Algorithm 1 as follows:

**Algorithm 1** Isogeometric Analysis with Subdivision Surfaces

**Step 1: Initialize the analysis stage**
Define modelers and extract data from input files.
Assign materials to model parts and create shell element properties.
Create sub-model parts for boundary conditions.
**Step 2: Integrate the Catmull-Clark subdivision algorithm**
Extract control points from input files or define manually.
Store element relations with control points in a list of faces.
Perform subdivision to generate new control points and faces.
**Step 3: Optimize geometry for Scordelis-Lo roof (if applicable)**
Approximate geometry using an optimization algorithm.
Perform subdivision within the optimization step.
**Step 4: Define the *perform_subdivision* function**
Inputs: control points, faces, corner indices, subdivision level.
Outputs: Subdivided geometry and updated control points.
**Step 5: Recreate nodes and assign degrees of freedom**
Recreate nodes in the model part to ensure correct DoF assignments.
**Step 6: Calculate valences for control points**
Compute valences and obtain control point sets (9, 12, or 16 points) for each element.
**Step 7: Iterate over each face (element)**
**for** each face in the geometry **do**
Access the control point set for the current element.
Add control points to sub-model parts.
Calculate Gauss-Legendre integration points in the parametric domain.
**Step 8: Iterate over each integration point**
**for** each integration point in the element **do**
Compute shape functions and their first/second derivatives.
Create quadrature point geometry for numerical integration.
**end for**
**Step 9: Create quadrature point geometry**
Use *CreateQuadraturePointsUtility* in Kratos core codes.
Inputs: Geometry, working space dimension, local space dimension, integration point, control point set indices, and shape function derivatives (first and second orders).
**Step 10: Add new elements and conditions**
Create and add new element:
Element name: "Shell3pElement".
Inputs: integration point ID, quadrature point geometry, shell properties.
Create and add new condition:
Condition name: "LoadCondition".
Inputs: quadrature point geometry, load properties.
Increment IDs for elements and integration points.
**end for**
**Step 11: Finalize the process for all elements and integration points**
Ensure all elements and integration points are processed.

### 6.1.2. Imposing Boundary Conditions

After defining the quadrature point geometry for numerical integration, the next step is to set the boundary conditions for this geometry. Before discussing how boundary conditions are applied at the Kratos Multiphysics level, it is important to understand the theory behind imposing these boundary conditions.

In plate and shell mechanics, boundary conditions typically restrict either displacement or rotation (or both) along the edge of a thin body. When only the displacement is restricted, the boundary condition is called simply supported. In contrast, when both the displacement and rotation are fixed, the boundary condition is referred to as clamped supports [49]. These two support types are used in our example test cases discussed in Chapter 7.

The subdivision basis functions in thin-shell analysis facilitate the representation of geometry and the mechanics of thin shells, such as deformation and stresses within the structure during the analysis [49].In contrast to conventional finite elements, subdivision basis functions have an impact that reaches beyond their adjacent faces. For instance, a vertex influences not only the local face it belongs to but also nearby connected faces and edges [49]. In conventional finite element, the element boundary is aligned with the boundary of the domain. Due to the overlapping nature of subdivision basis functions, methods based on subdivision require careful handling of boundaries, especially regarding the extension of a basis function's support to multiple neighboring elements. Therefore, a compatible boundary representation is necessary [49].

The boundary conditions can be modeled using two methods: explicit boundary modeling and implied boundary modeling [49]. In the explicit boundary modeling approach, the boundary is represented using ghost faces that surround the control mesh without directly parameterizing the geometry. These ghost faces provide flexibility in shaping the geometry and enhancing the smoothness of the subdivision surface. The vertices of the ghost faces are known as ghost vertices, whose positions indirectly affect the limit surface, even though they do not form part of the geometric model [49]. Although the explicit method provides additional flexibility in boundary modeling and compatibility for irregular meshes where the valence of vertex varies, this method requires manual adjustment of the control mesh, including ghost faces, and it is computationally more complex due to indirect boundary effects [49].

The implied boundary method, which is presented in Cirak et al.[33, 49] does not require the explicitly created ghost points. Instead, every face of the control mesh, including the boundary faces, is parametrized for the geometry [49]. Special subdivision rules define the geometric limits for boundary faces that do not have one complete neighborhood. This means that the boundaries of the limit surface are determined by the same curves that would be produced if one-dimensional subdivision were applied to the boundary [49]. These conditions can be implemented by forming a ring of ghost faces that originate from the positions of the control vertices along the boundary of the mesh, which is shown by Schweitzer and Duchamp [8, 49].

This method is easier to implement since it doesn't require ghost faces, and it automatically maintains smooth transitions at the boundaries. However, the design of the control mesh is limited by the need for regular valence, making the method less flexible when dealing with irregular topologies or complex boundary geometries [49].

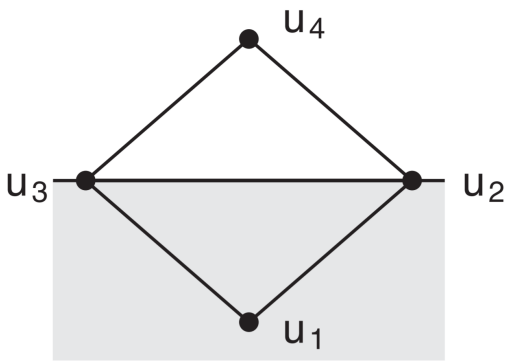


**Figure 6.1** Boundary condition geometry from the implementation of Cirak et al.[33] (Taken from Green et al.[49])

**Table 6.1** Boundary conditions implemented in Cirak et al. [33] (Adapted from Green et al.[49])

| Displacement | Rotation | Boundary Condition |
|---|---|---|
| Free | Free | $\mathbf{u}_4 = \mathbf{u}_2 + \mathbf{u}_3 - \mathbf{u}_1$ |
| Fixed | Free | $\mathbf{u}_4 = -\mathbf{u}_1$, $\mathbf{u}_2 = \mathbf{u}_3 = \mathbf{0}$ |
| Fixed | Fixed | $\mathbf{u}_1 = \mathbf{u_2} = \mathbf{u_3} = \mathbf{u_4} = \mathbf{0}$ |

Figure 6.1 illustrates the boundary condition geometry used by Cirak et al. [33], while Table 6.1 summarizes the boundary conditions based on the types of supports. The ghost faces in this implementation obey the rules defined in Schweitzer and Duchamp [8] such that all the faces along the boundary have regular valence (or number of neighborhoods) [49]. The relationships presented in Table 6.1 ensure that the positions of ghost faces are maintained after deformation while still adhering to the constraints established by Schweitzer and Duchamp [8, 49].

In this thesis, the subdivision basis functions at the boundaries are defined using ghost points, as discussed in Section 5.2.3. In Kratos Multiphysics, the support conditions are defined in "$physics.iga.json$", and "$ProjectParameters.json$" files. To apply a simply support along an edge, $geometry\_type$ is set to "$GeometrySurfaceNodes$", and "$local\_parameters$" defines the location of the applied support. The location of the applied supports is defined in parametric space, with the entire geometry parameterized using coordinates of $[0, 0] \times [1, 1]$. When the support is applied along an edge, the "$local\_parameter$" is represented by an index of $-1$. For example, if support is applied along the edge at $x = 0$, the corresponding entry for "$local\_parameter$" would be $[0, -1]$, where $-1$ indicates that the support is applied through that edge, and $0$ represents $x = 0$. Similarly, a support along $y = 1$ can be applied by inserting "$local\_parameter$" with $[-1, 1]$. Similarly, a clamped support is applied by including "$GeometrySurfaceNodes$", and "$GeometrySurfaceVariationNodes$", where the

latter is used to fix the rotation. "The $local\_parameters$" are applied in the same manner. The clamping support is created by fixing displacements at the boundary edge vertices and their adjacent vertices in the specified direction.

After specifying the boundary conditions in $physics.iga.json$ and $ProjectParameters.json$, the next step is to identify the control point indices for applying the supports based on the information from these two files. In the main Python script, the boundary conditions are strongly imposed and added through the following steps:

1. Obtaining the control point indices by calling $boundary\_condition\_vertices$ function of $extended\_catmull\_clark.py$ Python script.
2. Next, the code iterates over the list of sub-model parts designated for boundary conditions, "$sub\_model\_part\_bc$". For each sub-model part, it creates a new sub-model part within the "$IgaModelPart$" of the model.
3. The code then checks "$sub\_model\_part\_bc\_type$", which gives the type of boundary condition. If the type is specified to "$GeometrySurfaceNodes$", it further checks the parameters in "$sub\_model\_part\_bc\_param$ to determine which vertices to add to the sub-model part. Then, corresponding vertices are added to the sub-model part. These vertices are located at the specified boundary edges of the geometry. Similarly, if the type is specified to "$GeometrySurfaceVariationNodes$", the first adjacent vertices of the boundary vertices along specified edges are added to the sub-model part.
4. After the iteration step, the boundary support conditions are imposed to the geometry.

Applying boundary conditions in Kratos Multiphysics is similar to the method used in the implementation by Cirak et al. [33], which also obeys the constraints defined by Schweitzer and Duchamp [8]. However, this boundary implementation has some drawbacks in terms of convergence of solution, especially for clamped support case, which will be discussed in Section Section 6.2 and Chapter 7. A summary of the implementation details mentioned above is provided in Algorithm 2.

### 6.1.3. Obtaining the Displacements at each Control Point

After implementing the Catmull-Clark subdivision algorithm, creating the geometry for quadrature points, and imposing the boundary conditions, the main Python script is now ready for the solver step. Once the solver is set up, the analysis is performed, and the nodal displacements are obtained. The next step is to recalculate these deflections, taking into account the effects of subdivision basis functions, as they also influence the control points of the first ring element. These calculations are carried out in a separate Python script called $calculate\_deflections.py$. These real deflection values are then used to both result comparison and deflected shape visualizations in Chapter 7. To visualize the geometry, the result data is exported in .vtk format by calling the $ExportDataToVtk$ function, which exports the geometry with the calculated true deflections at each control point.

**Algorithm 2** Imposing Boundary Conditions for Subdivision Surfaces

**Step 1: Identify control point indices for supports**

Retrieve boundary condition information from the project parameters and project physics inputs.

Obtain the control point indices by calling the $boundary\_condition\_vertices$ function from $extended\_catmull\_clark.py$.

**Step 2: Iterate over boundary condition sub-model parts**

Loop through the list of sub-model parts designated for boundary conditions, $sub\_model\_part\_bc$.

For each sub-model part, create a new sub-model part within the $IgaModelPart$ of the model.

**Step 3: Check boundary condition type and parameters**

Check the type of boundary condition using $sub\_model\_part\_bc\_type$:

- If type is $GeometrySurfaceNodes$, check $sub\_model\_part\_bc\_param$ to determine vertices to add.
- Add the corresponding vertices located at the specified boundary edges of the geometry to the sub-model part.
- If type is $GeometrySurfaceVariationNodes$, add the first adjacent vertices of the boundary vertices along the specified edges to the sub-model part.

**Step 4: Impose boundary support conditions**

Apply the boundary support conditions to the geometry after the iteration step.

## 6.2. Issues on the Implementation of the Boundary Conditions

The prescribed boundary condition imposing method from Cirak et al.[33] is straightforward and easy to apply; however, it also brings some drawbacks regarding convergence and accuracy. In the field of solid mechanics, the application of the constraints listed in Table 6.1 is not optimal [49]. Although the constraints applied by Cirak et al.[33] are sufficient, they are too strong such that deformations in higher-order modes are inhibited, which reduces the flexibility and lowers the rate of convergence of the numerical solution. Consequently, not only displacement and slope are restricted, but also curvature and in-plane strains are unintentionally restricted [49]. For example, consider a boundary where several consecutive edges are clamped supported. In this scenario, the displacement along the boundary edge is entirely determined by the subdivision basis function and the constraints imposed. These constraints enforce zero displacement, as well as zero first derivatives (slopes and rotations) and second derivatives (curvature). However, traditional clamped boundary conditions only require zero displacement and slope while allowing finite curvature. Hence, the simulation becomes less flexible and less accurate, especially at boundaries, by eliminating higher-order deformation modes [49]. Another issue of the constraints applied by Cirak et al. [33] is that the bending and membrane deformation modes are unnecessarily coupled. For instance, when both displacement and rotation are constrained to zero, in-plane (membrane) strains are forced to vanish at the boundary [49]. Additionally, this method imposes artificial constraints even on free boundaries, where displacements and rotations are meant to be unconstrained. As a result, the ghost faces are required to follow the subdivision rules proposed by Schweitzer and Duchamp [8]. This enforcement eliminates the useful deformation modes that typically

arise under free boundary conditions, all while maintaining smoothness [49].

In this study, a similar method for imposing boundary conditions is applied as described in Section 6.1.2. According to the results, a slower convergence rate is observed in clamped support cases, which the details are given in Section 7. In the study of Green et al. [49], they proposed an appropriate boundary condition formulation to address these issues by relaxing unnecessary constraints such that higher-order deformation modes (e.g. curvature and membrane strains) vary naturally at the boundaries. The methods proposed by Green et al. [49] are difficult to implement at the Kratos Multiphysics level and were not used in this study. Instead, we developed an alternative approach to achieve a better clamping effect, which will be discussed in detail in the following sections, including a summary of the solution methods proposed by Green et al. [49].

## 6.3. Solution for the Boundary Condition Issues

Green et al. [49] introduced several boundary condition formulation to solve issues of boundary condition constraints from Cirak et al. [33]. Here, the solution methods for vertex position constraint, rotation and clamped constraints will be discussed, which are relevant to our study.

### 6.3.1. Solution to Vertex Point Constraint

The goal of this solution is to ensure that a specific point on the boundary or within the model achieves a target displacement, such as fixed, pinned, or simply supported boundary conditions. In this type of constraint, the displacement of a vertex at the limit surface is expressed as a weighted sum of the displacements of neighboring control mesh vertices as:

$$\mathbf{u}_{limit} = \sum_{i=1}^{k} \beta^i \mathbf{u}_i \tag{6.1}$$

where $\mathbf{u}_i$ represents displacements of neighboring control vertices, $\beta^i$ is the precomputed weights from the limit mask, which depend on the subdivision scheme, and $k$ is the number of the vertex neighborhood. For the Catmull-Clark subdivision scheme, the limit mask is described in Figure 6.2, and the coefficients $\alpha$, $\beta_i$, and $\gamma_i$ are calculated as [49]:

$$\alpha = \frac{n^2}{n(n+5)}, \quad \beta_i = \frac{4}{n(n+5)}, \quad \gamma_i = \frac{1}{n(n+5)}. \tag{6.2}$$

For fixed boundary conditions (e.g. clamped or pinned), the desired displacement $\mathbf{u}_{limit}$ is set to zero. Thus, the constraint in equation 6.2 can be rewritten as a linear constraint as [49]:

$$\mathbf{C}\mathbf{u} = \mathbf{g} \tag{6.3}$$

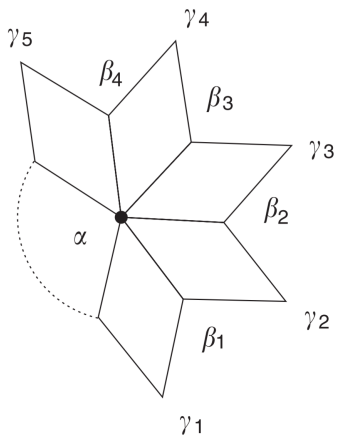


**Figure 6.2** Limit mask for Catmull-Clark Subdivision (Taken from Green et al.[49])

where $\mathbf{C}$ is a matrix containing the weights $\beta_i$ from the limit mask, $\mathbf{u}$ is the displacement vector of all control vertices, and $\mathbf{g}$ is the desired displacement, which is often zero for fixed constraints [49]. This constraint can be implemented by applying the constraint equation for each vertex of interest and evaluating the displacement for interior points at arbitrary locations using Stam's evaluation technique [44, 49].

### 6.3.2. Solution to Rotation and Clamped Constraints

The aim of this solution is to model clamped boundaries, where both displacement and rotation are restricted. During the formulation, second-order accuracy is ensured while avoiding over-constraining the system.

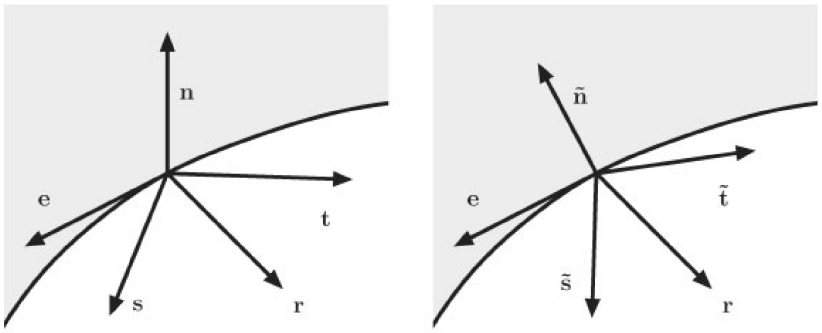


**Figure 6.3** Clamped boundaries. Left: Undeformed. Right: Deformed (Taken from Green et al. [49])

The rotational constraint maintains that the surface normal at any boundary point does not rotate around directions that are perpendicular to the boundary edge. This allows for rotations along the boundary edge while preventing out-of-plane deformations. In Figure 6.3, the undeformed and deformed configuration of a clamped boundary is given. According to this figure, $\mathbf{s}$ and $\mathbf{t}$ span the local tangent plane at any boundary point, $\mathbf{n}$ is the unit normal vector and $\mathbf{e}$ is the tangent direction of the boundary edge within the tangent plane[49]. The vector

$\mathbf{r}$ is defined as $\mathbf{r} = \mathbf{n} \times \mathbf{e}$ to inhibit rotation perpendicular to the boundary edge, and it is orthogonal to both $\mathbf{e}$ and $\mathbf{n}$. The constraint on deformed normal should satisfy the condition $\mathbf{r} \cdot \tilde{\mathbf{n}} = 0$. The implementation details are available in Green et al. [49], and the final linearized constraint formulation to inhibit rotation about a boundary edge is given as:

$$\sum_i \mathbf{u}_i \cdot (w_s^i(\mathbf{t} \times \mathbf{r}) + w_t^i(\mathbf{r} \times \mathbf{s})) = 0 \tag{6.4}$$

where the $w_s^i$ and $w_t^i$ are the weights of two tangent masks regardless of their computation method [49]. This equation provides a scalar constraint for each degree of freedom in the model. A clamped support at the boundary is then obtained by combining the constraints given in the equation 6.4 with the vertex point constraints [49]. A naive approach might attempt to set both the direction and magnitude of the deformed normal vector. However, this results in redundancy, as the displacement constraint already limits the direction of the normal vector when paired with the rotation constraint. Instead, the formulation should only fix the necessary degrees of freedom, ensuring compatibility with solvers and maintaining stability [49].

### 6.3.3. Introducing Inner Boundary Layer for Clamped Support Boundary Condition

Since the implementation of the boundary condition formulations introduced by Green et al. [49] is not straightforward for Kratos Multiphysics environment, another practical approach is developed to obtain better convergence results for the clamped supported boundary condition cases of rectangular plate examples given in Chapter 7. This approach is illustrated in Figure 6.4. The initial rectangular plate example geometry consists of 9 control points and 4 faces. The adjusted geometry contains 25 control points and 16 faces with the introduction of a very thin inner boundary layer. The dimensions $a$ and $b$ are determined by the multiplication of side lengths with a factor:

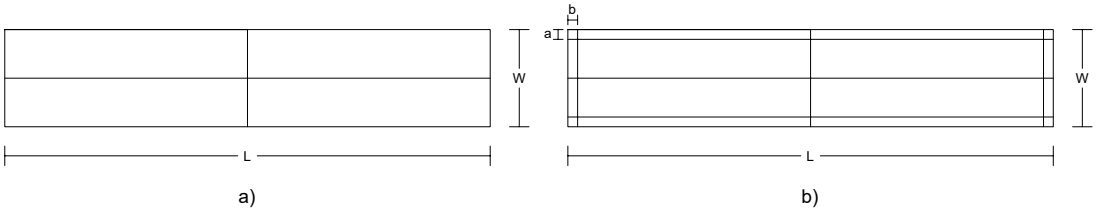


**Figure 6.4** Introduction of inner boundary line for clamped supported rectangular plate examples

$$a = W * factor \quad b = L * factor \tag{6.5}$$

Although this approach gives better convergence results compared to the initial geometry for the clamped supported rectangular plate examples, it is not always practical to implement it. First of all, the new control points and faces must be defined manually in Kratos Multiphysics,

including the dimensions $a$ and $b$. Selecting a factor that is too small causes $a$ and $b$ to become numerically zero, resulting in the loss of clamping effect, and the supports behave as simply supported. There is no defined factor that guarantees the best result; users must determine an appropriate factor through trial and error.

## 6.4. Current Progress on the Geometries with Irregular Elements

This section presents the current achievements related to the solution of geometries with irregular vertices. Due to implementation challenges in Kratos Multiphysics and time constraints associated with this thesis work, the geometries with irregular vertices were not tested.

The subdivision of the geometries having irregular vertices is performed in the same way as the regular geometries, which uses the same Python script "$extended\_catmull\_clark.py$". This Python script also includes a function for local subdivision around the irregular vertices, called "$local\_subdivision$". This local subdivision function works it the following logic:

- Before starting the local subdivision, the original geometry has to be at least two times subdivided as suggested by J. Stam [44]. Also, only the internal faces having exactly one irregular vertex are considered.
- For each irregular internal face after at least two times subdivisions of the initial geometry, the irregular faces and their first ring neighborhood faces are determined.
- Then, subdivision is applied. After subdivision, only the irregular faces and their first and second ring faces are stored to evaluate each new sub-element located within the irregular faces from the previous subdivision level. This means that only the irregular faces and their first-ring neighborhood faces are required for the evaluation with 16 control vertices as shown in Figure 5.6a and Figure 5.7c.
- The local subdivision is carried out iteratively for a specified amount based on the logic outlined in the previous clause.

After the local subdivision step, one can locate the Gauss-Legendre integration points only to the first-ring faces of each local subdivision level to perform numerical integration for the irregular faces of the initial geometry according to the integration scheme from Liu et al. [37], which is illustrated in Figure 5.7c-d. Although this method looks straightforward to implement, adding the new control points after the local subdivision step to the original geometry while maintaining the positions of the initial control vertices is complicated within the Kratos Multiphysics. The reason is one needs to ensure that the stiffness matrix is constructed correctly such that the same control points cannot be added to the stiffness matrix more than once. In addition, the positions of the initial control vertices may slightly change after each subdivision step, which requires correct indexing after each subdivision step such that very small changes in the control point location do not change the index of the control point from the pre-

vious subdivision step. That's why this step is designated for future work. The subdivision of an example geometry having irregular vertex is given in Figure 6.5, and the local subdivision for the two times subdivided geometry is given in Figure 6.6.

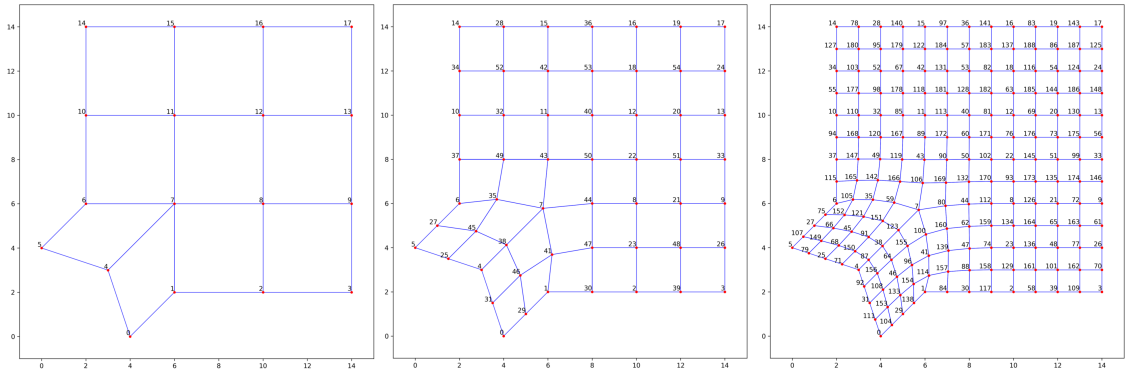

**Figure 6.5** An example geometry having one irregular vertex. Left: Initial geometry. Middle: One level of subdivision. Right: Two levels of subdivision (Adapted from Liu et al. [37])

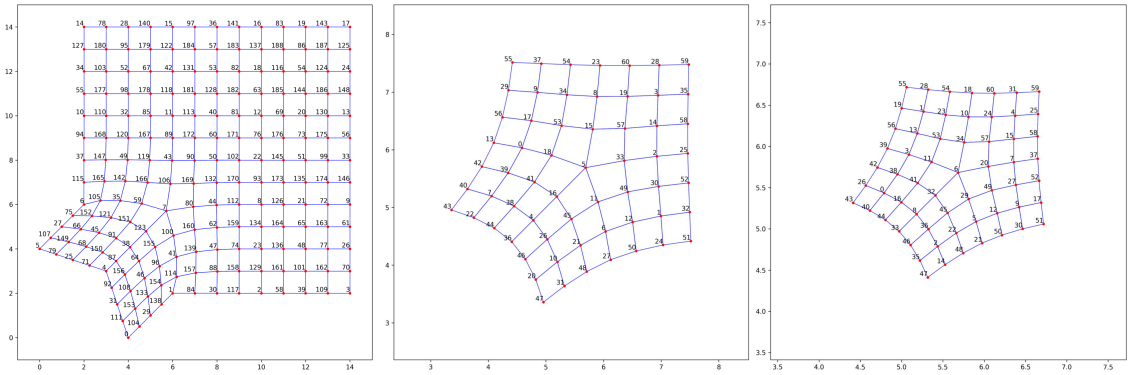

**Figure 6.6** Local subdivision results. Left: Two levels of subdivided geometry. Middle: One level of local subdivision around the first ring elements of the irregular elements. Right: Two levels of local subdivision around the first ring elements of the irregular elements from the previous subdivision level (Adapted from Liu et al. [37])

# 7. Results

This chapter compares isogeometric analysis using Catmull-Clark subdivision surfaces with NURBS-based isogeometric analysis, focusing on several benchmark examples. Specifically, these examples include rectangular thin plates with various support types and the Scordelis-Lo roof, which is one of the three geometrically linear problems from the Shell Obstacle Course [50]. This chapter first discusses the differences between isogeometric analysis (IGA) with Catmull-Clark subdivision surfaces and NURBS-based isogeometric analysis. Further, the results of the benchmark examples are compared.

## 7.1. Differences between NURBS-Based IGA and IGA with Catmull-Clark Subdivision Surfaces

The differences between IGA using Catmull-Clark subdivision surfaces and NURBS-based IGA can be categorized under some headings, such as element definition, basis functions and the existence of extraordinary vertices.

In NURBS-based IGA, the geometry elements are defined as subdomains in a parametric space. Each element is associated with a specific subset of control points influencing its geometry and the basis functions used. The parametric domain is divided into intervals known as knot spans, which are determined by knot vectors. Each of these knot spans corresponds to an element in the physical space. Refinement of the elements is achieved through $h$-refinement by knot insertion, $p$-refinement by order elevation, or $k$-refinement, which the details are given in Section 4.2. In Kratos Multiphysics, this can be achieved by adjusting the parameters within the "$refinements.iga.json$" file. Challenges of the element representation in NURBS-based IGA include the necessity for structured topologies, such as grid-like layouts in parametric space, and difficulties in representing irregular geometries without trimming or additional manipulations.

In IGA with Catmull-Clark subdivision surfaces, the initial control mesh defines the geometry, and elements correspond to the faces of the mesh. After the recursive subdivision of the elements, fines meshes are obtained. As a result of each refinement step, new elements are created from the existing control mesh. For the Catmull-Clark subdivision, the elements are quadrilateral, and their shape and topology are flexible. By using subdivision surfaces, irregular topologies can be handled by including extraordinary vertices. Furthermore, the refinement of the geometry is an iterative process in which elements are refined hierarchically, resulting in a finer mesh with each subdivision step. Challenges of the element representation I IGA with Catmull-Clark subdivision surfaces include reduced smoothness on the basis functions and resulting elements near extraordinary vertices and computational difficulties in

integration over irregular elements near extraordinary vertices.

Locating the integration points is also different in these two methods. In NURBS-based IGA, the integration points are placed through the parametric representation of the whole geometry. In IGA with Catmull-Clark subdivision surfaces, however, the integration points are placed into the parametric domain of each element. For the IGA with Catmull-Clark subdivision surfaces, 2x2 and 3x3 Gauss-Legendre integration points are used, and the results are compared with each other. Barendrecht et al. [51] conclude that the 2x2 Gauss-Legendre scheme guarantees exact integration for bicubic polynomials in the context of Catmull-Clark subdivision surfaces. Green et al. [49] suggested using the 3x3 Gauss-Legendre scheme to reduce numerical errors that may arise from numerical integration.

## 7.2. Rectangular Plate

In this section, a rectangular thin plate is tested for different support conditions: Two side edges clamped supported, four side edges clamped supported, two side edges simply supported, and four side edges simply supported. For the rectangular plate examples, reference analytical solutions for the comparison and error analysis are obtained from the reference [52]. All the examples are subjected to uniform loading in the $z$-direction. Additionally, each geometry initially contains 9 control points for IGA with Catmull-Clark subdivision surfaces. However, for cases with clamped support where an inner boundary layer is introduced, the initial geometry has 25 control points. For the NURBS-based IGA, both the polynomial degrees $p = 2$ and $p = 3$ are tested. For the case of $p = 2$, the initial geometry includes 9 control points. However, for the case of $p = 3$, the initial geometry includes 25 control points.

### 7.2.1. 2 Side Edges Clamped Supported

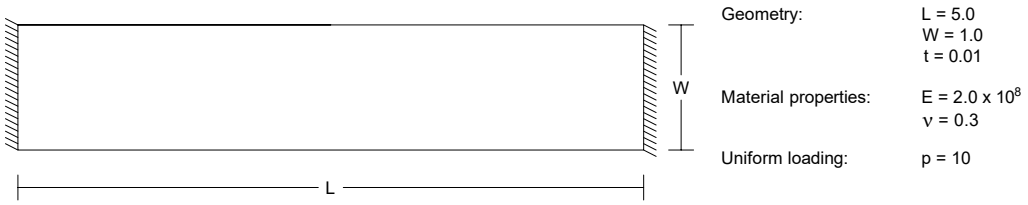


**Figure 7.1** Problem definition: Rectangular plate with 2 side edges are clamped supported

In this section, the results for the rectangular plate example having 2 side boundary edges clamped supported are discussed. The problem definition is given in Figure 7.1. The applied load is a uniform loading distribution in -$z$-direction. The initial control mesh has 4 faces, and 9 control vertices for IGA with Catmull-Clark subdivision surfaces case. However, there are no results for the initial control mesh case because of the insufficient amount of control vertices. In this specific case, all of the control vertices are affected by the support conditions, as discussed in Chapter 6. That's why, at least one level of subdivision for the subdivision

surfaces case is required. Similarly, one level of refinement is also applied to NURBS-based IGA case. For the case of NURBS-based IGA, the results are obtained for the polynomial degrees of both $p = 2$ and $p = 3$ as mentioned before. The reason is that $p = 2$ is not sufficient for this example, and the following examples. The correct degree should be at least $p = 3$. Also, the basis functions of Catmull-Clark subdivision surfaces have a degree of $p = 3$. So, it makes more sense to compare these results. The following results were obtained up to the 5th level of subdivision. Beyond this level, the calculations became excessively time-consuming with no significant improvements in the results. The corresponding control points amount is 4225 for the finest test cases. Furthermore, all these results are obtained for the deflection at the center of the geometry in the -$z$-direction, where the maximum deflection is expected to occur.

In Table 7.1, the deflection results for the NURBS-based IGA with polynomial degrees $p = 2$ and $p = 3$, and IGA with Catmull-Clark subdivision without the introduction of inner boundary layer with 2x2 Gauss-Legendre integration points are given. In addition, the relative errors are also given according to the analytical solution obtained from [52]. The analytical solution is calculated as $9.310E - 01$ accordingly. According to Table 7.1, one can observe that the results for IGA with subdivision surfaces did not converge as in the NURBS-based IGA. The reason comes from the imposing clamped supports at the boundaries, which the details are given in Chapter 6. Due to the strong clamping effect, we still have $3.83\%$ relative error at the 5th subdivision level for IGA with subdivision surfaces. In real life, 5 times subdivision may not be optimal for large geometries, that's why the boundary conditions should be reformulated as suggested by Green et al. [49]. For the case of NURBS-based IGA, results converge to the analytical solution better and faster than IGA with subdivision surfaces. Also, the convergence rate increases with the increase of the polynomial degree, which is expected because the polynomial degree should be at least $p = 3$ in this example. Additionally, Table 7.4 presents the results for IGA with subdivision surfaces without introducing the inner boundary layer utilizing 3x3 Gauss-Legendre integration scheme in this case. Here, it can be observed that using 3x3 Gauss-Legendre scheme does change the results slightly, which can be ignored. This observation proves that the 2x2 Gauss-Legendre scheme is sufficient to ensure exact integration for bi-cubic polynomials in the context of Catmull-Clark subdivision surfaces [51]. The converge plots are given for the 2x2 and 3x3 Gauss-Legendre schemes in Figures 7.2 and 7.4. As can be seen from these figures, the NURBS-based IGA outperforms the IGA with subdivision surfaces.

Next, the effect of the introduction of the inner boundary region is observed for the IGA with subdivision surfaces case to diminish the strong clamping effect as mentioned in Sections 6.2 and 6.3.3 for both 2x2 and 3x3 Gauss-Legendre integration schemes. Tables 7.2 and 7.3 present the deflections and relative errors respectively for both cases with and without the inner boundary layer, across various factors, utilizing the 2x2 Gauss-Legendre integration scheme. For the 3x3 Gauss-Legendre integration scheme, the results are available in Tables 7.5 and 7.6. The relative error vs number of control points plots are given in Figures 7.3 and

7.5 for both 2x2 and 3x3 Gauss-Legendre integration schemes. According to these plots, first of all, the change in the integration scheme does not change the results significantly as expected. However, the introduction of the inner boundary layer significantly increased the convergence rate of the solutions for IGA with subdivision surfaces. From these plots, one can observe that IGA+SubD with inner boundary layer significantly outperforms the case without inner boundary layer. Also, the introduction of inner boundary layer gives better results compared to NURBS-based IGA with polynomial degree $p = 2$ after the number of control points around 289, which can be observed also in the tables. Nevertheless, there is no proper way of selecting the factor for inner boundary layer, and the results differ for the choice of the factor. The strong clamping effect can be observed in Figure 7.6 for the case IGA+SubD without inner boundary layer. As shown in this figure, the clamping effect is too strong at the lower subdivision levels; however, the convergence to the analytical solution is still insufficient for the 5th subdivision level. The effect of introducing the inner boundary layer on the clamping effect is illustrated in Figure 7.7 for the factor of $1/1000$. In Figure 7.8, a closer look is given. According to these two figures, it can be concluded that the strong clamping effect is diminished with the introduction of the inner boundary layer. The plots showing the clamping effects are obtained from Paraview, and one needs to keep in mind that these results do not show the deflections at the limit surface because Paraview uses linear interpolation while plotting the results. Here, the results are obtained at the control points, and linear interpolation is used between the control points. Nevertheless, it gives significant results that demonstrate the effect of strong clamping for visualization.

Lastly, the deflected geometries at the finest subdivision/refinement levels are given in Figures 7.9, 7.10, and 7.11. For a better visualization, a scaling factor of 0.5 has been applied to the displacements. The deflected geometries for IGA+SubD are obtained from Paraview, and the deflected geometry for NURBS-based IGA is obtained from Rhino 8. These deflected geometries show the same deflection pattern, which shows the obtained results are comparable and our implementation of IGA+SubD works fine. Again one needs to keep in mind that Paraview linearly interpolates between deflections at each control point while plotting the result.

**Table 7.1** Rectangular Plate with 2 side edges supported by clamped supports and 2x2 Gauss-Legendre quadrature points for IGA + SubD: No introduced inner boundary layer

| Subdivision Amount | Number of Control Points without Inner Boundary Layer | Deflection Classical IGA $p = 2$ | Deflection Classical IGA $p = 3$ | Deflection IGA+Subdivision without Inner Boundary Layer | Error Classical IGA $p = 2$ | Error Classical IGA $p = 3$ | Error IGA + Subdivision without Inner Boundary Layer |
|---|---|---|---|---|---|---|---|
| 0 | 9 | | | | | | |
| 1 | 25 | 5.363E-01 | 9.083E-01 | 3.384E-01 | 42.40% | 2.44% | 63.66% |
| 2 | 81 | 8.424E-01 | 9.215E-01 | 6.335E-01 | 9.52% | 1.02% | 31.95% |
| 3 | 289 | 9.088E-01 | 9.285E-01 | 7.842E-01 | 2.39% | 0.27% | 15.77% |
| 4 | 1089 | 9.252E-01 | 9.304E-01 | 8.587E-01 | 0.63% | 0.06% | 7.76% |
| 5 | 4225 | 9.295E-01 | 9.309E-01 | 8.954E-01 | 0.17% | 0.01% | 3.83% |

**Table 7.2** Rectangular Plate with 2 side edges supported by clamped supports and 2x2 Gauss-Legendre quadrature points for IGA + SubD: Deflections for introducing inner boundary layer

| Subdivision Amount | Number of Control Points without Inner Boundary Layer | Number of Control Points with Inner Boundary Layer | Deflection IGA+Subdivision without Inner Boundary Layer | Deflection IGA+Subdivision with Inner Boundary Layer Factor: $1/100$ | Deflection IGA+Subdivision with Inner Boundary Layer Factor: $1/250$ | Deflection IGA+Subdivision with Inner Boundary Layer Factor: $1/500$ | Deflection IGA+Subdivision with Inner Boundary Layer Factor: $1/1000$ | Deflection IGA+Subdivision with Inner Boundary Layer $a = b = 0.01$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 9 | 25 | | 7.833E-02 | 9.757E-02 | 1.939E-01 | 4.965E-01 | 1.939E-01 |
| 1 | 25 | 81 | 3.384E-01 | 7.930E-01 | 7.789E-01 | 7.781E-01 | 7.810E-01 | 7.783E-01 |
| 2 | 81 | 289 | 6.335E-01 | 9.250E-01 | 9.138E-01 | 9.108E-01 | 9.103E-01 | 9.110E-01 |
| 3 | 289 | 1089 | 7.842E-01 | 9.294E-01 | 9.304E-01 | 9.295E-01 | 9.287E-01 | 9.295E-01 |
| 4 | 1089 | 4225 | 8.587E-01 | 9.298E-01 | 9.306E-01 | 9.307E-01 | 9.317E-01 | 9.308E-01 |
| 5 | 4225 | | 8.954E-01 | | | | | |

**Table 7.3** Rectangular Plate with 2 side edges supported by clamped supports and 2x2 Gauss-Legendre quadrature points for IGA + SubD: Errors for introducing inner boundary layer

| Subdivision Amount | Number of Control Points without Inner Boundary Layer | Number of Control Points with Inner Boundary Layer | Error IGA+Subdivision without Inner Boundary Layer | Error IGA+Subdivision with Inner Boundary Layer Factor: $1/100$ | Error IGA+Subdivision with Inner Boundary Layer Factor: $1/250$ | Error IGA+Subdivision with Inner Boundary Layer Factor: $1/500$ | Error IGA+Subdivision with Inner Boundary Layer Factor: $1/1000$ | Error IGA+Subdivision with Inner Boundary Layer $a = b = 0.01$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 9 | 25 | | 91.59% | 89.52% | 79.17% | 46.67% | 79.17% |
| 1 | 25 | 81 | 63.66% | 14.82% | 16.34% | 16.43% | 16.11% | 16.40% |
| 2 | 81 | 289 | 31.95% | 0.65% | 1.85% | 2.17% | 2.22% | 2.15% |
| 3 | 289 | 1089 | 15.77% | 0.17% | 0.07% | 0.16% | 0.25% | 0.16% |
| 4 | 1089 | 4225 | 7.76% | 0.13% | 0.04% | 0.03% | 0.07% | 0.02% |
| 5 | 4225 | | 3.83% | | | | | |

**Table 7.4** Rectangular Plate with 2 side edges supported by clamped supports and 3x3 Gauss-Legendre quadrature points for IGA + SubD: No introduced inner boundary layer

| Subdivision Amount | Number of Control Points without Inner Boundary Layer | Deflection Classical IGA $p = 2$ | Deflection Classical IGA $p = 3$ | Deflection IGA+Subdivision without Inner Boundary Layer | Error Classical IGA $p = 2$ | Error Classical IGA $p = 3$ | Error IGA + Subdivision without Inner Boundary Layer |
|---|---|---|---|---|---|---|---|
| 0 | 9 | | | | | | |
| 1 | 25 | 5.363E-01 | 9.083E-01 | 3.380E-01 | 42.40% | 2.44% | 63.69% |
| 2 | 81 | 8.424E-01 | 9.215E-01 | 6.336E-01 | 9.52% | 1.02% | 31.95% |
| 3 | 289 | 9.088E-01 | 9.285E-01 | 7.842E-01 | 2.39% | 0.27% | 15.76% |
| 4 | 1089 | 9.252E-01 | 9.304E-01 | 8.587E-01 | 0.63% | 0.06% | 7.76% |
| 5 | 4225 | 9.295E-01 | 9.309E-01 | 8.953E-01 | 0.17% | 0.01% | 3.83% |

**Table 7.5** Rectangular Plate with 2 side edges supported by clamped supports and 3x3 Gauss-Legendre quadrature points for IGA + SubD: Deflections for introducing inner boundary layer

| Subdivision Amount | Number of Control Points without Inner Boundary Layer | Number of Control Points with Inner Boundary Layer | Deflection IGA+Subdivision without Inner Boundary Layer | Deflection IGA+Subdivision with Inner Boundary Layer Factor: $1/100$ | Deflection IGA+Subdivision with Inner Boundary Layer Factor: $1/250$ | Deflection IGA+Subdivision with Inner Boundary Layer Factor: $1/500$ | Deflection IGA+Subdivision with Inner Boundary Layer Factor: $1/1000$ | Deflection IGA+Subdivision with Inner Boundary Layer $a = b = 0.01$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 9 | 25 | | 4.916E-02 | 1.528E-02 | 1.212E-02 | 1.673E-02 | 1.212E-02 |
| 1 | 25 | 81 | 3.380E-01 | 8.315E-01 | 7.986E-01 | 7.947E-01 | 7.945E-01 | 7.951E-01 |
| 2 | 81 | 289 | 6.336E-01 | 9.234E-01 | 9.201E-01 | 9.158E-01 | 9.138E-01 | 9.160E-01 |
| 3 | 289 | 1089 | 7.842E-01 | 9.290E-01 | 9.298E-01 | 9.297E-01 | 9.293E-01 | 9.297E-01 |
| 4 | 1089 | 4225 | 8.587E-01 | 9.298E-01 | 9.306E-01 | 9.307E-01 | 9.314E-01 | 9.308E-01 |
| 5 | 4225 | | 8.953E-01 | | | | | |

**Table 7.6** Rectangular Plate with 2 side edges supported by clamped supports and 3x3 Gauss-Legendre quadrature points for IGA + SubD: Errors for introducing inner boundary layer

| Subdivision Amount | Number of Control Points without Inner Boundary Layer | Number of Control Points with Inner Boundary Layer | Error IGA+Subdivision without Inner Boundary Layer | Error IGA+Subdivision with Inner Boundary Layer Factor: $1/100$ | Error IGA+Subdivision with Inner Boundary Layer Factor: $1/250$ | Error IGA+Subdivision with Inner Boundary Layer Factor: $1/500$ | Error IGA+Subdivision with Inner Boundary Layer Factor: $1/1000$ | Error IGA+Subdivision with Inner Boundary Layer $a = b = 0.01$ |
|---|---|---|---|---|---|---|---|---|
| 0 | 9 | 25 | | 94.72% | 98.36% | 98.70% | 98.20% | 98.70% |
| 1 | 25 | 81 | 63.69% | 10.69% | 14.22% | 14.64% | 14.66% | 14.60% |
| 2 | 81 | 289 | 31.95% | 0.82% | 1.17% | 1.63% | 1.85% | 1.61% |
| 3 | 289 | 1089 | 15.76% | 0.21% | 0.13% | 0.14% | 0.18% | 0.14% |
| 4 | 1089 | 4225 | 7.76% | 0.13% | 0.05% | 0.03% | 0.04% | 0.03% |
| 5 | 4225 | | 3.83% | | | | | |

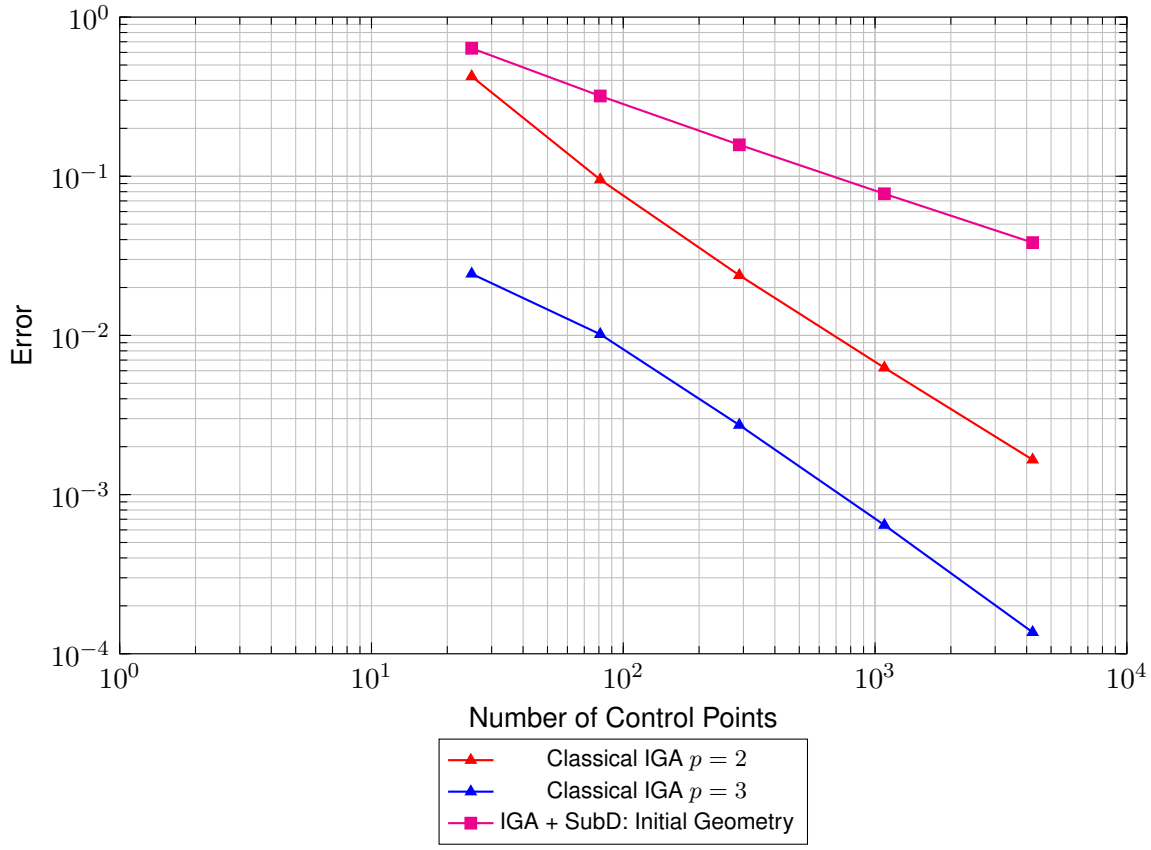


**Figure 7.2** Convergence plot: Rectangular plate with 2 side edges supported by clamped supports and 2x2 Gauss-Legendre quadrature points for IGA + SubD (w/o introduction of internal boundary layer)

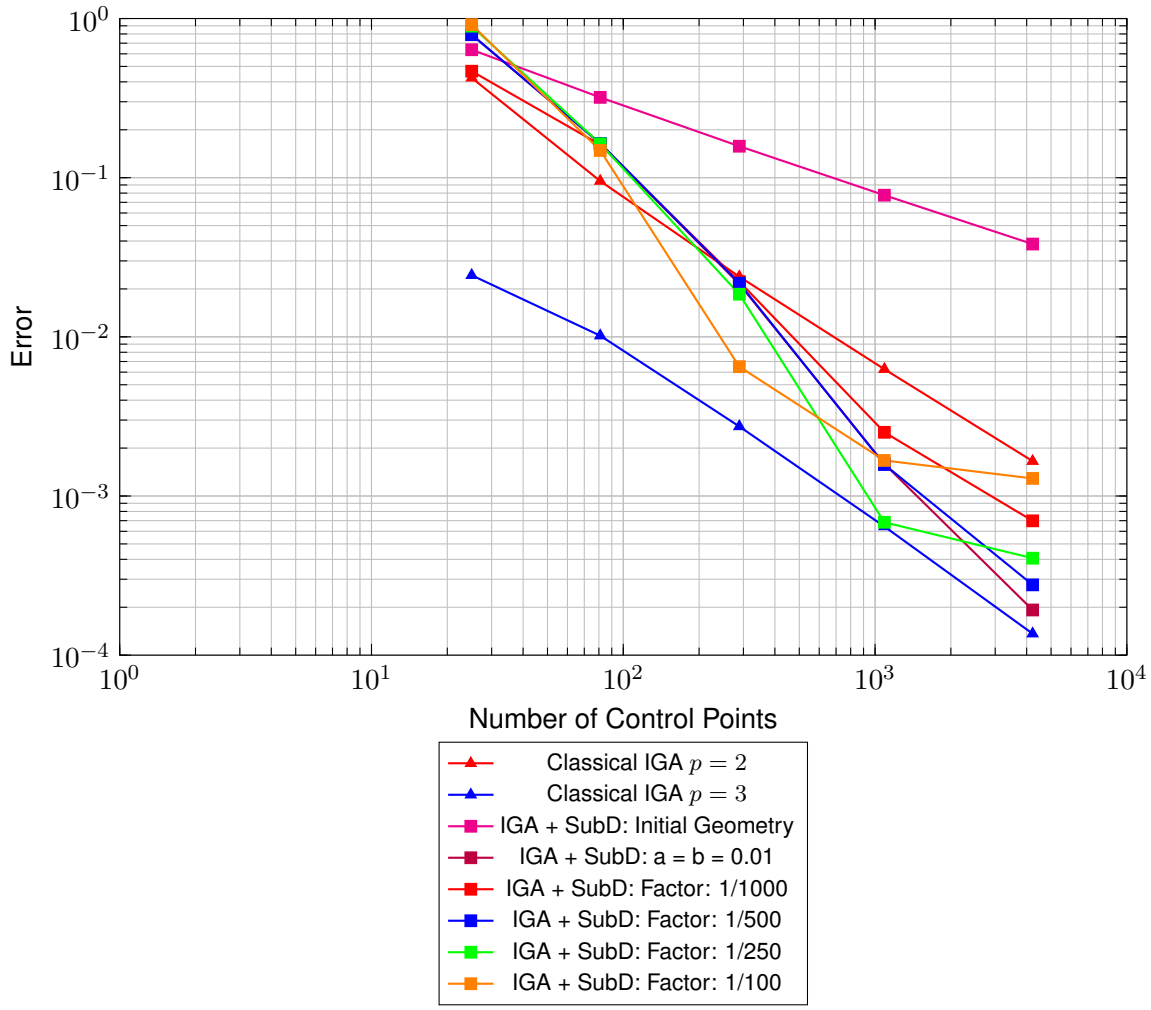


**Figure 7.3** Convergence plot: Rectangular plate with 2 side edges supported by clamped supports and 2x2 Gauss-Legendre quadrature points for IGA + SubD - Errors for different factors and initial geometry in IGA + SubD

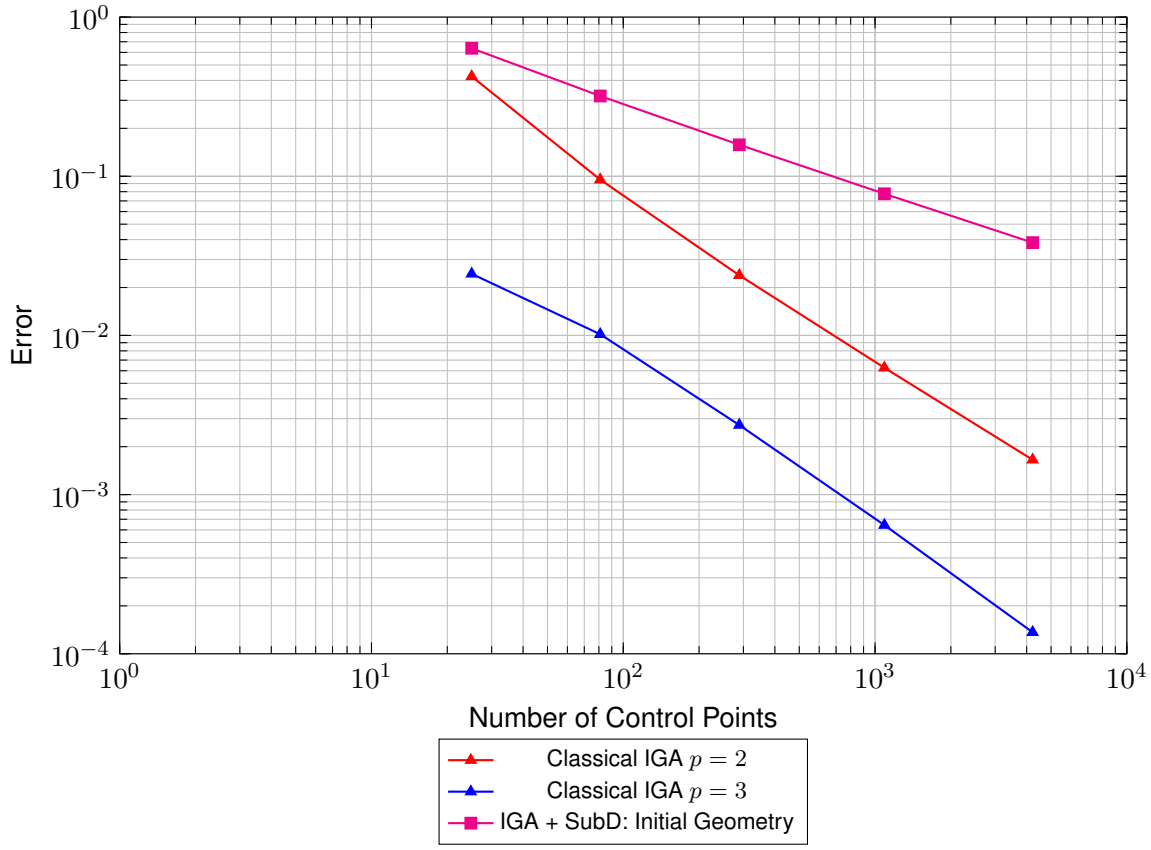


**Figure 7.4** Convergence plot: Rectangular plate with 2 side edges supported by clamped supports and 3x3 Gauss-Legendre quadrature points for IGA + SubD (w/o introduction of internal boundary layer)

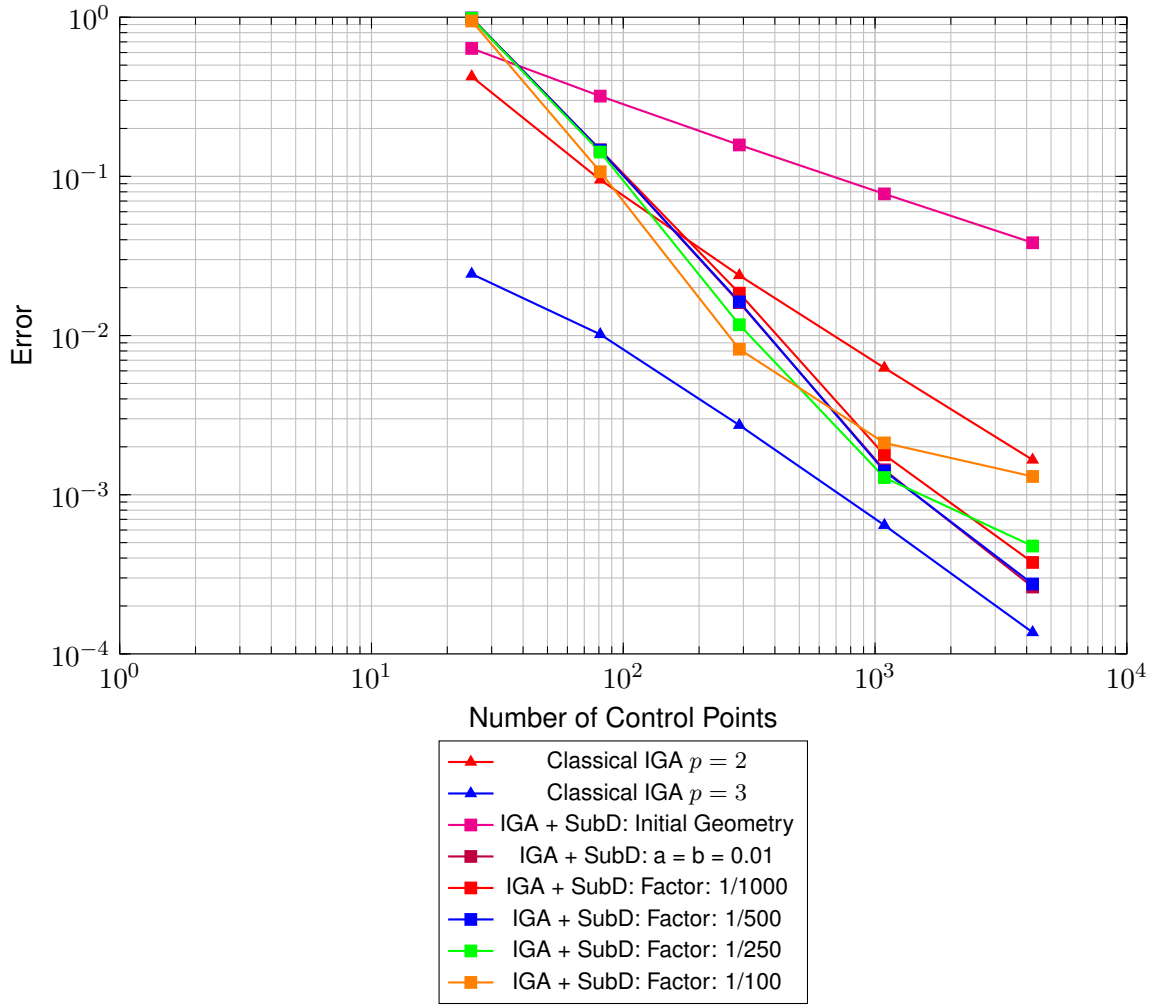


**Figure 7.5** Convergence plot: Rectangular plate with 2 side edges supported by clamped supports and 3x3 Gauss-Legendre quadrature points for IGA + SubD - Errors for different factors and initial geometry in IGA + SubD

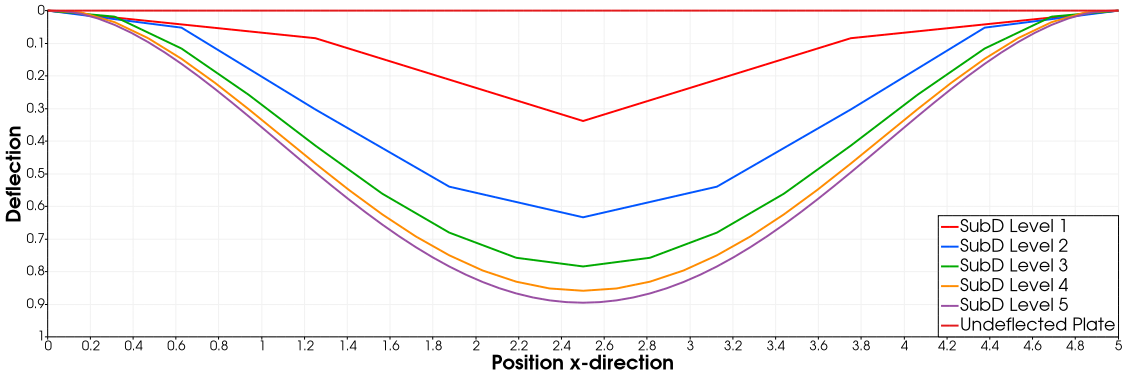


**Figure 7.6** Clamping effect at the boundaries: Rectangular plate with 2 side edges are supported by clamped supports. Results for IGA+SubD without introducing inner boundary layer and 2 Gauss-Legendre points per side. Section from $y = 0.5$

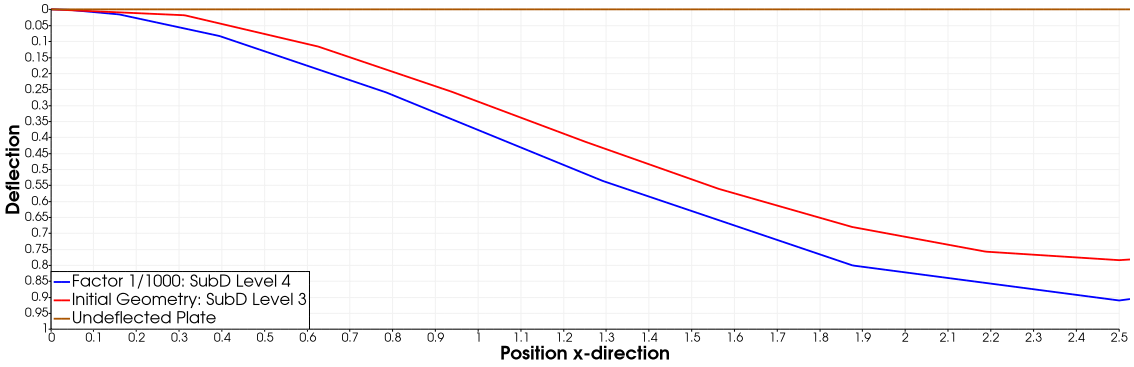


**Figure 7.7** Clamping effect at the boundaries: Rectangular plate with 2 side edges are supported by clamped supports. Results for IGA+SubD without introducing inner boundary layer and IGA+SubD with inner boundary layer with Factor = $1/1000$ and 2 Gauss-Legendre points per side. Section from $y = 0.5$

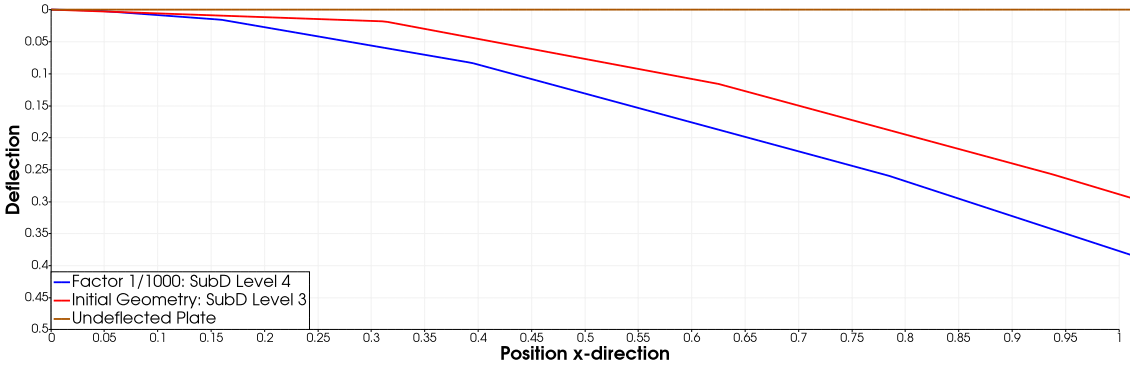


**Figure 7.8** Closer look to clamping effect at the boundary: Rectangular plate with 2 side edges are supported by clamped supports. Results for IGA+SubD without introducing inner boundary layer and IGA+SubD with inner boundary layer with Factor = $1/1000$ and 2 Gauss-Legendre points per side. Section from $y = 0.5$

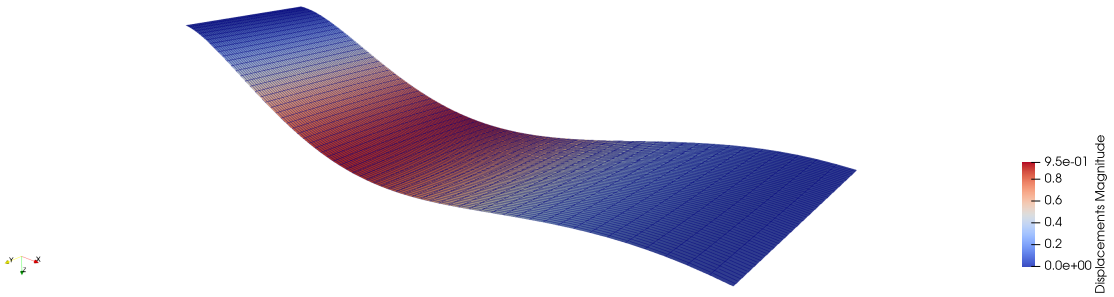


**Figure 7.9** Deflected rectangular plate with 2 side edges are supported by clamped supports. Result for IGA + SubD without introducing inner boundary layer and with 2 Gauss-Legendre points per side.

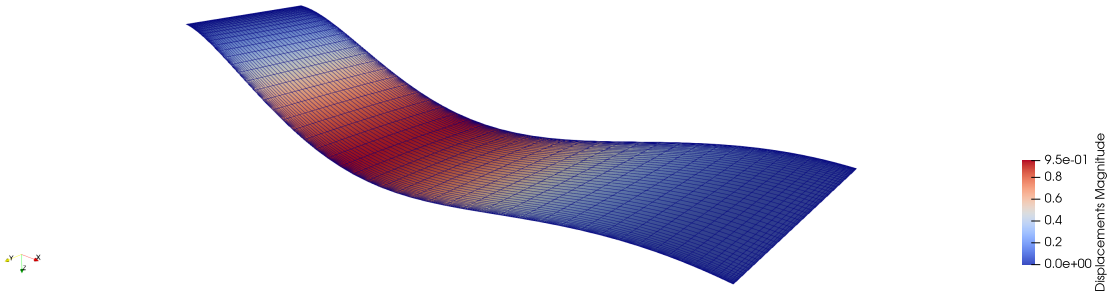


**Figure 7.10** Deflected rectangular plate with 2 side edges are supported by clamped supports. Result for IGA + SubD with introducing inner boundary layer by a factor $1/1000$ and with 2 Gauss-Legendre points per side.

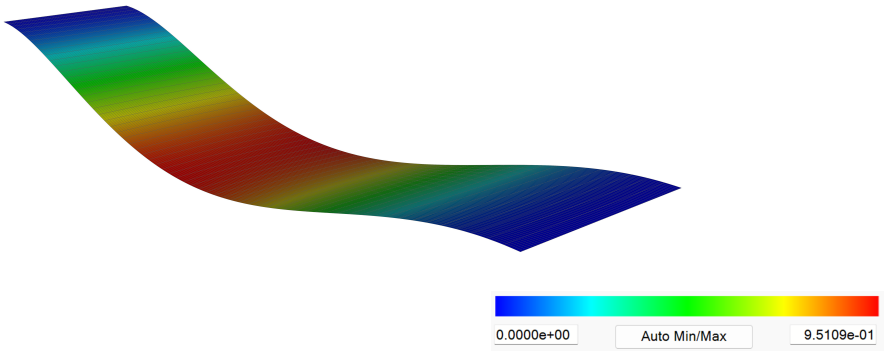


**Figure 7.11** Deflected rectangular plate with 2 side edges are supported by clamped supports. Result for Classical IGA for $p = 3$

### 7.2.2. 4 Side Edges Clamped Supported

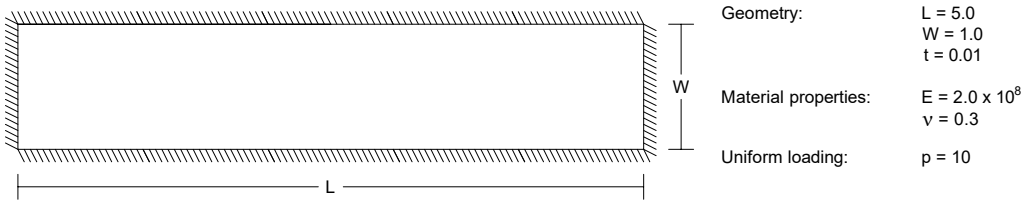


**Figure 7.12** Problem definition: Rectangular plate with 4 side edges are clamped supported

In this section, the results for the rectangular plate example having 4 side boundary edges clamped supported are discussed. The same analysis from Section 7.2.1 was also performed for the case where all four boundary edges are clamped supported. The analytical solution of the deflection in -$z$-direction is calculated as $1.422E-03$ from [52]. All the deflection values that are presented in the tables are obtained for the deflection at the center of the geometry in the -$z$-direction.

In Table 7.7, the deflection results for the NURBS-based IGA with polynomial degrees $p = 2$ and $p = 3$, and IGA with Catmull-Clark subdivision without the introduction of inner boundary

layer with 2x2 Gauss-Legendre integration points are given. The results for the 3x3 Gauss-Legendre integration scheme are given in Table 7.10 for the case without introducing inner boundary layer. According to these two tables, one can observe that the use of 3x3 Gauss-Legendre integration scheme does not have a significant effect in terms of the obtained results for the IGA+SubD case. Additionally, the results did not converge effectively compared to the NURBS-based IGA when the inner boundary layer was not introduced. This can also be observed in the plots shown in Figures 7.13 and 7.15. As can be observed from these plots, the NURBS-based IGA outperforms the IGA+SubD if the inner boundary layer is not introduced.

The results of the comparison between IGA+SubD with and without introduction of inner boundary layer are given in the Tables 7.8, 7.11, 7.9, and 7.12 for both 2x2 and 3x3 Gauss-Legendre integration schemes. Using 3x3 Gauss-Legendre integration scheme yields slightly varying results for lower levels of subdivision; however, the differences become insignificant at higher subdivision levels compared with the 2x2 Gauss-Legendre integration scheme. The relative error vs number of control points plots are given in Figures 7.14 and 7.16 for the 2x2 and 3x3 Gauss-Legendre integration schemes, respectively. From these plots, the same conclusion can be obtained with the rectangular plate example with 2 side edges clamped supported, as discussed in Section 7.2.1 for the influence of the inner boundary layer. Similarly, the clamping effect plots are given in Figures 7.17, 7.18, and 7.19. Figure 7.17 shows the deflection behavior from the section $y = 0.5$ along the $x$-axis, and one can conclude that the strong clamping effect diminishes as subdividing the geometry more. The introduction of inner boundary layer on the effect of strong clamping effect is illustrated in Figure 7.18 and Figure 7.19 in a closer look. From Figure 7.19, one can conclude that the strong clamping effect still exists at the 5th level of subdivision, which causes the slow convergence rate of the solution. Noticing that care is essential when interpreting these plots due to Paraview's linear interpolation behavior. The deflected geometries at the highest subdivision/refinement levels for the IGA+SubD and NURBS-based IGA are given in Figures 7.20, 7.21, and 7.22. For a better visualization, a scaling factor of 200 has been applied to the displacements.

**Table 7.7** Rectangular Plate with 4 side edges supported by clamped supports and 2x2 Gauss-Legendre quadrature points for IGA + SubD: No introduced inner boundary layer

| Subdivision Amount | Number of Control Points without Inner Boundary Layer | Deflection Classical IGA $p = 2$ | Deflection Classical IGA $p = 3$ | Deflection IGA+Subdivision without Inner Boundary Layer | Error Classical IGA $p = 2$ | Error Classical IGA $p = 3$ | Error IGA + Subdivision without Inner Boundary Layer |
|---|---|---|---|---|---|---|---|
| 0 | 9 | | | | | | |
| 1 | 25 | 1.120E-03 | 1.863E-03 | 7.151E-04 | 21.23% | 31.02% | 49.71% |
| 2 | 81 | 1.315E-03 | 1.474E-03 | 1.014E-03 | 7.53% | 3.69% | 28.69% |
| 3 | 289 | 1.397E-03 | 1.422E-03 | 1.210E-03 | 1.78% | 0.00% | 14.91% |
| 4 | 1089 | 1.416E-03 | 1.422E-03 | 1.318E-03 | 0.42% | 0.00% | 7.34% |
| 5 | 4225 | 1.420E-03 | 1.422E-03 | 1.370E-03 | 0.10% | 0.00% | 3.64% |

**Table 7.8** Rectangular Plate with 4 side edges supported by clamped supports and 2x2 Gauss-Legendre quadrature points for IGA + SubD: Deflections for introducing inner boundary layer

| Subdivision Amount | Number of Control Points without Inner Boundary Layer | Number of Control Points with Inner Boundary Layer | Deflection IGA+Subdivision without Inner Boundary Layer | Deflection IGA+Subdivision with Inner Boundary Layer Factor: $1/250$ | Deflection IGA+Subdivision with Inner Boundary Layer Factor: $1/500$ | Deflection IGA+Subdivision with Inner Boundary Layer Factor: $1/1000$ | Deflection IGA+Subdivision with Inner Boundary Layer Factor: $1/5000$ |
|---|---|---|---|---|---|---|---|
| 0 | 9 | 25 | | 1.967E-04 | 3.895E-04 | 9.877E-04 | 5.925E-03 |
| 1 | 25 | 81 | 7.151E-04 | 1.105E-03 | 1.102E-03 | 1.107E-03 | 1.228E-03 |
| 2 | 81 | 289 | 1.014E-03 | 1.396E-03 | 1.391E-03 | 1.391E-03 | 1.392E-03 |
| 3 | 289 | 1089 | 1.210E-03 | 1.421E-03 | 1.420E-03 | 1.418E-03 | 1.418E-03 |
| 4 | 1089 | 4225 | 1.318E-03 | 1.421E-03 | 1.422E-03 | 1.422E-03 | 1.421E-03 |
| 5 | 4225 | | 1.370E-03 | | | | |

**Table 7.9** Rectangular Plate with 4 side edges supported by clamped supports and 2x2 Gauss-Legendre quadrature points for IGA + SubD: Errors for introducing inner boundary layer

| Subdivision Amount | Number of Control Points without Inner Boundary Layer | Number of Control Points with Inner Boundary Layer | Error IGA+Subdivision without Inner Boundary Layer | Error IGA+Subdivision with Inner Boundary Layer Factor: $1/250$ | Error IGA+Subdivision with Inner Boundary Layer Factor: $1/500$ | Error IGA+Subdivision with Inner Boundary Layer Factor: $1/1000$ | Error IGA+Subdivision with Inner Boundary Layer Factor: $1/5000$ |
|---|---|---|---|---|---|---|---|
| 0 | 9 | 25 | | 86.17% | 72.61% | 30.54% | 316.72% |
| 1 | 25 | 81 | 49.71% | 22.27% | 22.52% | 22.12% | 13.65% |
| 2 | 81 | 289 | 28.69% | 1.84% | 2.15% | 2.19% | 2.11% |
| 3 | 289 | 1089 | 14.91% | 0.07% | 0.16% | 0.26% | 0.30% |
| 4 | 1089 | 4225 | 7.34% | 0.04% | 0.02% | 0.01% | 0.04% |
| 5 | 4225 | | 3.64% | | | | |

**Table 7.10** Rectangular Plate with 4 side edges supported by clamped supports and 3x3 Gauss-Legendre quadrature points for IGA + SubD: No introduced inner boundary layer

| Subdivision Amount | Number of Control Points without Inner Boundary Layer | Deflection Classical IGA $p = 2$ | Deflection Classical IGA $p = 3$ | Deflection IGA+Subdivision without Inner Boundary Layer | Error Classical IGA $p = 2$ | Error Classical IGA $p = 3$ | Error IGA + Subdivision without Inner Boundary Layer |
|---|---|---|---|---|---|---|---|
| 0 | 9 | | | | | | |
| 1 | 25 | 1.120E-03 | 1.863E-03 | 7.206E-04 | 21.23% | 31.02% | 49.32% |
| 2 | 81 | 1.315E-03 | 1.474E-03 | 1.017E-03 | 7.53% | 3.69% | 28.45% |
| 3 | 289 | 1.397E-03 | 1.422E-03 | 1.210E-03 | 1.78% | 0.00% | 14.91% |
| 4 | 1089 | 1.416E-03 | 1.422E-03 | 1.318E-03 | 0.42% | 0.00% | 7.34% |
| 5 | 4225 | 1.420E-03 | 1.422E-03 | 1.370E-03 | 0.10% | 0.00% | 3.64% |

**Table 7.11** Rectangular Plate with 4 side edges supported by clamped supports and 3x3 Gauss-Legendre quadrature points for IGA + SubD: Deflections for introducing inner boundary layer

| Subdivision Amount | Number of Control Points without Inner Boundary Layer | Number of Control Points with Inner Boundary Layer | Deflection IGA+Subdivision without Inner Boundary Layer | Deflection IGA+Subdivision with Inner Boundary Layer Factor: $1/250$ | Deflection IGA+Subdivision with Inner Boundary Layer Factor: $1/500$ | Deflection IGA+Subdivision with Inner Boundary Layer Factor: $1/1000$ | Deflection IGA+Subdivision with Inner Boundary Layer Factor: $1/5000$ |
|---|---|---|---|---|---|---|---|
| 0 | 9 | 25 | | 1.967E-04 | 2.496E-05 | 3.446E-05 | 2.919E-04 |
| 1 | 25 | 81 | 7.206E-04 | 1.105E-03 | 1.101E-03 | 1.099E-03 | 1.102E-03 |
| 2 | 81 | 289 | 1.017E-03 | 1.396E-03 | 1.399E-03 | 1.396E-03 | 1.396E-03 |
| 3 | 289 | 1089 | 1.210E-03 | 1.421E-03 | 1.420E-03 | 1.419E-03 | 1.418E-03 |
| 4 | 1089 | 4225 | 1.318E-03 | 1.421E-03 | 1.422E-03 | 1.422E-03 | 1.422E-03 |
| 5 | 4225 | | 1.370E-03 | | | | |

**Table 7.12** Rectangular Plate with 4 side edges supported by clamped supports and 3x3 Gauss-Legendre quadrature points for IGA + SubD: Errors for introducing inner boundary layer

| Subdivision Amount | Number of Control Points without Inner Boundary Layer | Number of Control Points with Inner Boundary Layer | Error IGA+Subdivision without Inner Boundary Layer | Error IGA+Subdivision with Inner Boundary Layer Factor: $1/250$ | Error IGA+Subdivision with Inner Boundary Layer Factor: $1/500$ | Error IGA+Subdivision with Inner Boundary Layer Factor: $1/1000$ | Error IGA+Subdivision with Inner Boundary Layer Factor: $1/5000$ |
|---|---|---|---|---|---|---|---|
| 0 | 9 | 25 | | 86.17% | 98.24% | 97.58% | 79.47% |
| 1 | 25 | 81 | 49.32% | 22.27% | 22.60% | 22.70% | 22.47% |
| 2 | 81 | 289 | 28.45% | 1.84% | 1.60% | 1.80% | 1.80% |
| 3 | 289 | 1089 | 14.91% | 0.07% | 0.14% | 0.17% | 0.25% |
| 4 | 1089 | 4225 | 7.34% | 0.04% | 0.02% | 0.02% | 0.03% |
| 5 | 4225 | | 3.64% | | | | |

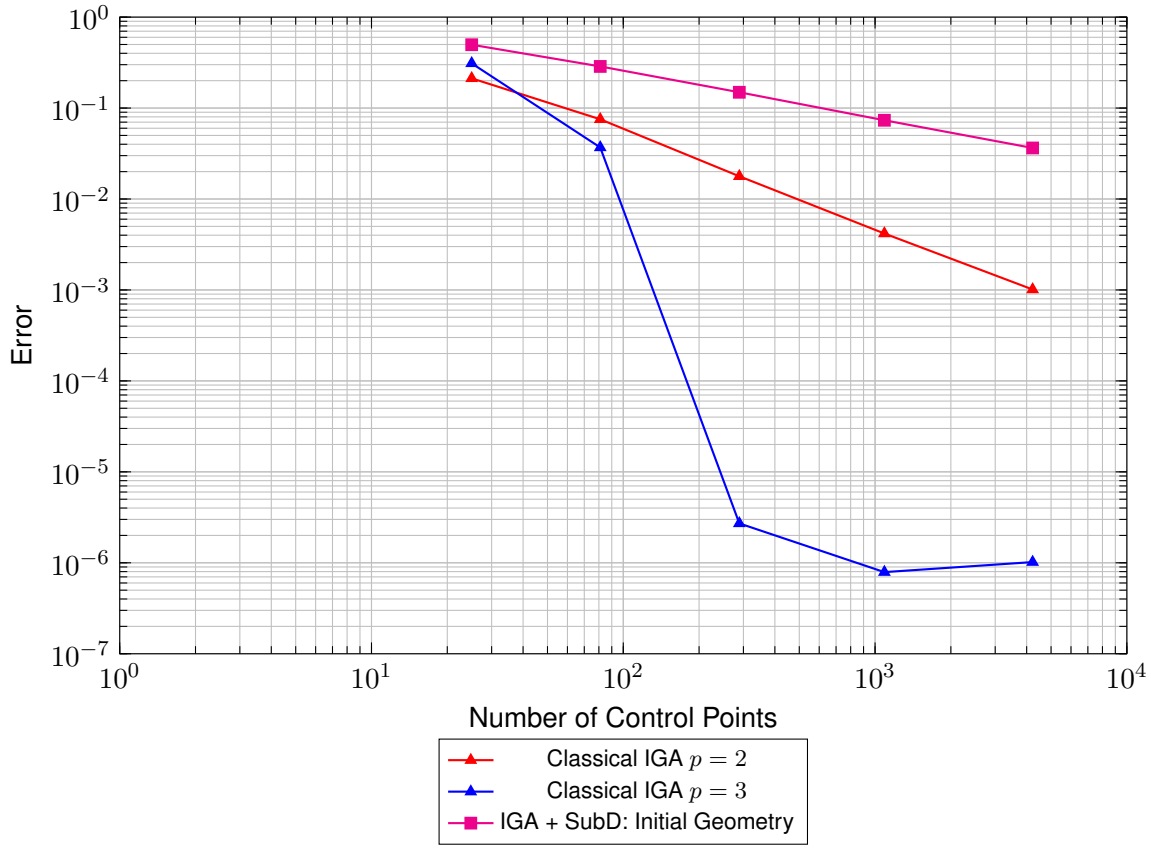


**Figure 7.13** Convergence plot: Rectangular plate with 4 side edges supported by clamped supports and 2x2 Gauss-Legendre quadrature points for IGA + SubD (w/o introduction of internal boundary layer)

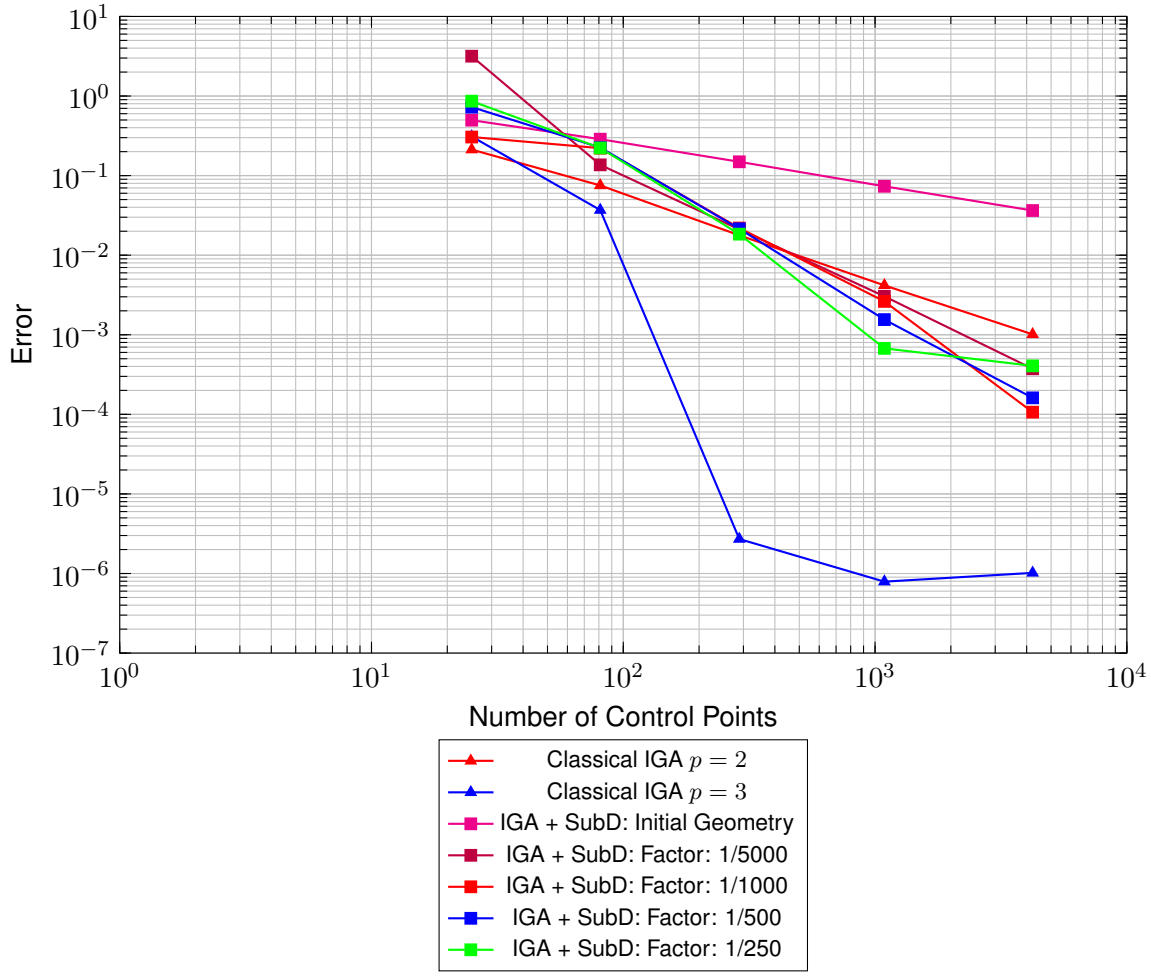


**Figure 7.14** Convergence plot: Rectangular plate with 4 side edges supported by clamped supports and 2x2 Gauss-Legendre quadrature points for IGA + SubD - Errors for different factors and initial geometry in IGA + SubD

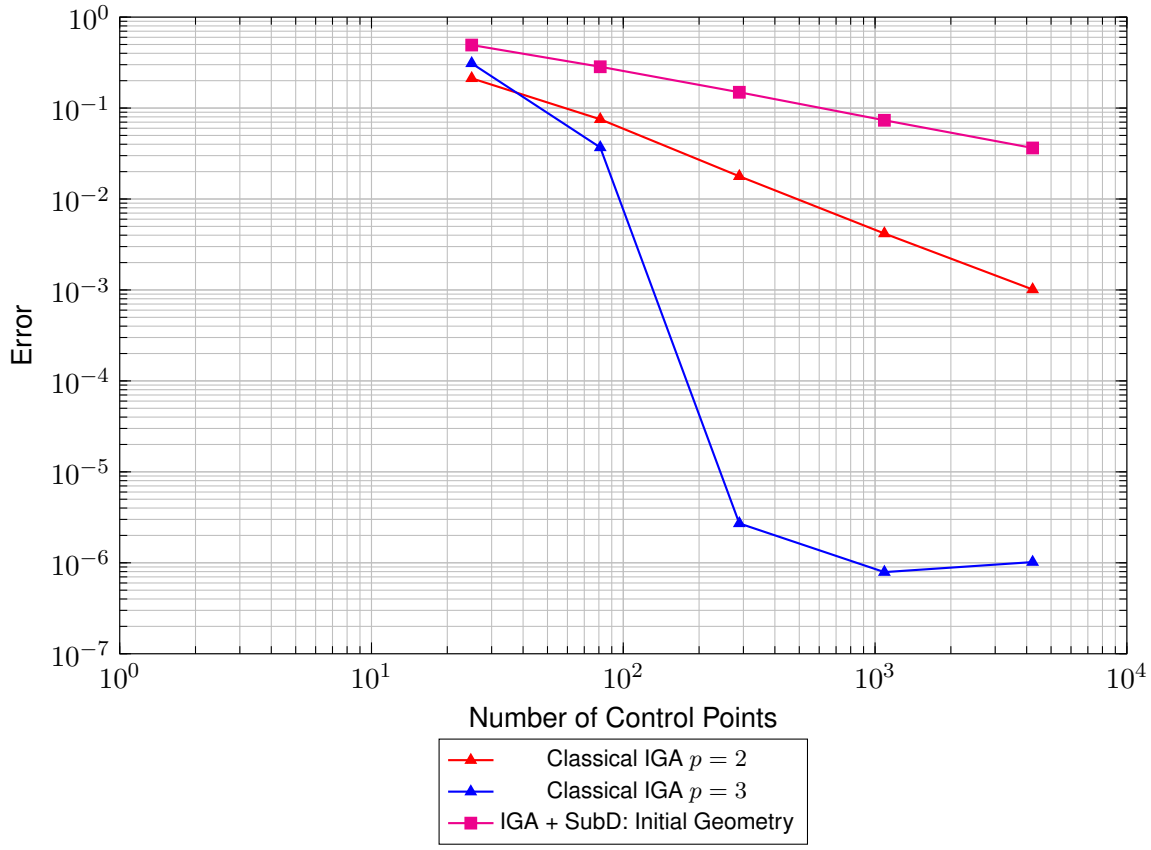


**Figure 7.15** Convergence plot: Rectangular plate with 4 side edges supported by clamped supports and 3x3 Gauss-Legendre quadrature points for IGA + SubD (w/o introduction of internal boundary layer)

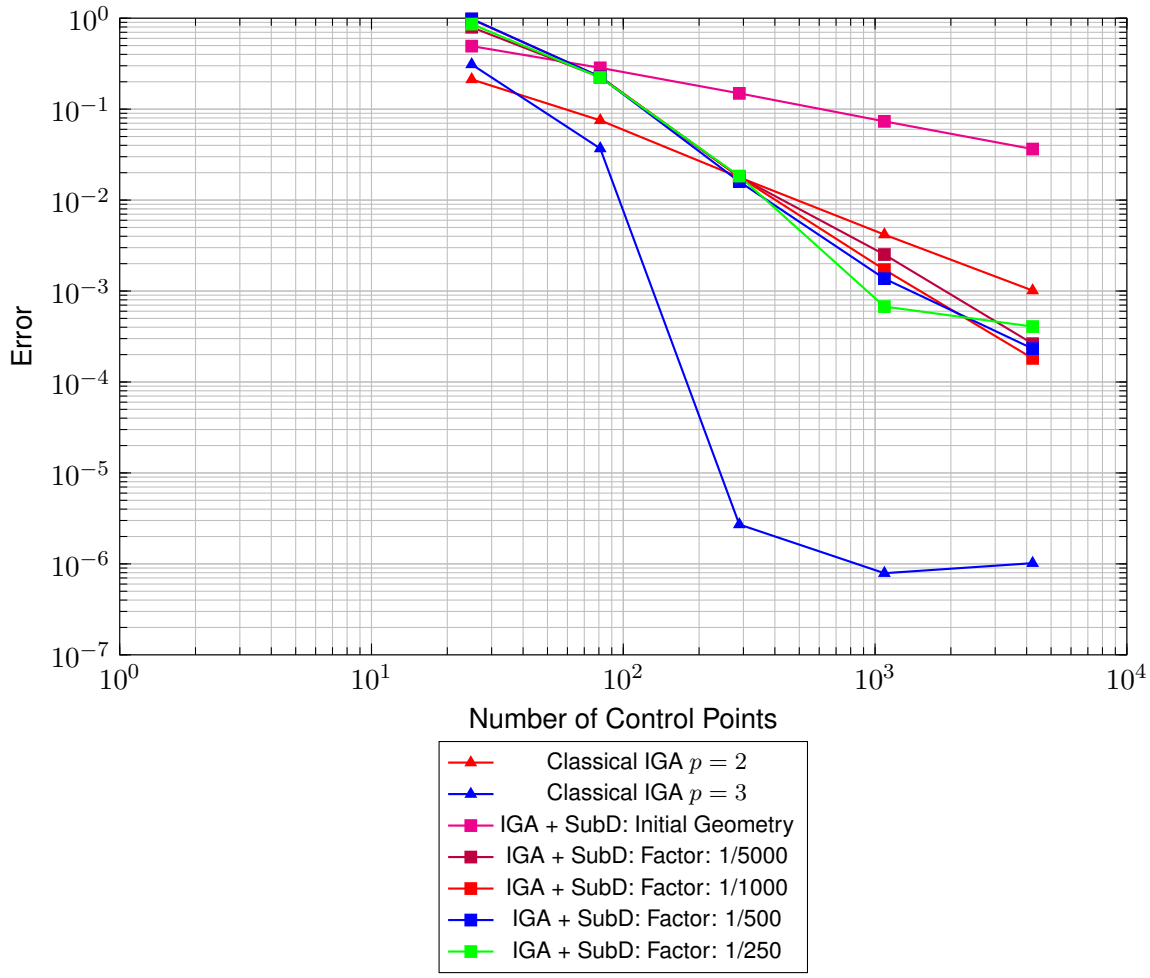


**Figure 7.16** Convergence plot: Rectangular plate with 4 side edges supported by clamped supports and 3x3 Gauss-Legendre quadrature points for IGA + SubD - Errors for different factors and initial geometry in IGA + SubD

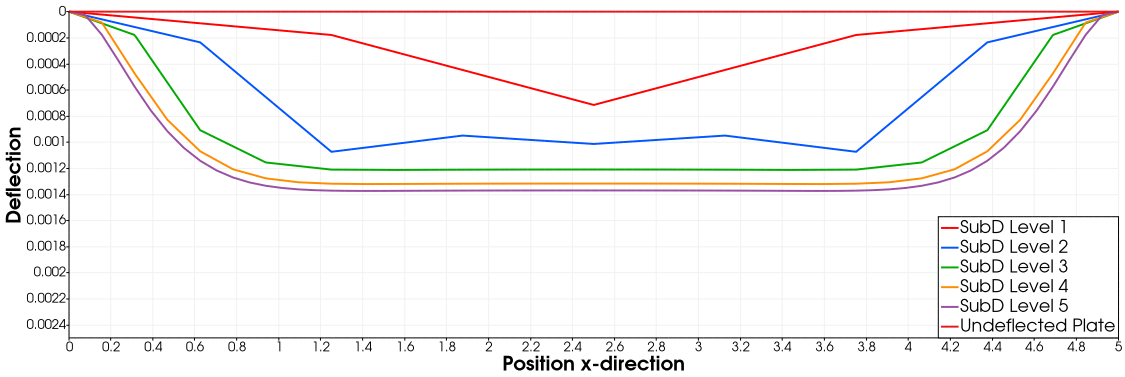


**Figure 7.17** Clamping effect at the boundaries: Rectangular plate with 4 side edges are supported by clamped supports. Results for IGA+SubD without introducing inner boundary layer and 2 Gauss-Legendre points per side. Section from $y = 0.5$

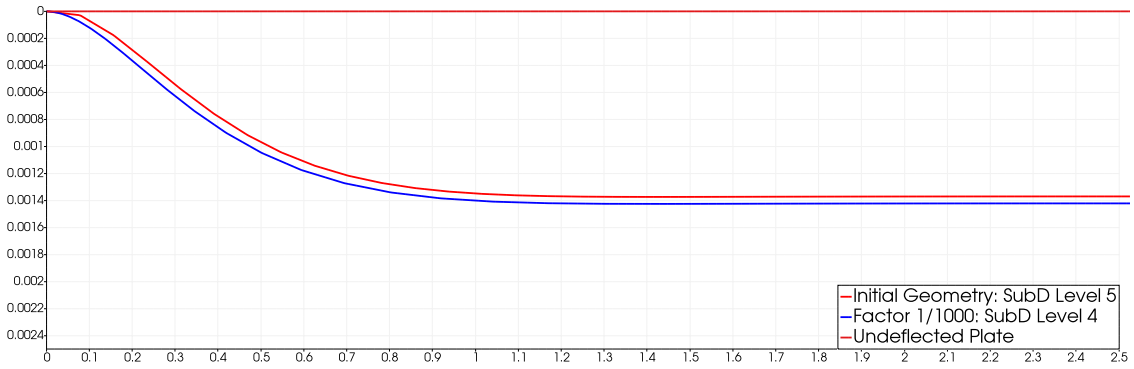


**Figure 7.18** Clamping effect at the boundaries: Rectangular plate with 4 side edges are supported by clamped supports. Results for IGA+SubD without introducing inner boundary layer and IGA+SubD with inner boundary layer with Factor = $1/1000$ and 2 Gauss-Legendre points per side. Section from $y = 0.5$

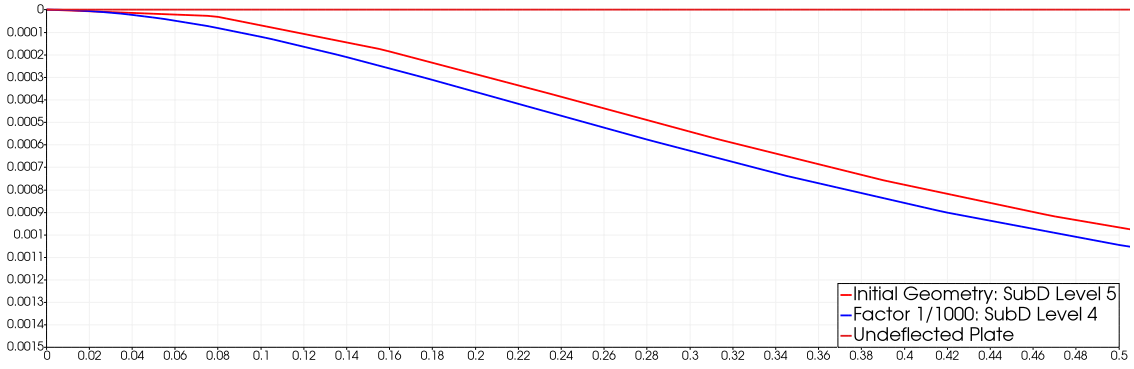


**Figure 7.19** Closer look to clamping effect at the boundary: Rectangular plate with 4 side edges are supported by clamped supports. Results for IGA+SubD without introducing inner boundary layer and IGA+SubD with inner boundary layer with Factor = $1/1000$ and 2 Gauss-Legendre points per side. Section from $y = 0.5$

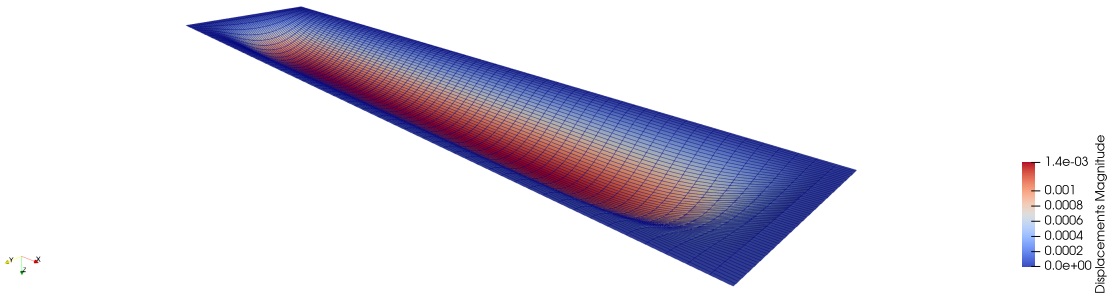


**Figure 7.20** Deflected rectangular plate with 4 side edges are supported by clamped supports. Result for IGA + SubD without introducing inner boundary layer and with 2 Gauss-Legendre points per side.

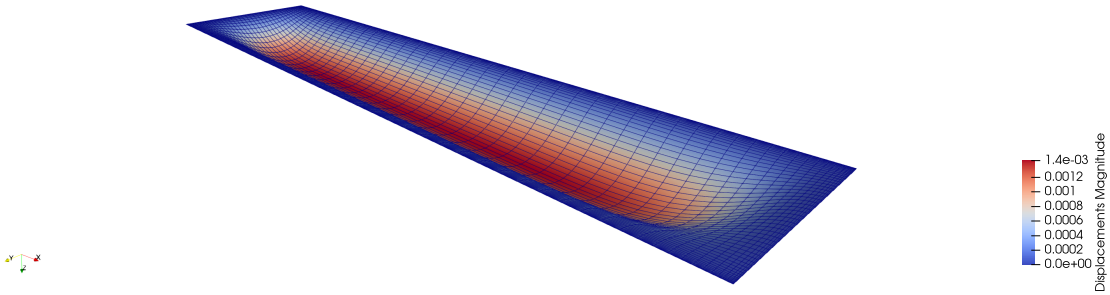


**Figure 7.21** Deflected rectangular plate with 4 side edges are supported by clamped supports. Result for IGA + SubD with introducing inner boundary layer by a factor $1/1000$ and with 2 Gauss-Legendre points per side.

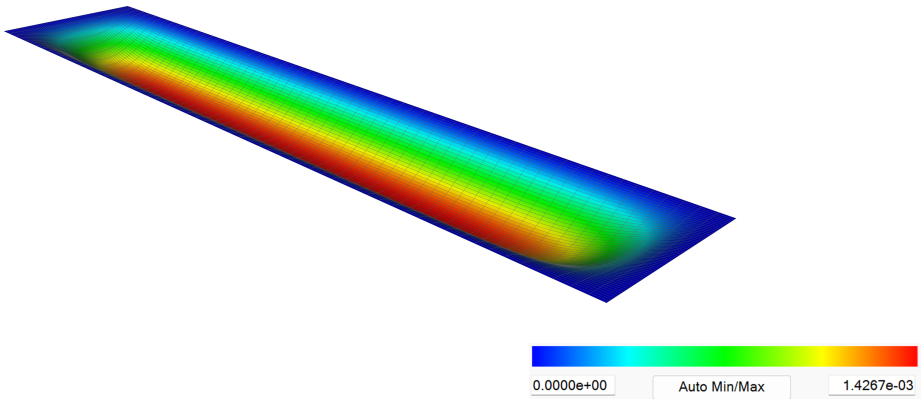


**Figure 7.22** Deflected rectangular plate with 4 side edges are supported by clamped supports. Result for Classical IGA for $p = 3$

### 7.2.3. 2 Side Edges Simply Supported

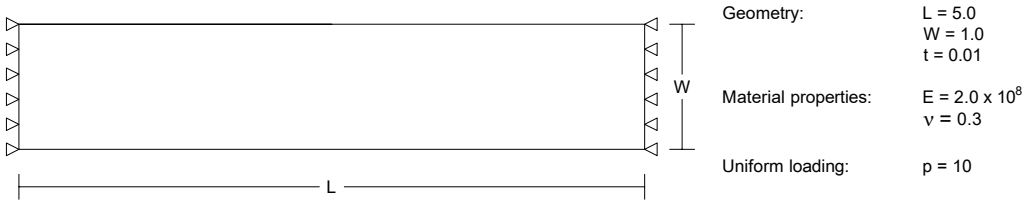


**Figure 7.23** Problem definition: Rectangular plate with 2 side edges are simply supported

In this section, the results for the rectangular plate example having 2 side boundary edges simply supported are discussed. The problem definition is given in Figure 7.23. The deflection and relative error results for the NURBS-based IGA with polynomial orders $p = 2$ and $p = 3$, and IGA+SubD with 2x2 and 3x3 Gauss-Legendre integration schemes are given in Tables 7.13 and 7.14 respectively. The analytical solution for the comparison is again obtained from [52], which is $4.840E+00$ in -$z$-direction. Noticing that, since there exists no clamped support, then the support conditions are applied only to the boundary control points, as discussed in Sections 6.1.2 and 6.2. That's why the results for the initial geometry, which contains 9 control points, can be obtained for the IGA+SubD and NURBS-based IGA for polynomial

order $p = 2$. For the case of polynomial order of $p = 3$, the geometry must be refined at least once to satisfy the condition $m = n + p + 1$ where $m$ is the number of knots, $p$ is the polynomial degree, and $n$ is the number of control points. It is also known that the open knot vectors, which have the first and last knot has a multiplicity of $p + 1$, are used to interpolate the first and last control points of the NURBS. In this case, the minimum required control point amount per axis is 4 for $p = 3$. However, the initial geometry has 3 control points per axis. That's why, at least one refinement is required for the $p = 3$ case. According to the tables, one can observe that using the 3x3 Gauss-Legendre integration scheme has no significant influence on the convergence of the displacement result. Also, the results for all the cases converge to the analytical solution much faster than the clamped supported cases. For instance, the relative error is already around $2.5\%$ for the IGA+SubD without subdivision of the geometry. In addition, the IGA+SubD outperforms NURBS-based IGA with $p = 2$ until the 4th subdivision level; however, the case of $p = 3$ already converges to the analytical solution without any further refinement. This shows that the correct polynomial degree should be $p = 3$ for the NURBS-based IGA. Relative error vs number of control points plots are given in Figures 7.24 and 7.25 for the 2x2 and 3x3 Gauss-Legendre integration schemes for IGA+SubD respectively. Lastly, the deflected geometries are given in Figures 7.26 and 7.27 for the highest level of subdivision/refinement levels for better visualization. Here, a scaling factor of 0.1 has been applied to the displacements. Again, the deflections show the same pattern, which concludes that our implementation works as expected.

**Table 7.13** Rectangular Plate with 2 side edges are simply supported and 2x2 Gauss-Legendre quadrature points for IGA + SubD

| Subdivision Amount | Number of Control Points | Deflection Classical IGA $p = 2$ | Deflection Classical IGA $p = 3$ | Deflection IGA+Subdivision | Error Classical IGA $p = 2$ | Error Classical IGA $p = 3$ | Error IGA + Subdivision |
|---|---|---|---|---|---|---|---|
| 0 | 9 | 3.811E+00 | | 4.720E+00 | 21.27% | | 2.49% |
| 1 | 25 | 4.410E+00 | 4.837E+00 | 4.772E+00 | 8.88% | 0.06% | 1.41% |
| 2 | 81 | 4.755E+00 | 4.840E+00 | 4.805E+00 | 1.76% | 0.00% | 0.72% |
| 3 | 289 | 4.821E+00 | 4.840E+00 | 4.823E+00 | 0.39% | 0.00% | 0.36% |
| 4 | 1089 | 4.836E+00 | 4.840E+00 | 4.831E+00 | 0.09% | 0.00% | 0.18% |
| 5 | 4225 | 4.839E+00 | 4.840E+00 | 4.836E+00 | 0.02% | 0.00% | 0.09% |

**Table 7.14** Rectangular Plate with 2 side edges are simply supported and 3x3 Gauss-Legendre quadrature points for IGA + SubD

| Subdivision Amount | Number of Control Points | Deflection Classical IGA $p = 2$ | Deflection Classical IGA $p = 3$ | Deflection IGA+Subdivision | Error Classical IGA $p = 2$ | Error Classical IGA $p = 3$ | Error IGA + Subdivision |
|---|---|---|---|---|---|---|---|
| 0 | 9 | 3.811E+00 | | 4.717E+00 | 21.27% | | 2.55% |
| 1 | 25 | 4.410E+00 | 4.837E+00 | 4.772E+00 | 8.88% | 0.06% | 1.42% |
| 2 | 81 | 4.755E+00 | 4.840E+00 | 4.805E+00 | 1.76% | 0.00% | 0.72% |
| 3 | 289 | 4.821E+00 | 4.840E+00 | 4.823E+00 | 0.39% | 0.00% | 0.36% |
| 4 | 1089 | 4.836E+00 | 4.840E+00 | 4.831E+00 | 0.09% | 0.00% | 0.18% |
| 5 | 4225 | 4.839E+00 | 4.840E+00 | 4.836E+00 | 0.02% | 0.00% | 0.09% |

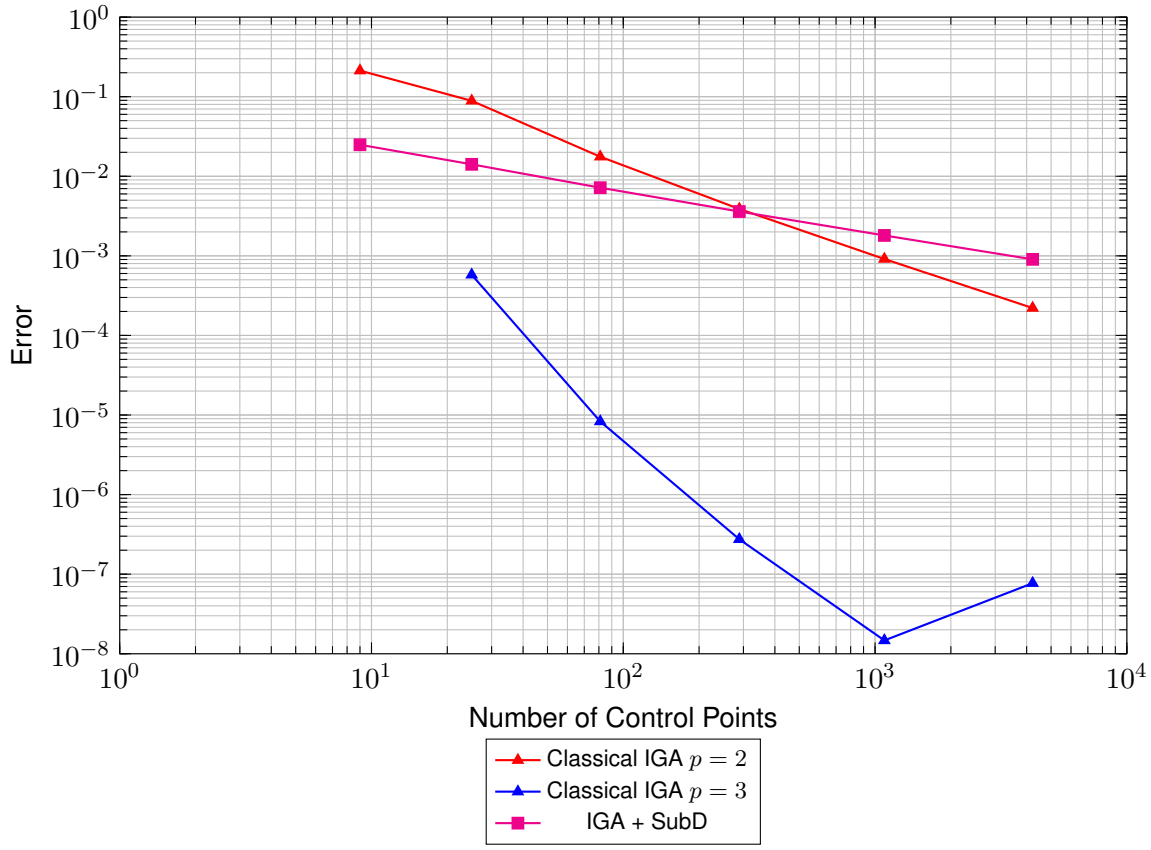


**Figure 7.24** Convergence plot: Rectangular plate with 2 side edges simply supported and 2x2 Gauss-Legendre quadrature points for IGA + SubD

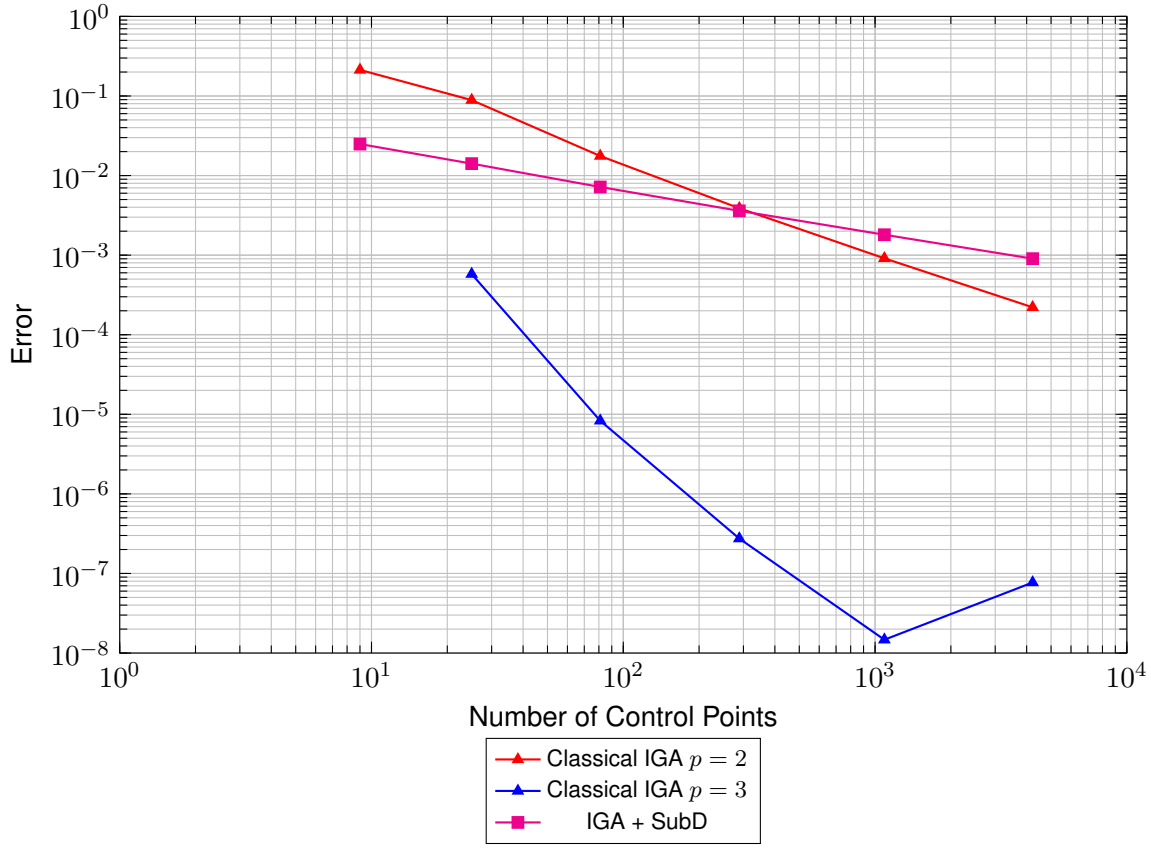


**Figure 7.25** Convergence plot: Rectangular plate with 2 side edges simply supported and 3x3 Gauss-Legendre quadrature points for IGA + SubD

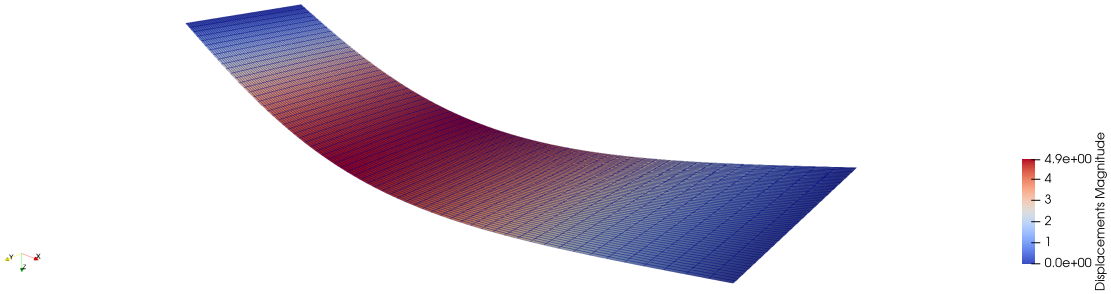


**Figure 7.26** Deflected rectangular plate with 2 side edges are supported by simply supports. Result for IGA + SubD with 2 Gauss-Legendre points per side.

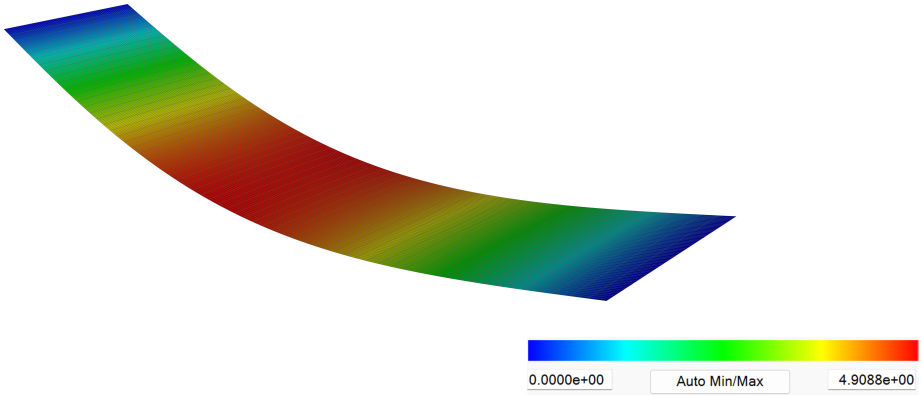


**Figure 7.27** Deflected rectangular plate with 2 side edges are supported by simply supports. Result for Classical IGA for $p = 3$

### 7.2.4. 4 Side Edges Simply Supported

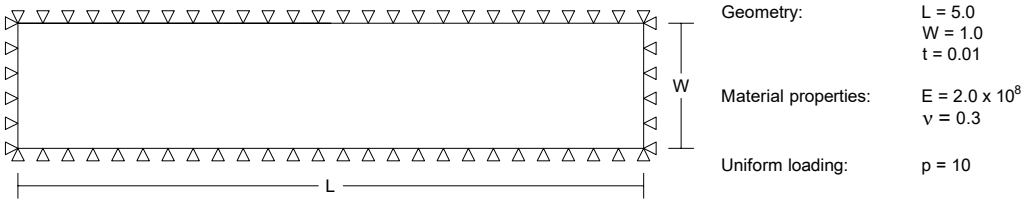


**Figure 7.28** Problem definition: Rectangular plate with 4 side edges are simply supported

In this section, the results for the rectangular plate example having 4 side boundary edges simply supported are discussed. The problem definition is given in Figure 7.28. The deflection and relative error results for the NURBS-based IGA with polynomial orders $p = 2$ and $p = 3$, and IGA+SubD with 2x2 and 3x3 Gauss-Legendre integration schemes are given in Tables 7.15 and 7.16 respectively. The analytical solution for the comparison is again obtained from [52], which is $7.082E-03$ in -$z$-direction. Noticing that, since there exists no clamped support, then the support conditions are applied only to the boundary control points, as discussed in Sections 6.1.2 and 6.2. The same explanation for the minimum required number of control points for NURBS-based IGA is valid in this example as well. The first observation from the tables is that the use of 3x3 Gauss-Legendre integration scheme has no significant effect on the results compared to 2x2 integration scheme. In addition, the results for the IGA+SubD outperform the NURBS-based IGA from the 3rd level of subdivision. This can be observed in the relative error vs number of control point plots given in Figures 7.29 and 7.30. Among the rectangular plate examples, this specific boundary condition case gives the best performance by outperforming the NURBS-based IGA with $p = 3$. The reason why the rectangular plate example with 2 simply supported cases does not outperform the NURBS-based IGA would be the existence of free ends at the boundaries, where there are no supports located. As mentioned in Green et al. [49], the simply supports and free ends also need to be formulated in a proper way. This might be a reason for the convergence result obtained for the case where the rectangular plate is supported from 2 side edges with simply supports. Finally, the

deflected geometries are given in Figures 7.31 and 7.32 for the highest subdivision/refinement levels respectively for IGA+SubD and NURBS-based IGA. For a better visualization, a scaling factor of 50 has been applied to the displacements.

**Table 7.15** Rectangular Plate with 4 side edges are simply supported and 2x2 Gauss-Legendre quadrature points for IGA + SubD

| Subdivision Amount | Number of Control Points | Deflection Classical IGA $p = 2$ | Deflection Classical IGA $p = 3$ | Deflection IGA+Subdivision | Error Classical IGA $p = 2$ | Error Classical IGA $p = 3$ | Error IGA + Subdivision |
|---|---|---|---|---|---|---|---|
| 0 | 9 | 6.655E-03 | | 8.428E-03 | 6.03% | | 19.01% |
| 1 | 25 | 6.289E-03 | 6.843E-03 | 6.747E-03 | 11.19% | 3.38% | 4.73% |
| 2 | 81 | 6.965E-03 | 7.098E-03 | 7.077E-03 | 1.65% | 0.23% | 0.07% |
| 3 | 289 | 7.057E-03 | 7.082E-03 | 7.082E-03 | 0.36% | 0.00% | 0.00% |
| 4 | 1089 | 7.076E-03 | 7.082E-03 | 7.082E-03 | 0.08% | 0.00% | 0.00% |
| 5 | 4225 | 7.081E-03 | 7.082E-03 | 7.082E-03 | 0.02% | 0.00% | 0.00% |

**Table 7.16** Rectangular Plate with 4 side edges are simply supported and 3x3 Gauss-Legendre quadrature points for IGA + SubD

| Subdivision Amount | Number of Control Points | Deflection Classical IGA $p = 2$ | Deflection Classical IGA $p = 3$ | Deflection IGA+Subdivision | Error Classical IGA $p = 2$ | Error Classical IGA $p = 3$ | Error IGA + Subdivision |
|---|---|---|---|---|---|---|---|
| 0 | 9 | 6.655E-03 | | 8.467E-03 | 6.03% | | 19.55% |
| 1 | 25 | 6.289E-03 | 6.843E-03 | 6.719E-03 | 11.19% | 3.38% | 5.12% |
| 2 | 81 | 6.965E-03 | 7.098E-03 | 7.078E-03 | 1.65% | 0.23% | 0.06% |
| 3 | 289 | 7.057E-03 | 7.082E-03 | 7.082E-03 | 0.36% | 0.00% | 0.00% |
| 4 | 1089 | 7.076E-03 | 7.082E-03 | 7.082E-03 | 0.08% | 0.00% | 0.00% |
| 5 | 4225 | 7.081E-03 | 7.082E-03 | 7.082E-03 | 0.02% | 0.00% | 0.00% |

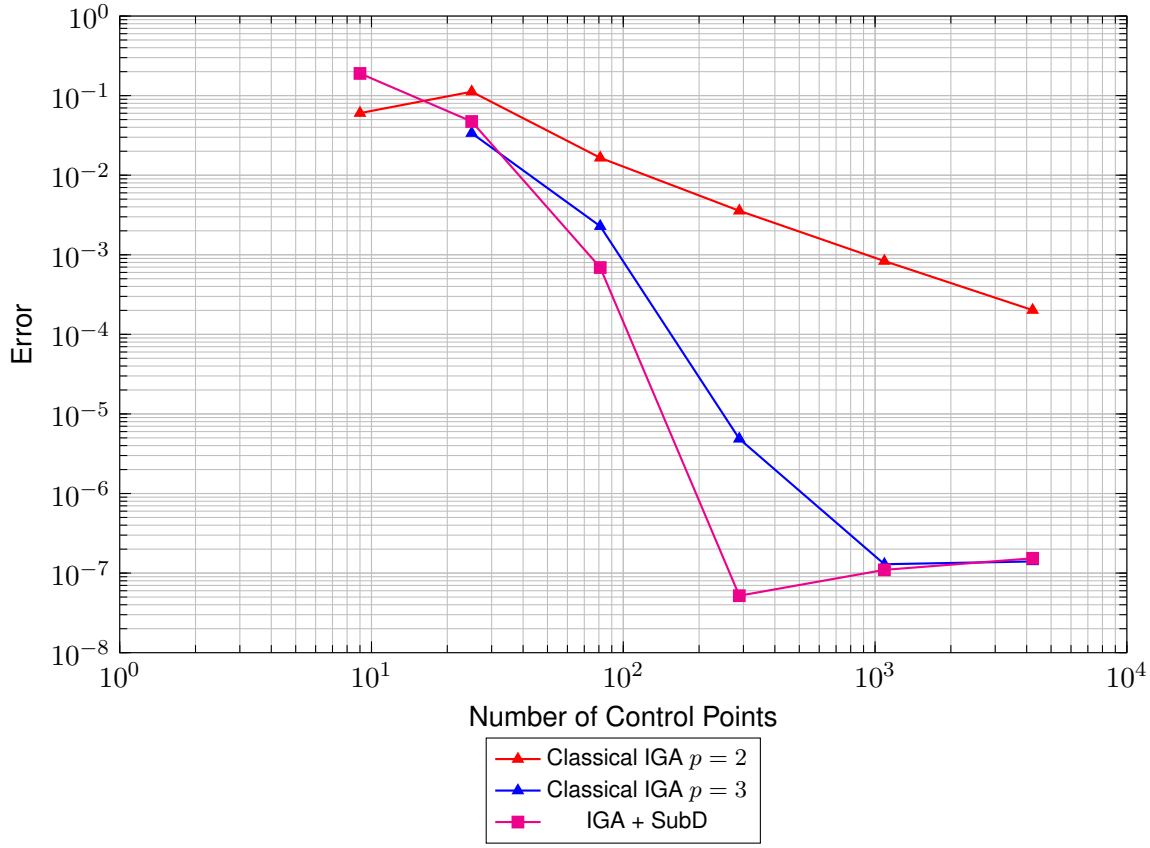


**Figure 7.29** Convergence plot: Rectangular plate with 4 side edges simply supported and 2x2 Gauss-Legendre quadrature points for IGA + SubD

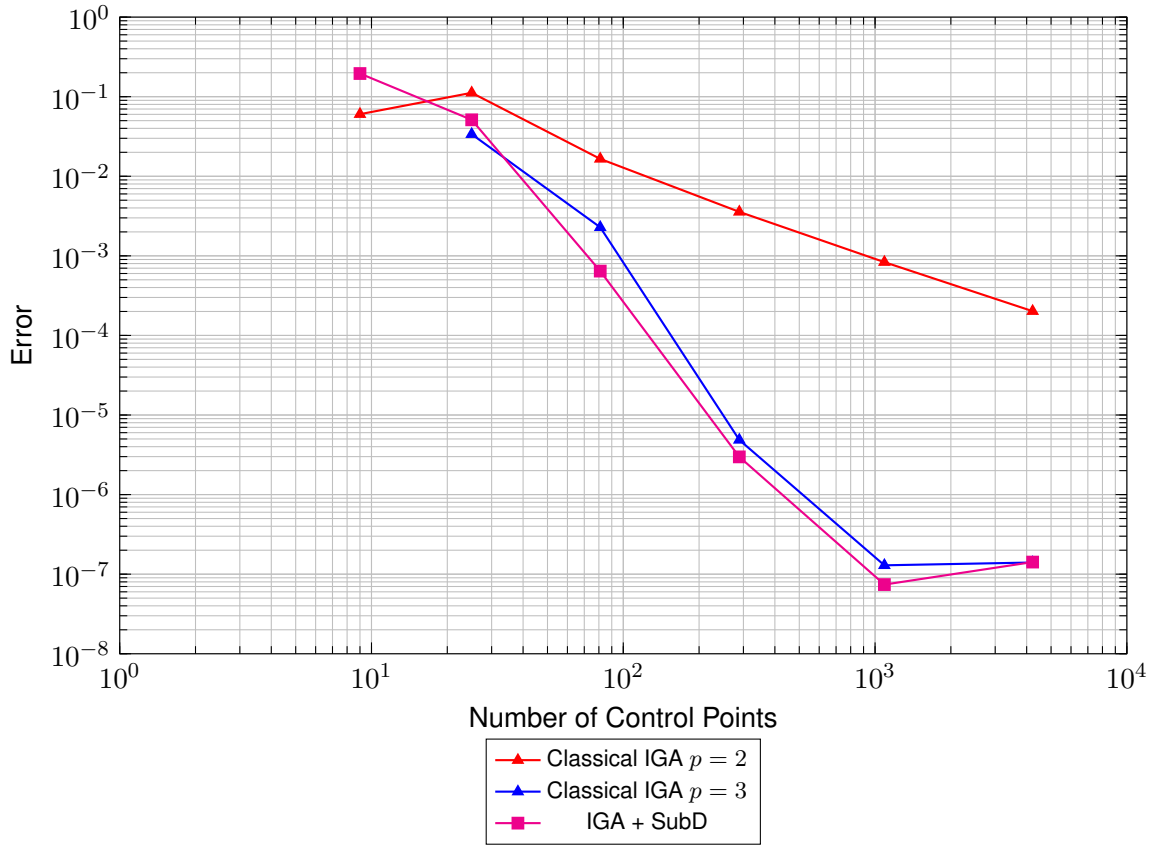


**Figure 7.30** Convergence plot: Rectangular plate with 4 side edges simply supported and 3x3 Gauss-Legendre quadrature points for IGA + SubD

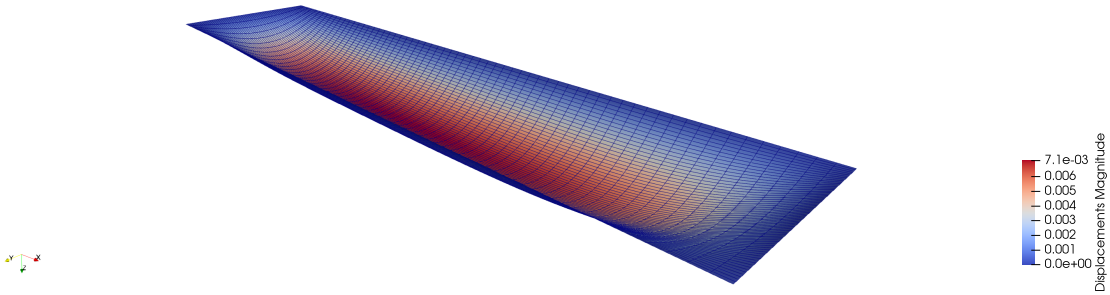


**Figure 7.31** Deflected rectangular plate with 4 side edges are supported by simply supports. Result for IGA + SubD with 2 Gauss-Legendre points per side.

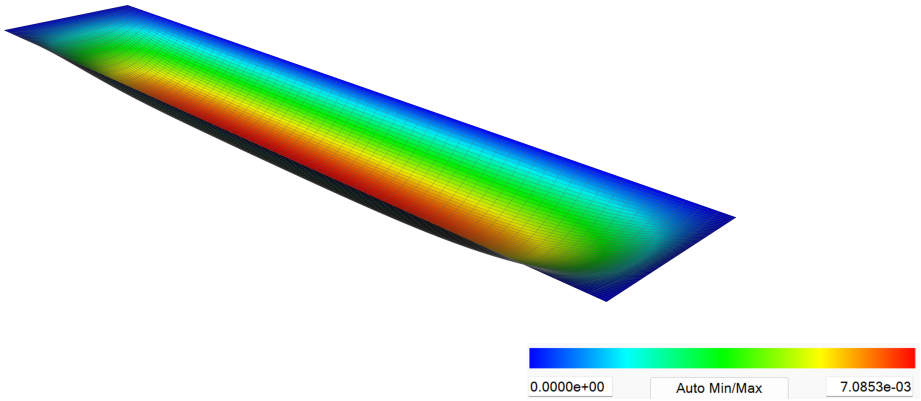


**Figure 7.32** Deflected rectangular plate with 4 side edges are supported by simply supports. Result for Classical IGA for $p = 3$

## 7.3. Scordelis-Lo Roof

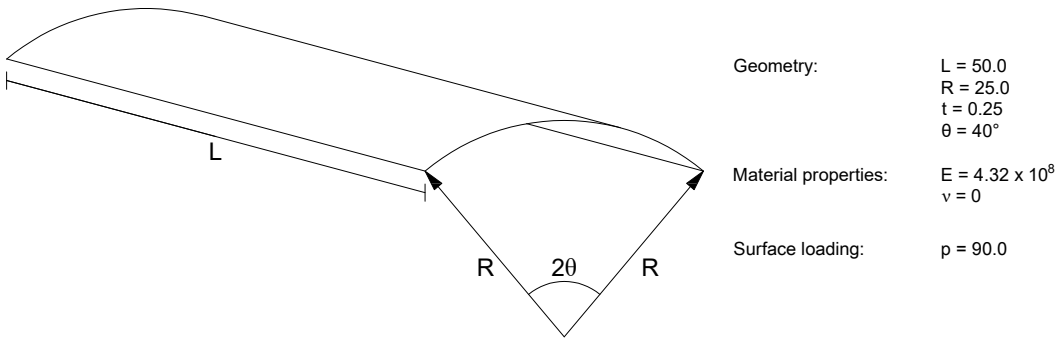


**Figure 7.33** Problem definition: Scordelis-Lo roof

In this section, the results for the Scordelis-Lo roof from Shell Obstacle Course [50] are discussed. The problem parameters and the geometric dimensions are given in Figure 7.33. The roof is supported by rigid diagrams at its ends, and the side edges remain free. A uniform gravity load of 90 is applied in -$z$-direction, and the displacements at the midpoint of side edges in -$z$-direction are considered as a reference solution. Since the roof geometry

is a section of a cylindrical shell, it is not possible to model exactly spherical and cylindrical shapes with subdivision hierarchy due to the limitations coming from bi-cubic basis functions. That's why, an optimization algorithm is developed to best represent a cylindrical section at the given level of discretization, as suggested in Green et al. [49]. The reference solution is given in [50] as $3.024E-01$. The deflection and relative error results for NURBS-based IGA and IGA+SubD are given in Tables 7.17 and 7.18 for 2x2 and 3x3 Gauss-Legendre integration scheme for IGA+SubD respectively. According to these results, one can observe that using a 3x3 integration scheme slightly changes the results for the coarser grid in IGA+SubD. However, these results did not converge to the reference solution. For the higher level of subdivision, the difference between the 2x2 and 3x3 integration schemes becomes insignificant. Additionally, one can observe that the final converged deflection amount at the 5th subdivision/refinement level is $3.006E-01$ for NURBS-based IGA, and $3.005E-01$ for IGA+SubD. The relative error vs number of control point plots are given in Figures 7.34 and 7.35 for 2x2 and 3x3 Gauss-Legendre integration schemes respectively. These plots indicate that the IGA+SubD outperforms the NURBS-based IGA with $p = 2$, but not with $p = 3$. Also, they converge almost similar deflection values at the latest subdivision/refinement level. These results also indicate that the optimization algorithm for the IGA+SubD geometry works well. Finally, the deflected geometries are given in Figures 7.36 and 7.37 with a scaling factor of 20 for a better visualization. Again, one needs to keep in mind that all the deflected shapes for IGA+SubD are obtained from Paraview, which linearly interpolates the control points of the geometry to create a surface.

**Table 7.17** Scordelis-Lo Roof with 2x2 Gauss-Legendre quadrature points for IGA + SubD

| Subdivision Amount | Number of Control Points | Deflection Classical IGA $p=2$ | Deflection Classical IGA $p=3$ | Deflection IGA+Subdivision | Error Classical IGA $p=2$ | Error Classical IGA $p=3$ | Error IGA + Subdivision |
|---|---|---|---|---|---|---|---|
| 0 | 9 | 2.054E-02 | | | 93.21% | | |
| 1 | 25 | 3.998E-02 | 2.308E-01 | 2.175E-01 | 86.78% | 23.67% | 28.09% |
| 2 | 81 | 2.077E-01 | 2.979E-01 | 2.946E-01 | 31.32% | 1.49% | 2.60% |
| 3 | 289 | 2.943E-01 | 3.006E-01 | 2.987E-01 | 2.67% | 0.60% | 1.23% |
| 4 | 1089 | 3.002E-01 | 3.006E-01 | 3.002E-01 | 0.73% | 0.60% | 0.73% |
| 5 | 4225 | 3.006E-01 | 3.006E-01 | 3.005E-01 | 0.61% | 0.60% | 0.62% |

**Table 7.18** Scordelis-Lo Roof with 3x3 Gauss-Legendre quadrature points for IGA + SubD

| Subdivision Amount | Number of Control Points | Deflection Classical IGA $p=2$ | Deflection Classical IGA $p=3$ | Deflection IGA+Subdivision | Error Classical IGA $p=2$ | Error Classical IGA $p=3$ | Error IGA + Subdivision |
|---|---|---|---|---|---|---|---|
| 0 | 9 | 2.054E-02 | | | 93.21% | | |
| 1 | 25 | 3.998E-02 | 2.308E-01 | 2.481E-01 | 86.78% | 23.67% | 17.95% |
| 2 | 81 | 2.077E-01 | 2.979E-01 | 2.963E-01 | 31.32% | 1.49% | 2.02% |
| 3 | 289 | 2.943E-01 | 3.006E-01 | 2.987E-01 | 2.67% | 0.60% | 1.21% |
| 4 | 1089 | 3.002E-01 | 3.006E-01 | 3.002E-01 | 0.73% | 0.60% | 0.73% |
| 5 | 4225 | 3.006E-01 | 3.006E-01 | 3.005E-01 | 0.61% | 0.60% | 0.62% |

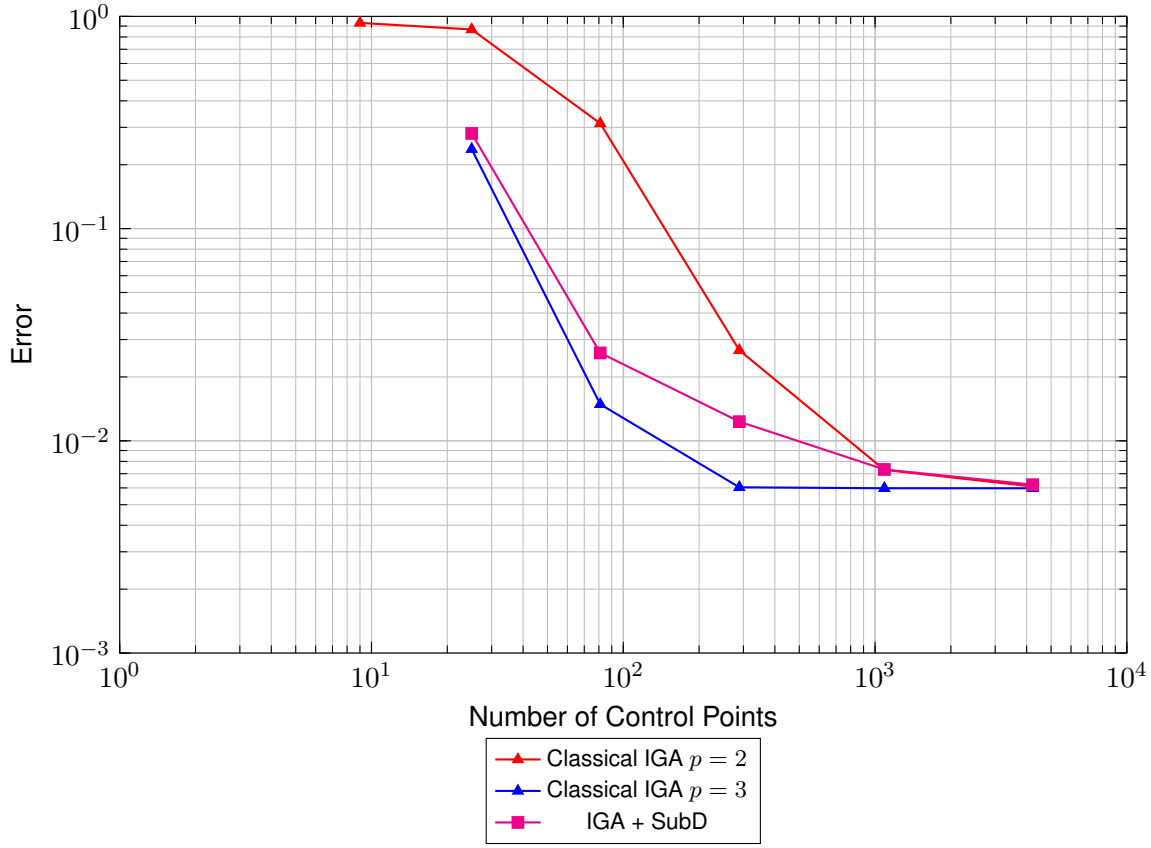


**Figure 7.34** Convergence plot: Scordelis-Lo roof with 2x2 Gauss-Legendre quadrature points for IGA + SubD

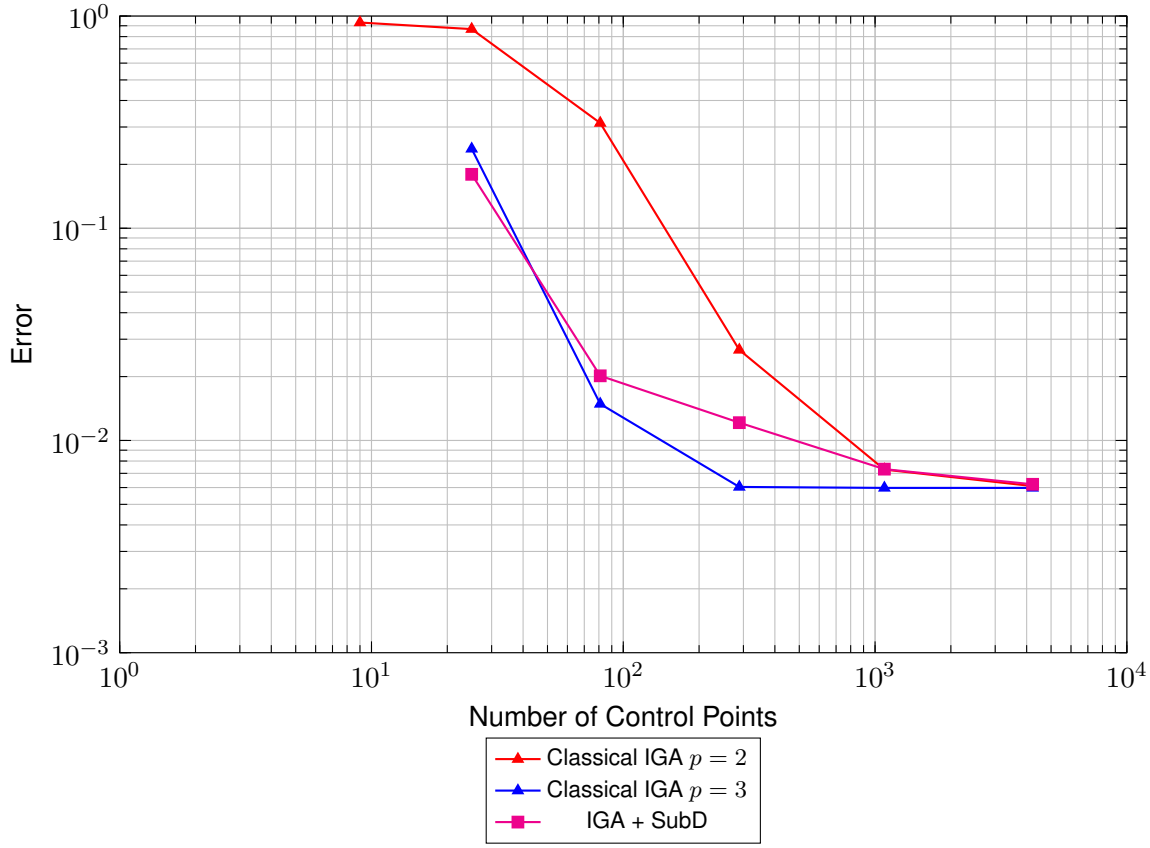


**Figure 7.35** Convergence plot: Scordelis-Lo roof with 3x3 Gauss-Legendre quadrature points for IGA + SubD

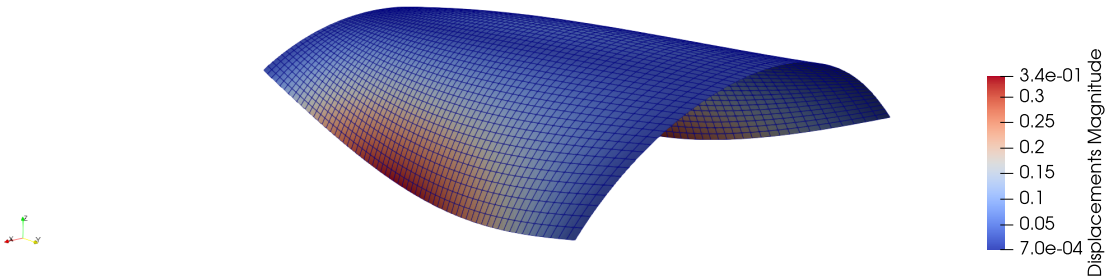


**Figure 7.36** Deflected Scordelis-Lo roof. Result for IGA + SubD with 2 Gauss-Legendre points per side.

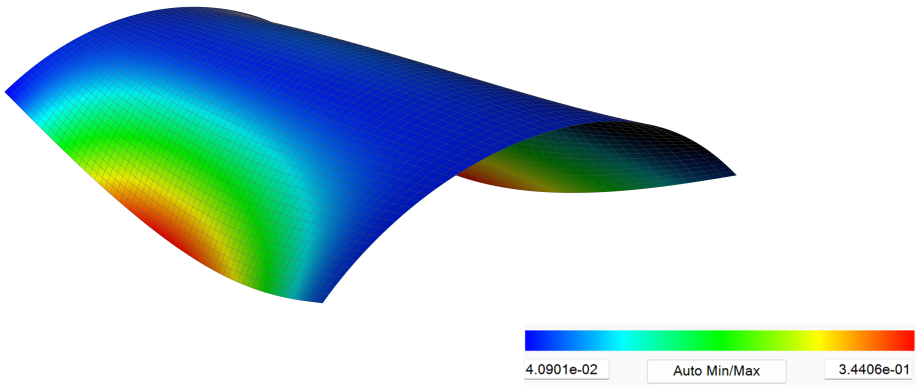


**Figure 7.37** Deflected Scordelis-Lo roof. Result for Classical IGA for $p = 3$

# 8. Conclusions and Outlook

In this thesis, isogeometric analysis using Catmull-Clark subdivision surfaces for the Kirchhoff-Love shells is conducted within the framework of Kratos Multiphysics. The results showed that the developed subdivision algorithm can be adapted to the "$IgaApplication$" of the Kratos Multiphysics. During this work, the significance of an appropriate subdivision scheme was emphasized, particularly regarding the definition of subdivision rules for boundaries and creases. The reason is that properly addressing boundaries and creases is essential in isogeometric analysis with subdivision surfaces. That's why the extended Catmull-Clark subdivision algorithm was implemented and used during the thesis work. In addition, the importance of the correct formulation of support conditions at the boundaries was also investigated. In this thesis, a similar approach defined by Cirak et al. [33] was implemented for the imposing of boundary conditions. The results indicate that this implementation experiences slow convergence rates, particularly in cases of clamped support. A correct way of formulating the support conditions at the boundaries was introduced by Green et al. [49]; however, the suggested implementations were not implemented within the scope of this thesis due to implementation difficulties in Kratos Multiphysics and time limitations of the work. Instead, an inner boundary layer with a factor is introduced inside the geometry for the rectangular plate examples with clamped supports. With the introduction of this inner boundary layer, the strong clamping effect that comes from implementing the support conditions at the boundaries was reduced, resulting in better and faster-converging results. For the simply supported rectangular plate examples, the results obtained are satisfactory. However, faster convergence might be achieved if the formulations suggested by Green et al. [49] are implemented. The results obtained for the Scordelis-Lo roof are also satisfactory because both the NURBS-based isogeometric analysis and subdivision-based isogeometric analysis converged to almost the same deflection value. Additionally, the effect of polynomial order in NURBS-based isogeometric analysis and the number of Gauss-Legendre integration points in the subdivision-based isogeometric analysis were observed. According to the results, the polynomial order significantly changed the results of NURBS-based isogeometric analysis, because the predefined value of polynomial degree $p = 2$ was not sufficient to represent the geometries, and it did not make sense to compare the results with $p = 2$ to subdivision-based isogeometric analysis due to bi-cubic basis function characteristics of the Catmull-Clark subdivision scheme. Instead, polynomial degree $p = 3$ was tested, and the results showed that the NURBS-based isogeometric analysis with the correct polynomial degree converges much faster and gives better results. Moreover, it was shown that using the 2x2 or 3x3 Gauss-Legendre scheme did not change the results significantly because the 2x2 Gauss-Legendre scheme already guarantees the exactness of the integration within the bicubic polynomial space in the case of Catmull-Clark subdivision. Finally, some progress has been made for the irregular geometries; however, the results were not obtained for this case due to implementation difficulties in Kratos Multiphysics and the time limitation of the thesis work. To date, the achievement regarding irregular geometries involves performing local subdivisions around the first ring faces

of extraordinary vertices. This step is necessary for integration near those vertices.

As an outlook, further developments in the implementation of Catmull-Clark subdivision-based isogeometric analysis within the Kratos Multiphysics framework can be collected under several topics. First of all, all the subdivision-related codes were implemented in Python because of the language's easy interpretability and the thesis work's time limitations. However, this resulted in slow computation times, especially for the high level of subdivision cases. Instead, all the algorithms would be developed in C++, and Python can be used again to run the simulations, as in the case of "$IgaApplicaiton$" in Kratos Multiphysics. Secondly, a new application would be created to deal with subdivision-based isogeometric analysis instead of integrating the subdivision rules to "$IgaApplication$" in Kratos Multiphysics. Although both NURBS-based isogeometric analysis and subdivision-based isogeometric analysis rely on the same element formulation, implementing the subdivision algorithms in "$IgaApplication$" brought some difficulties during the implementation. For instance, defining the control mesh of geometry for the subdivision-based analysis required a manual definition of the faces. Due to this issue, the test cases, even for regular geometries, are very limited in this work. Implementing an algorithm that automatically determines the faces with control point relations or importing the geometry in "vtk" format and extracting the control points and faces from the CAD model would be a solution to this issue. Moreover, it is crucial to implement the correct support formulations at the boundaries to obtain better results and to avoid strong clamping effects that arise from the implementation with the ghost vertices and elements given by Cirak et al.[33]. After addressing all these issues for further development, the application will be extended to analyze geometries with irregular elements by correctly adapting the locally subdivided sub-geometries into the analysis stage, specifically regarding the correct assignment in the stiffness matrix.

# List of Figures

# Index of Abbreviations

## C

CAD - Computer Aided Design ..... 1, 2, 22, 28
CAE - Computer Aided Engineering ..... 2

## F

FEA - Finite Element Analysis ..... 1, 2, 22–24, 27
FEM - Finite Element Method ..... 17, 23

## I

IGA - Isogeometric Analysis ..... 1–3, 22, 23, 25, 28, 44, 46, 57–65, 67–73, 75–81, 83–85, 91–93, 95, 96, II

## N

NURBS - Non-uniform Rational B-Splines 2, 3, 22–25, 27, 28, 57–60, 67, 68, 75, 76, 79, 80, 83, 87–89, II

## S

SubD - Subdivision Surfaces. 22, 25, 28, 44, 60–65, 68–73, 75–81, 83–85, 91–93, 95, 96

# List of Tables

# Bibliography


[1] J. Kiendl. *Isogeometric Analysis and Shape Optimal Design of Shell Structures*. PhD thesis, Technische Universität München, 2011.

[2] G. Kirchhoff. Über das gleichgewicht und die bewegung einer elastischen scheibe. *Journal für die reine und angewandte Mathematik*, 40:51–88, 1850.

[3] A.E.H. Love. On the small vibrations and deformations of thin elastic shells. *Philosophical Transactions of the Royal Society of London. (A.)*, 179:491–546, 1888.

[4] E. Reissner. The effect of transverse shear deformation on the bending of elastic plates. *ASME Journal of Applied Mechanics*, 12:69–77, 1945.

[5] T.J.R. Hughes, J.A. Cottrell, and Y. Bazilevs. Isogeometric analysis: Cad, finite elements, nurbs, exact geometry and mesh refinement. *Computer Methods in Applied Mechanics and Engineering*, 194(39):4135–4195, 2005.

[6] J.A. Cottrell, T.J.R. Hughes, and Y. Bazilevs. *Isogeometric Analysis: Toward Integration of CAD and FEA*. John Wiley Sons, 2009.

[7] J. Kiendl, K.-U. Bletzinger, J. Linhard, and R. Wüchner. Isogeometric shell analysis with kirchhoff–love elements. *Computer Methods in Applied Mechanics and Engineering*, 198(49):3902–3914, 2009.

[8] K. Schweizerhof and E. Ramm. Displacement dependent pressure loads in nonlinear finite element analyses. *Computers & Structures*, 18:1099–1114, 1984.

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

[10] F. Cirak and M. Ortiz. Fully c1-conforming subdivision elements for finite deformation thin-shell analysis. *International Journal for Numerical Methods in Engineering*, 51(7):813–833, 2001.

[11] T. Oberbichler and K. U. Bletzinger. Cad-integrated form-finding of structural membranes using extended catmull–clark subdivision surfaces. *Computer-Aided Design*, 151:103360, 2022.

[12] B. Hamann, D. Burkhart, and G. Umlauf. Isogeometric finite element analysis based on catmull–clark subdivision solids. *Computer Graphics Forum*, 29:1575–1584, 2010.

[13] 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*, 291(1):1–20, 2015.

[14] Q. Pan, G. Xu, G. Xu, and Y. Zhang. Isogeometric analysis based on extended loop's subdivision. *Journal of Computational Physics*, 299(15):731–746, 2015.

[15] Q. Pan, G. Xu, G. Xu, and Y. Zhang. Isogeometric analysis based on extended catmull–clark subdivision. *Computers & Mathematics with Applications*, 71:105–119, 2016.

[16] P. Dadvand, R. Rossi, and E. Oñate. An object-oriented environment for developing finite element codes for multi-disciplinary applications. *Archives of Computational Methods in Engineering*, 17:253–297, 2010.

[17] P. Dadvand, R. Rossi, M. Gil, X. Martorell, Cotela J., Juanpere E., S.R. Idelsohn, and E. Oñate. Migration of a generic multi-physics framework to hpc environments. *Computers Fluids*, 80:301–309, 2013. Selected contributions of the 23rd International Conference on Parallel Fluid Dynamics ParCFD2011.

[18] V. Mataix Ferrándiz, P. Bucher, R. Zorrilla, S. Warnakulasuriya, R. Rossi, A. Cornejo, J. Cotela, C. Roig, J. Maria, T. Teschemacher, M. Masó, G. Casas, M. Núñez, P. Dadvand, S. Latorre, I. de Pouplana, J. Irazábal González, A. Franci, F. Arrufat, R. Tosi, A. Ghantasala, K. B. Sautter, P. Wilson, D. Baumgaertner, B. Chandra, A. Geiser, I. Lopez, J. G. Usua, and J. Gárate. Kratosmultiphysics/kratos: Release v9.5, April 2024.

[19] I. N. Bronstein, K. A. Semendjajew, G. Musiol, and H. Mühlig. *Taschenbuch der Mathematik*. Verlag Harri Deutsch, Frankfurt am Main, 2001.

[20] L. A. Piegl and W. Tiller. *The NURBS book*. Springer, Berlin and and New York, 2 edition, op. 1997.

[21] R. Wüchner, M. Breitenberger, and A. Bauer. *Isogeometric Structural Analysis and Design*. Technische Universität München, Chair of Structural Analysis, 2019. Lecture notes, Summer Term 2019.

[22] D.F. Rogers. *An Introduction to NURBS - With Historical Perspective*. Academic Press, San Francisco, CA, 1st edition, 2001.

[23] E. Klingbeil. *Tensorrechnung fur Ingenieure*. Bibliographisches Institut & F.A Brockhaus AG, Zürich, 2nd edition, 1989.

[24] G.A. Holzapfel. *Nonlinear solid mechanics*. John Wiley & Sons, Chichester, 2004.

[25] Y. Basar and D. Weichert. *Nonlinear continuum mechanics of solids*. Springer Verlag, Berlin, Heidelberg, New York, 2000.

[26] W. Wunderlich and W.D. Pilkey. *Mechanics of Structures, Variational and Computational Methods*. CRC Press, 2nd edition, 2003.

[27] O.C. Zienkiewicz, R.L. Taylor, and J.Z. Zhu. *The Finite Element Method, Its Basis and Fundamentals*. Elsevier, Oxford, 2005.

[28] Y. Bazilevs, V.M. Calo, J.A. Cottrell, J.A. Evans, T.J.R. Hughes, S. Lipton, M.A. Scott, and T.W. Sederberg. Isogeometric analysis using t-splines. *Computer Methods in Applied Mechanics and Engineering*, 199:229–263, 2010.

[29] M. Dörfel, B. Jüttler, and B. Simeon. Adaptive isogeometric analysis by local h-refinement with t-splines. *Computer Methods in Applied Mechanics and Engineering*, 199:264–275, 2010.

[30] T.-K. Uhm, Y. D. Kim, and S.-K. Youn. A locally refinable t-spline finite element method for cad/cae integration. *Structural Engineering and Mechanics*, 80:225–245, 2008.

[31] T.-K. Uhm and S.-K. Youn. T-spline finite element method for the analysis of shell structures. *International Journal for Numerical Methods in Engineering*, 80:507–536, 2009.

[32] D. J. Benson, Y. Bazilevs, E. De Luycker, M.-C. Hsu, M. Scott, T. J. R. Hughes, and T. Belytschko. A generalized finite element formulation for arbitrary basis functions: From isogeometric analysis to xfem. *International Journal for Numerical Methods in Engineering*, 83:765–785, 2010.

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

[34] J.A. Cottrell, Hughes T.J.R., and A. Reali. Studies of refinement and continuity in isogeometric structural analysis. *Computer Methods in Applied Mechanics and Engineering*, 196(41):4160–4183, 2007.

[35] J. Linhard, R. Wüchner, and K.-U. Bletzinger. Upgrading membranes to shells – the ceg rotation free shell element and its application in structural analysis. *Finite Elements in Analysis and Design*, 44:63–74, 2007.

[36] E. Oñate and F. Zárate. Rotation-free triangular plate and shell elements. *International Journal for Numerical Methods in Engineering*, 47:557–603, 2000.

[37] Z. Liu, A. McBride, P. Saxena, and S. Bordas. Assessment of an isogeometric approach with catmull–clark subdivision surfaces using the laplace–beltrami problems. *Computational Mechanics*, 66:851–876, 2020.

[38] L. Kobbelt, T. Hesse, H. Prautzsch, and K. Schweizerhof. Iterative mesh generation for fe-computations on free form surfaces. *Engineering Computations*, 14:806–820, 1997.

[39] N. Dyn, D. Levin, and J. A. Gregory. A butterfly subdivision scheme for surface interpolation with tension control. *ACM Transactions on Graphics*, 9(2):160–169, 1990.

[40] D. Zorin, P. Schröder, and W. Sweldens. Interpolating subdivision for meshes with arbitrary topology. In *Computer Graphics (SIGGRAPH '96 Proceedings)*, pages 189–192, 1996.

[41] C. Loop. Smooth subdivision surfaces based on triangles. Master's thesis, University of Utah, Department of Mathematics, 1987.

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

[43] J. Peters and U. Reif. Analysis of algorithms generalizing b-spline subdivision. *SIAM Journal on Numerical Analysis*, 35(2):728–748, 1998.

[44] J. Stam. Exact evaluation of catmull–clark subdivision surfaces at arbitrary parameter values. In *SIGGRAPH Course Notes*, pages 395–404, 1998.

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

[46] T. Nguyen, K. Karciauskas, and J. Peters. A comparative study of several classical, discrete differential and isogeometric methods for solving poisson's equation on the disk. *Axioms*, 3(2):280–299, 2014.

[47] T. Takacs and B. Jüttler. $h^2$ regularity properties of singular parameterizations in isogeometric analysis. *Graphical Models*, 74(6):361–372, 2012.

[48] H. Biermann, A. Levin, and D. Zorin. Piecewise smooth subdivision surfaces with normal control. In *Proceedings of the 27th Annual Conference on Computer Graphics and Inter-*

*active Techniques*, pages 113–120. ACM Press/Addison-Wesley Publishing Co., 2000.

[49] S. Green and G. Turkiyyah. Second-order accurate constraint formulation for subdivision finite element simulation of thin shells. *International Journal for Numerical Methods in Engineering*, 61(3):380–405, 2004.

[50] T. Belytschko, H. Stolarski, W. K. Liu, N. Carpenter, and J. S.-J. Ong. Stress projection for membrane and shear locking in shell finite elements. *Computer Methods in Applied Mechanics and Engineering*, 51:221–258, 1985.

[51] P. J. Barendrecht, Bartoň M., and J. Kosinka. Efficient quadrature rules for subdivision surfaces in isogeometric analysis. *Computer Methods in Applied Mechanics and Engineering*, 340:1–23, 2018.

[52] M. Batista. Uniformly loaded rectangular thin plates with symmetrical boundary conditions, 2010.

# Declaration

I hereby declare that the thesis submitted is my own unaided work. All direct or indirect sources used are acknowledged as references. In addition, I declare that I make the present work available to the Chair of Structural Analysis and Dynamics for academic purposes and in this connection also approve of dissemination for academic purposes.

Munich, 30/12/2024, Signature