



Norwegian University of
Science and Technology

Implementation of isogeometric thin shell analysis within an object oriented computing environment

Simen Skogholt Haave
Marit Rakvåg

Master of Science in Engineering and ICT

Submission date: June 2018

Supervisor: Kjell Magne Mathisen, KT

Co-supervisor: Knut Morten Okstad, SINTEF

Norwegian University of Science and Technology
Department of Structural Engineering



MASTER THESIS 2018

SUBJECT AREA: Computational mechanics	DATE: June 11, 2018	NO. OF PAGES: 117 (110 + 7)
--	------------------------	--------------------------------

TITLE: Implementation of isogeometric thin shell analysis within an object oriented computing environment Implementering av isogeometrisk tynnskal analyse i et objektorientert programmeringsmiljø
BY: Simen Skogholt Haave Marit Gaarder Rakvåg

SUMMARY: <p>The drawback in today's computational analysis in structural engineering is that design and analysis are not integrated in a unified platform. A majority of the time spent in structural analysis is used to transform a design model into an analysis suitable model. Furthermore, this process can induce a geometry which is based on approximations. Isogeometric analysis (IGA) was developed to remedy this loss of efficiency as it aims to unify the geometrical representation in both design and analysis. NURBS (Non-Uniform Rational B-Splines) are the most widespread mathematical basis in computer-aided design (CAD) as it is able to depict the geometry in an exact manner, which makes NURBS a suitable basis for an IGA.</p> <p>The purpose of this thesis is to implement the isogeometric rotation-free Kirchhoff-Love thin shell element proposed by Kiendl et al. (2009), for both geometrically linear and nonlinear analyses within an object-oriented environment. The element is integrated into the open-source software IFEM Elasticity developed by SINTEF Digital, which is part of a research project devoted to merging design and analysis in computational mechanics.</p> <p>To verify the implementation and the element's performance, a set of both linear and nonlinear benchmark problems have been simulated. The performance has been evaluated through a convergence study for the linear cases, while the nonlinear cases have been compared to results obtained by Lagrange-based finite element analysis (FEA) simulations. The results of the implemented shell element yielded a coherence with the reported results in relevant scientific publications.</p>

RESPONSIBLE TEACHER: Kjell Magne Mathisen SUPERVISORS: Kjell Magne Mathisen and Knut Morten Okstad CARRIED OUT AT: Department of Structural Engineering, NTNU

TKT4915 Beregningsmekanikk, masteroppgave

Masteroppgave 2018

for

Simen Skogholt Haave og Marit Rakvåg

Implementering av isogeometrisk tynns skall analyse i et objekt orientert programmeringsmiljø

Implementation of isogeometric thin shell analysis within an object oriented computing environment

Isogeometric analysis was introduced by Hughes *et al.* (2005) as a generalization of standard finite element analysis. Isogeometric analysis is a new method of computational analysis with the goal of merging design and analysis into one model by using a unified geometric representation. NURBS (Non-Uniform Rational B-Splines) are the most widespread technology in today's CAD modeling tools and therefore well suited as basis functions for analysis. Adopting the isogeometric concept has shown computational advantages over standard finite element analysis in terms of accuracy in many application areas, including solid and structural mechanics.

In this thesis, the isogeometric concept should be applied to the analysis of thin-walled shell structures. For this purpose the rotation-free shell element proposed by Kiendl *et al.* (2009), based on the Kirchhoff-Love shell theory and using NURBS as basis functions should be employed. NURBS-based analysis provides advantages especially for shells, since the structural behavior of a shell is mainly determined by its geometry and therefore a good geometric description is essential. Furthermore, due to the exact geometry description with NURBS, curvatures can be evaluated directly on the surface without rotational degrees of freedom or nodal directors.

The main purpose of this master thesis is to implement the rotation-free Kirchhoff-Love shell element into an object oriented computing environment. The report should provide a review of the isogeometric concept in general and emphasize theory and computational formulation of isogeometric analysis when applied to analyse thin-walled shell structures. The study should also demonstrate how isogeometric analysis compares to standard finite element analysis when solving both geometrically linear and nonlinear problems.

The master thesis should be organized as a research report. It is emphasized that clarity and structure together with precise references are central requirements in writing a scientific report.

Principal advisor: Kjell Magne Mathisen

The master thesis should be handed in at the Department of Structural Engineering within June 11, 2018.

NTNU, January 22, 2018
Kjell Magne Mathisen
Principal Advisor

Abstract

The drawback in today's computational analysis in structural engineering is that design and analysis are not integrated in a unified platform. A majority of the time spent in structural analysis is used to transform a design model into an analysis suitable model. Furthermore, this process can induce a geometry which is based on approximations. Isogeometric analysis (IGA) was developed to remedy this loss of efficiency as it aims to unify the geometrical representation in both design and analysis. NURBS (Non-Uniform Rational B-Splines) are the most widespread mathematical basis in computer-aided design (CAD) as they are able to depict the geometry in an exact manner, which makes NURBS a suitable basis for an IGA.

The purpose of this thesis is to implement the isogeometric rotation-free Kirchhoff-Love thin shell element proposed by Kiendl *et al.* [16], for both geometrically linear and nonlinear analyses within an object-oriented environment. The element is integrated into the open-source software IFEM Elasticity developed by SINTEF Digital, which is part of a research project devoted to merging design and analysis in computational mechanics.

To verify the implementation and the element's performance, a set of both linear and nonlinear benchmark problems have been simulated. The performance has been evaluated through a convergence study for the linear cases, while the nonlinear cases have been compared to results obtained by Lagrange-based finite element analysis (FEA) simulations. The results of the implemented shell element yielded a coherence with the reported results in relevant scientific publications.

Sammendrag

Slik som modelleringsprosessen i konstruksjonsteknikk er lagt opp i dag blir broparten av analysiden brukt til å omgjøre en dataassistert konstruksjonsmodell (DAK-modell) til en modell som det kan bli utført beregningsanalyser på, da det ikke finnes en felles plattform til begge formål. Foruten om tidsforbruket som kreves for å omgjøre en DAK-modell ligger det også en svakhet ved at denne prosessen kan innføre unøyaktigheter. Dette er fordi analysemodellen ikke klarer å representere krumme geometrier eksakt. Isogeometrisk analyse (IGA) er utviklet med den hensikt å integrere design og analyse. NURBS (ikke-uniforme rasjonelle B-splines) er den mest utstrakte matematiske funksjonen brukt for å representere geometri i DAK-verktøy, og er derfor ofte valgt som basisfunksjon i IGA.

Formålet med denne masteravhandlingen er å implementere det isogeometriske rotasjonsfrie Kirchhoff-Love tynnskall-elementet først foreslått av Kiendl *et al.* [16] i et objektorientert programmeringsmiljø, for både lineære og ikkelineære analyser. Elementet er integrert inn i IFEM Elasticity som er et program med åpne kildekode, utviklet av SINTEF Digital som en del av et forskningsprosjekt med den hensikt å fusjonere design og analyse i beregningsmekanikk.

For å verifisere implementasjonen av skallelementet og dens ytelse har et sett med lineære og ikkelineære referanseeksempler blitt analysert. Ytelsen har blitt målt ved et konvergenstudie for de lineære eksemplene, mens de ikkelineære har blitt sammenlignet med resultater fra tidligere publiserte artikler hvor samme eksempel har blitt simulert med den klassiske elementmetoden. Alle resultatene stemte godt overens med tidligere publisert litteratur.

Preface

This master thesis in the subject Computational mechanics is carried out at Department of Structural Engineering at Norwegian University of Science and Technology (NTNU). The thesis is written over a period of 20 weeks in the spring of 2018. This work has been supervised by principal advisor Kjell Magne Mathisen, professor at Department of Structural Engineering (NTNU), and advisor Knut Morten Okstad, research scientist at SINTEF.

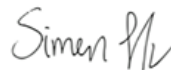
Acknowledgements

We would like to express our gratitude to our advisors for excellent guidance and dedication to our thesis. Their commitment has exceeded our expectations. As principal advisor, Kjell Magne Mathiesen has given us insight in the theory behind IGA, and Knut Morten Okstad has helped us with the challenges we encountered during the implementation.

A sincere thanks to Dr.-Ing. Josef Kiendl for providing us with his MATLAB implementation of the rotation-free Kirchhoff-Love shell element. Lastly, a thanks to Kjetil André Johannessen and Arne Morten Kvarving, research scientists at SINTEF, for helping us with the practical aspects of IFEM Elasticity.



Marit Gaarder Rakvåg



Simen Skogholt Haave

Trondheim, June 11, 2018

Contents

1	Introduction	1
1.1	Motivation	1
1.2	Outline	2
2	B-splines and NURBS	3
2.1	B-splines	4
2.1.1	Knot vectors	4
2.1.2	Basis functions	5
2.1.3	B-spline curves	7
2.1.4	B-spline surfaces	7
2.1.5	B-spline solids	8
2.2	NURBS	8
2.2.1	A geometrical conceptualisation of NURBS	9
2.2.2	The algebraic underpinnings of NURBS	11
2.2.3	Derivatives of NURBS	12
2.2.4	Defining circle segments with NURBS	12
3	NURBS-based isogeometric analysis	15
3.1	Refinements of NURBS	16
3.1.1	Knot insertion	16
3.1.2	Order elevation	18
3.1.3	k -refinement	19
3.2	Analysis spaces	21
3.3	Code architecture	22
3.4	Numerical methods	24

3.4.1	Galerkin's method	25
3.4.2	Gauss quadrature integration	25
4	The NURBS-based rotation-free Kirchhoff-Love shell	27
4.1	Mathematical notation	28
4.2	Differential geometry of surfaces	29
4.3	Structural mechanics of shells	31
4.3.1	Kirchhoff-Love shell element	32
4.3.2	NURBS discretized Kirchhoff-Love shell	33
4.3.3	Discretization	35
5	Implementation of the IGA shell element	37
5.1	Implementation	37
5.1.1	Classes	38
5.1.2	Methods	39
5.2	Implementational Issues	41
5.2.1	Connectivity matrices	42
5.2.2	Edge collapse	43
5.2.3	Loading	44
5.2.4	Boundary conditions	44
5.2.5	Nonlinear implementation	46
6	Verification of the IGA shell element	49
6.1	Benchmark examples with geometric linearity	49
6.1.1	Scordelis-Lo roof	50
6.1.2	Pinched cylinder	59
6.1.3	Pinched hemisphere	61
6.1.4	Pinched hemisphere with cut	64
6.1.5	Clamped hemisphere with periodic loading	66
6.2	Benchmark examples with geometric nonlinearity	70
6.2.1	Cantilever beam	70
6.2.2	Pinched cantilever cylinder	73
6.2.3	Hemispherical shell with cut	77
7	Conclusion	81

<i>CONTENTS</i>	ix
8 Further work	83
Bibliography	85
A Constructing a B-spline curve	89
B Multimesh extrapolation	93
C ABAQUS elements	95

List of Figures

2.1	Quadratic basis functions	4
2.2	Basis functions for an arbitrary knot vector	6
2.3	B-spline surface	8
2.4	The projective transformation of a B-spline	10
2.5	A NURBS circle segment.	13
3.1	Knot insertion example	18
3.2	Order elevation example	19
3.3	Illustration of how knot insertion and order elevation are not commutative	20
3.4	Illustration of the three different spaces for a NURBS surface	22
3.5	Flow chart of the code architecture	23
4.1	Cartesian coordinate system versus curvilinear coordinate system	30
5.1	Inheritance diagram of the integrand classes in IFEM Elasticity	38
5.2	Method diagram of the implemented methods	40
5.3	Bi-quadratic NURBS with two elements	42
5.4	Illustration of an edge collapse	44
5.5	Symmetry boundary condition	46
6.1	Scordelis-Lo roof problem setup	51
6.2	Scordelis-Lo roof: Vertical displacement	51
6.3	Reduced model Scordelis-Lo roof: Convergence plot of the vertical displacement	53
6.4	Full model Scordelis-Lo roof: Convergence plot of the vertical displacement	53
6.5	Reduced model Scordelis-Lo roof: Convergence plot of the internal energy	54
6.6	Full model Scordelis-Lo roof: Convergence plot of the internal energy	54

6.7	Reduced model Scordelis-Lo roof: Convergence plot of the von Mises stress . . .	55
6.8	Full model Scordelis-Lo roof: Convergence plot of the von Mises stress	55
6.9	Scordelis-Lo roof with a boundary refined mesh	56
6.10	Boundary refined model Scordelis-Lo roof: Convergence plot of the vertical displacement	57
6.11	Scordelis-Lo roof: Membrane stresses plotted with two visualization points . . .	58
6.12	Scordelis-Lo roof: Membrane stresses plotted with four visualization points . . .	58
6.13	Scordelis-Lo roof: Bending moment plotted with two visualization points	59
6.14	Pinched cylinder problem setup	60
6.15	Pinched cylinder: Vertical displacement	60
6.17	Pinched cylinder: Convergence plot of the displacement w.r.t. elements per edge	62
6.16	Pinched cylinder: Convergence plot of the displacement w.r.t. unknowns	62
6.18	Pinched hemisphere problem setup	63
6.19	Pinched hemisphere: Deformed geometry	63
6.20	Pinched hemisphere: Convergence plot of the displacement	64
6.21	Pinched hemisphere with cut problem setup	65
6.22	Pinched hemisphere with cut: Deformed geometry	65
6.23	Pinched hemisphere with cut: Convergence plot of the displacement	66
6.24	Clamped hemisphere problem setup	67
6.25	Clamped hemisphere: Deformed geometry	67
6.26	Clamped hemisphere: Convergence plot of the displacement	68
6.27	Clamped hemisphere: Convergence plot of the normalized displacement	68
6.28	Clamped hemisphere: Convergence plot of the internal energy	69
6.29	Cantilever beam: Load step versus displacement	71
6.30	Cantilever beam: Deformed geometry	71
6.31	Cantilever beam: Convergence results from the NR iterations	72
6.32	Pinched cantilever cylinder problem setup	74
6.33	Pinched cantilever cylinder: Deformed geometry	74
6.34	Pinched cantilever cylinder: Membrane stresses	75
6.35	Pinched cantilever cylinder: Bending moments	75
6.36	Pinched cantilever cylinder: von Mises stresses	76
6.37	Pinched cantilever cylinder: Normalized displacement versus the reaction force .	76
6.38	Hemispherical shell with cut: Load step versus displacement	78

6.39 Hemispherical shell with cut: Deformed geometry	79
A.1 Basis functions for a given knot vector	90
A.2 B-spline curve	91

List of Tables

3.1	Differences between IGA and FEA	16
6.1	Cantilever beam: NR iteration results	72
6.2	Pinched cantilever cylinder: NR iteration results	77
6.3	Hemispherical shell with cut: NR iteration results	79
A.1	Coordinates for the control points	91

Abbreviations

All abbreviations used in this thesis are listed here.

BC(s)	Boundary condition(s)
BVP	Boundary value problem
CAD	Computer-aided design
DOF(s)	Degree(s) of freedom
FEA	Finite element analysis
IEN	Connectivity array (Internal entry number)
IGA	Isogeometric analysis
NR	Newton-Raphson
NURBS	Non-uniform rational B-splines

Chapter 1

Introduction

The purpose of this master thesis is to implement an isogeometric thin shell element within an object-oriented environment. The specific element that is emphasized in this thesis is the rotation-free thin shell element proposed by Kiendl *et al.* [16]. The element is based on the Kirchhoff-Love shell theory and referred to as the Kirchhoff-Love shell element or the IGA shell element in this thesis. The thesis also includes a review of the isogeometric analysis (IGA) in general and in comparison with the classical finite element analysis (FEA), together with theory and computational formulation of IGA when applied to thin shell elements. This is done for both geometrically linear and nonlinear problems.

This thesis is organized as a research report and it is structured in such a way that readers do not need any prerequisite knowledge about IGA. By the term FEA it is in this thesis referred to the finite element analysis where geometry is described by Lagrange-polynomials and either Lagrange or Hermite polynomials are used as shape functions. By IGA it is in this thesis referred to an analysis where NURBS are used to describe the geometry and used as shape functions.

1.1 Motivation

IGA was developed by Hughes *et al.* [12] as an approach of integrating standard FEA with computer-aided design (CAD) in order to make computational analysis more efficient by applying a unified geometric description. CAD is in most cases based on non-uniform rational basis splines (NURBS) to define the geometry of the model, while FEA applies Lagrange polynomials. This change in geometrical description causes a decrease in efficiency as creating a geometry applicable for analysis requires roughly 80% of overall analysis time according to Cottrell *et al.*

[8]. This is not a problem when IGA is applied as it unifies the geometrical description.

NURBS has the ability to describe the geometry in an exact manner, while Lagrange polynomials are only able to represent the geometry approximately. This results in an advantage in terms of accuracy when IGA is applied compared to FEA. The use of NURBS as a basis functions is especially advantageous when applied to shell elements as shell's behavior is determined primarily by its geometry, therefore an exact geometrical description is desirable. The exact description of shells geometry also leads to the fact that the curvatures of the shell can be evaluated directly on the surface without the need of rotational degrees of freedom, hence the rotation-free shell element.

The Kirchhoff-Love shell element is implemented in the open-source software IFEM Elasticity [24], which is a computer program used to solve differential equations of elasticity problems. IFEM is part of the ICADA (Integrated computer-aided design and analysis) project and it is developed by SINTEF Digital in cooperation with NTNU (Norwegian University of Science and Technology). ICADA is a research project aiming to integrate design and analysis.

1.2 Outline

The first chapters of this master thesis is a study of the isogeometric concept in general. Then theory, computational formulation and implementational issues of thin shell structures is discussed. Finally, this implementation is used to analyze benchmark examples with both geometrically linear and nonlinear problems. Following is a brief description of each chapter in the thesis:

- Chapter 2 reviews the theory of B-splines and NURBS.
- Chapter 3 compares IGA with FEA and briefly explains the advantages of IGA.
- Chapter 4 reviews the structural mechanics of shell elements in general and the Kirchhoff-Love shell element formulation.
- Chapter 5 discusses the implementational aspects of this thesis, both in terms of the object-oriented environment and the implementational issues that arise when dealing with IGA.
- Chapter 6 verifies the implementation of the Kirchhoff-Love shell element with benchmark examples.
- Chapter 7 presents the thesis' conclusion.
- Chapter 8 presents suggestions of further work.

Chapter 2

B-splines and NURBS

This chapter presents the theory of B-splines and NURBS which is the most common basis functions used in IGA. The use of NURBS is widespread in the CAD domain, while its presence is lacking from the structural analysis scene. For structural analyses of today, the use of Lagrange polynomials is prevalent although they may not recreate geometries in an exact manner. NURBS and B-splines are mathematical functions which differ from the Lagrange polynomials as their properties and formulations are more complex and abstract. Thus, this chapter will present relevant mathematical concepts and implications relevant for B-splines and NURBS.

A NURBS geometry is a linear combination of shape functions, referred to as basis functions, control points and control point weights. See Figure 2.1 for an example of a set of basis functions constructed from a knot vector. The domain in the figure represents what is called a patch. NURBS and B-splines are local to patches and not to elements, which is of great importance as it is one of the most central differences compared to the basis functions used in standard FEA. In this thesis, only single patch domains will be examined.

NURBS are a generalized formulation of its predecessor, B-splines, and share multiple properties. For this reason, the B-splines will firstly be reviewed before embarking onto the NURBS theory.

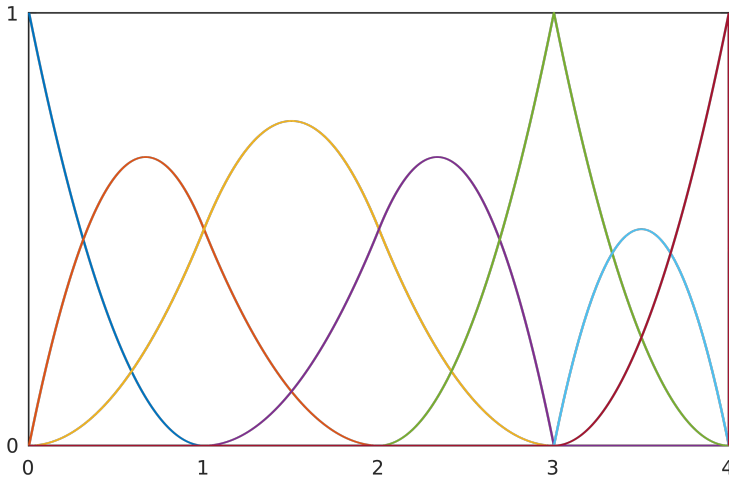


Figure 2.1: Quadratic basis functions generated from the knot vector $\Xi = \{0, 0, 0, 1, 2, 3, 3, 4, 4, 4\}$

2.1 B-splines

The B-splines are non-rational mathematical functions most often used to describes geometries. A B-spline is constructed by a knot vector, a polynomial degree and a set of control points, denoted Ξ , p and \mathbf{B} , respectively. The knot vector and the polynomial degree are used to generate the basis functions, and the control points determine the linear combinations of the basis functions to represent the geometry. The control points use different terminology depending if the geometry is a curve, surface or solid, and are referred to as control points, control net or control lattice, respectively.

2.1.1 Knot vectors

The knot vector is defined as

$$\Xi = \{\xi_1, \xi_2, \dots, \xi_{n+p+1}\}, \quad (2.1)$$

where $\xi_i \in \mathbb{R}$ and every entry is equal or greater than the previous, $\xi_i \leq \xi_{i+1}$. The subscript i is the knot index, n is the number of basis functions used to construct the B-spline, and p is the spline's polynomial degree. Here, the polynomial degree is defined such that $p = 1$ yields a linear polynomial. The knot vector partitions the patch into $s - 1$ intervals, where s is equal to the number of unique entries in the knot vector, which is referred to as knot spans. Knot spans are the IGA

equivalent to elements in standard FEA. In this thesis, the terms knot span and element are to be regarded as interchangeable.

The knot vector may contain non-unique entries, thus having coinciding knots for a given position in the patch. The number of coinciding knots for a given position is referred to as the knot's multiplicity, denoted k . This property has important consequences for the geometry's behavior and continuity as it directly alters the basis functions. If the first and last knot in the knot vector has a multiplicity of $k = p + 1$, the knot vector is said to be open, and if there are any other multiplicities present, the spline is non-uniform.

2.1.2 Basis functions

The basis functions, denoted \mathbf{N} , can be derived from the knot vector and the polynomial degree using the Cox-de Boor recursion formula

$$N_{i,0}(\xi) = \begin{cases} 1 & \xi_i \leq \xi \leq \xi_{i+1}, \\ 0 & \text{otherwise,} \end{cases} \quad (2.2)$$

$$N_{i,p}(\xi) = \frac{\xi - \xi_i}{\xi_{i+p} - \xi_i} N_{i,p-1}(\xi) + \frac{\xi_{i+p+1} - \xi}{\xi_{i+p+1} - \xi_{i+1}} N_{i+1,p-1}(\xi), \quad (2.3)$$

where $i = 1, \dots, n$. The basis functions are defined over the whole patch, and their values will vary according to their polynomial degree. Hence, $N_{i,0}(\xi)$ will have constant values, $N_{i,1}(\xi)$ will have linearly varying values, and so forth. Figure 2.2 illustrates this recursive process. The basis functions will have a positive value in the interval $[\xi_i, \xi_{i+p+1}]$ and be equal to zero elsewhere. This constitutes what is referred to as compact support. A high polynomial degree may seem to require an increased bandwidth in a numerical method, but this is an erroneous conclusion. The fact is that every function will share support with exactly $2p + 1$ other functions, which is the case for both B-splines and standard Lagrangian basis [8].

Unlike the Lagrange polynomials, the basis functions of B-splines will always be nonnegative and have a maximum value of 1. Another property of the basis functions is that they will always form a partition of unity over the entire patch, i. e.

$$\sum_{i=1}^n N_{i,p}(\xi) = 1. \quad (2.4)$$

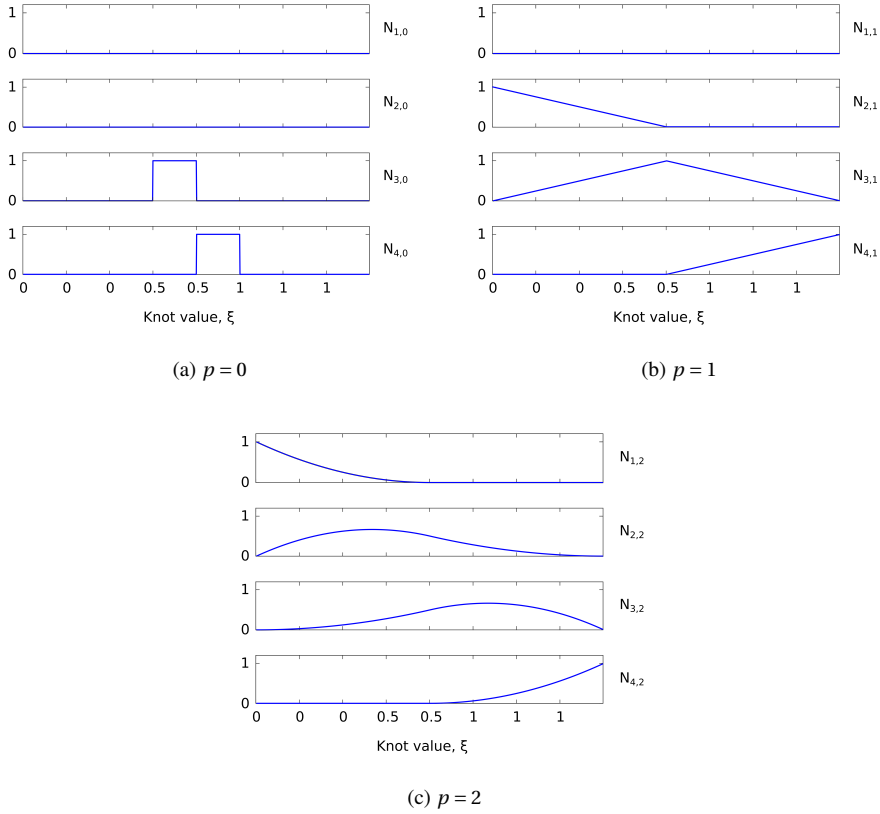


Figure 2.2: Basis functions for an arbitrary knot vector, for multiple polynomial degrees.

The B-spline's cardinal advantage over Lagrange polynomials is its ability to depict an arbitrary geometry with a greater precision, and while doing so, the geometries will have a higher order of continuity [8]. A B-spline function has C^{p-k_i} continuous derivatives across knot ξ_i , whereas C^0 -continuity is ubiquitous in FEA.

In this thesis, open knot vectors will be used unless the opposite is stated. As mentioned, open knot vectors are characterized by a multiplicity of $k = p + 1$ at the ends, hence resulting in a C^{-1} -continuity. This continuity marks the patch's boundary, and in the case of multiple patches a C^0 -continuity is achieved between the patches. A C^1 -continuity can be obtained across patches by adding constraint to the slopes' angles at the interface, but this is however not in the scope of this thesis [23].

2.1.3 B-spline curves

A B-spline curve, denoted \mathbf{C} , is a linear combination of the basis functions and the control points

$$\mathbf{C}(\xi) = \sum_{i=1}^n \mathbf{N}_{i,p}(\xi) \mathbf{B}_i. \quad (2.5)$$

The B-spline curve adheres to a strong convex hull property, which implies that the curve can only lie inside the convex hull of the control points. This property is a consequence of the formerly mentioned compact support, nonnegativity and partition of unity properties of the B-spline curve [10]. The strong convex hull property is desirable, because it assures that the curve does not arbitrarily shoot off, and it prevents unnecessary cancellations in arbitrary affine combinations. This gives a relatively higher numerical precision. In Appendix A an example is provided in order to demonstrate how to derive a B-spline curve from a knot vector and a set of control points.

2.1.4 B-spline surfaces

To construct a B-spline surface, denoted \mathbf{S} , two sets of knot vectors, namely Ξ and \mathcal{H} , and a control net \mathbf{B} are required. Here $\mathcal{H} = \{\eta_1, \eta_2, \dots, \eta_{m+q+1}\}$, η_j is the j^{th} knot index, m is the number of basis functions, and q is the polynomial degree. The directions of ξ and η must be orthogonal to each other. The two sets of basis functions are derived from the knot vectors, and standard notation is to let $N_{i,p}$ be derived from Ξ , and $M_{j,q}$ from \mathcal{H} . The B-spline surface is a tensor product of both η and ξ , hence $\mathbf{S} = \mathbf{S}(\xi, \eta) \in \mathbb{R}^2$, and is defined as

$$\mathbf{S}(\xi, \eta) = \begin{bmatrix} x_{i,j} \\ y_{i,j} \end{bmatrix} (\xi, \eta) = \sum_{i=1}^n \sum_{j=1}^m N_{i,p}(\xi) M_{j,q}(\eta) \mathbf{B}_{i,j}. \quad (2.6)$$

Here, both N and M will satisfy Equation (2.4), thus their product will also form a partition of unity. Figure 2.3 is an illustration of a B-spline surface constructed with a control net and a set of basis functions. The support of the bivariate basis function $N_{i,p}(\xi)M_{j,q}(\eta)$ will be exactly $[\xi_i, \xi_{i+p+1}] \times [\eta_j, \eta_{j+q+1}]$, which is a 2D extension of the compact support property mentioned in Section 2.1.2.

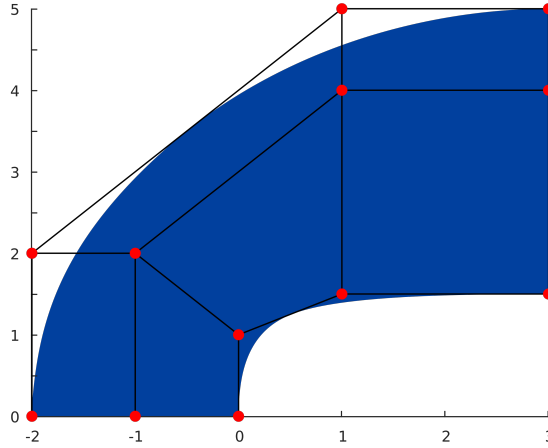


Figure 2.3: Illustration of a B-spline surface generated from the knot vectors $\Xi = \{0, 0, 0, 0.5, 1, 1, 1\}$, $\mathcal{H} = \{0, 0, 0, 1, 1, 1\}$ and with the control points as marked in the figure.

2.1.5 B-spline solids

The extension from surface to solid follows the same principles as the extension from curves to surfaces. To be able to construct a B-spline solid, one additional set of basis functions $L_{k,r}(\zeta)$ and a control lattice \mathbf{B} is required. The B-spline solid's properties are trivariate formulations of those which apply to the surfaces, thus the mathematical formulation of a B-spline solid is expressed as

$$\mathbf{S}(\xi, \eta, \zeta) = \begin{bmatrix} x_{i,j,k} \\ y_{i,j,k} \\ z_{i,j,k} \end{bmatrix} (\xi, \eta, \zeta) = \sum_{i=1}^n \sum_{j=1}^m \sum_{k=1}^l N_{i,p}(\xi) M_{j,q}(\eta) L_{k,r}(\zeta) \mathbf{B}_{i,j,k}. \quad (2.7)$$

2.2 NURBS

B-splines are able to depict arbitrary geometries with a higher precision than Lagrange polynomials, but B-splines are not able to do so in an exact manner. This is due to the fact that B-splines are non-rational and geometries derived from B-splines will also be non-rational. NURBS (non-uniform rational B-splines) have evolved from B-splines and as they are rational, they are able to depict an arbitrary geometry in an exact manner [20]. In order to conceptualize NURBS and differentiate them from B-splines, it is convenient to start by examining the geometrical differences prior to deriving the algebraic expressions.

2.2.1 A geometrical conceptualisation of NURBS

A B-spline entity in \mathbb{R}^{d+1} , where d is the number of spatial directions, can be expressed as a NURBS entity in \mathbb{R}^d by applying projective transformation to the B-spline entity. The projective transformation of the curves and control points are illustrated in Figure 2.4, where the B-spline is referred to as the projective geometry, with superscript w . This process projects a ray from the origin, through the plane $z = 1$, and onto each of the points of the geometry. This leaves an outline of the B-spline geometry on the plane $z = 1$, and the elevation of the original geometry is retained with use of a weighting function. The weights are, in geometric terms, the height of the projective control points of the B-spline curve, and will be equal to or greater than zero in all cases relevant here. The juxtaposition of how control points are defined in a NURBS setting with Cartesian coordinates and an associated weight often causes a misunderstanding, it is not correct to conceptualize the weight as a component of the control points.

To attain the weighting functions for the NURBS entity, it is necessary to first define its control points by using

$$(\mathbf{B}_i)_j = (\mathbf{B}_i^w)_j / w_i, \quad (2.8)$$

where

$$w_i = (\mathbf{B}_i^w)_{d+1}, \quad (2.9)$$

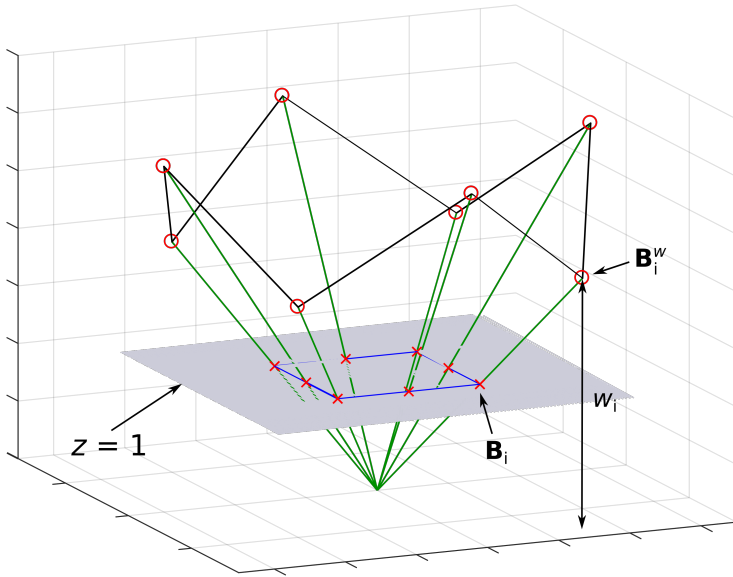
$(\mathbf{B}_i)_j$ is the j^{th} component of \mathbf{B}_i , w_i is the i^{th} weight, and $j = 1, 2, \dots, d$. In the example presented in Figure 2.4, it is evident that a projective transformation is equivalent to dividing $(\mathbf{B}_i^w)_j$ by w_i , as the weights are set equal to the z -component of the projective control points. The same alteration has to be applied to each point along the curve to create the NURBS curve \mathbf{C} . This is achieved by defining a weighting function, denoted W , defined as

$$W(\xi) = \sum_{i=1}^n N_{i,p}(\xi) w_i, \quad (2.10)$$

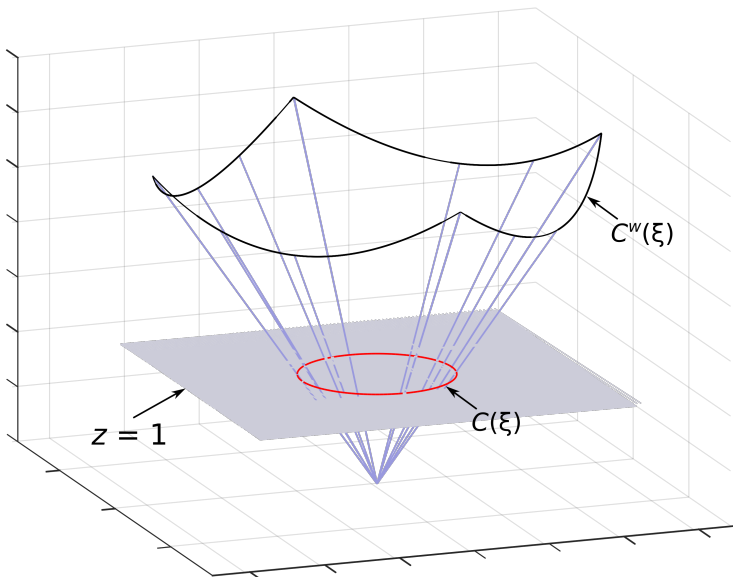
The weighting function together with the B-spline curve \mathbf{C}^w will unambiguously define the NURBS curve as

$$(\mathbf{C}(\xi))_j = \frac{(\mathbf{C}(\xi)^w)_j}{W(\xi)}, \quad \text{where } j = 1, 2, \dots, d. \quad (2.11)$$

Here, both the function in the numerator and denominator is a piecewise polynomial function which yields a piecewise rational function. This may lead to an ambiguous polynomial degree; hence it is conventional to report the polynomial degree of the B-spline it is based on.



(a) Projective transformation of the control points.



(b) Projective transformation of the B-spline curve.

Figure 2.4: A projective transformation of a piecewise quadratic B-spline in \mathbb{R}^3 to a NURBS entity in \mathbb{R}^2 .

The projective curve may contain points of lower continuity, the Figure 2.4b has four points with a C^0 -continuity, which cannot be deduced by only examining the red circle. Thus, it is important to be aware that the continuity restrictions come from the projective geometry and not the NURBS object itself. An interesting point to make regarding Figure 2.4 is that the NURBS curve would not change even if all the control points of the B-spline were multiplied with a constant. The resulting projective curve would simply follow the projective ray. From the figure it may also be evident that a B-spline may be interpreted as a special case of a NURBS object where all weights are equal to one.

2.2.2 The algebraic underpinnings of NURBS

The geometric conceptualization of the NURBS entity makes it easy to differentiate between the two concepts B-splines and NURBS as the differences are easily pointed out with Figure 2.4, giving some intuition of how they are constructed. To alter and manipulate the NURBS entities, an explicit algebraic understanding needs to be established. The already derived equations in Section 2.1 have to be altered as they now must account for the weighting function. The NURBS basis functions, R_i^p , will be derived using the B-spline basis functions and the weighting function. For a NURBS curve, the basis is defined as

$$R_i^p(\xi) = \frac{N_{i,p}(\xi) w_i}{W(\xi)} = \frac{N_{i,p}(\xi) w_i}{\sum_{\alpha=1}^n N_{\alpha,p}(\xi) w_{\alpha}}, \quad (2.12)$$

where $N_{i,p}$ is the B-spline basis functions and w_i is the i^{th} weight. Extending a NURBS entity into additional spatial dimensions requires additional sets of basis functions. The expression can with ease be expanded into surfaces and solids. The basis functions for surfaces and solids are defined as

$$R_{i,j}^{p,q}(\xi, \eta) = \frac{N_{i,p}(\xi) M_{j,q}(\eta) w_{i,j}}{\sum_{\alpha=1}^n \sum_{\beta=1}^m N_{\alpha,p}(\xi) M_{\beta,q}(\eta) w_{\alpha,\beta}}, \quad R \in \mathbb{R}^2, \quad (2.13)$$

$$R_{i,j,k}^{p,q,r}(\xi, \eta, \zeta) = \frac{N_{i,p}(\xi) M_{j,q}(\eta) L_{k,r}(\zeta) w_{i,j,k}}{\sum_{\alpha=1}^n \sum_{\beta=1}^m \sum_{\gamma=1}^l N_{\alpha,p}(\xi) M_{\beta,q}(\eta) L_{\gamma,r}(\zeta) w_{\alpha,\beta,\gamma}}, \quad R \in \mathbb{R}^3. \quad (2.14)$$

With the NURBS basis functions \mathbf{R} the procedure employed to obtain the function for the NURBS entity is analogous with that of the B-spline. The function which unambiguously defines the NURBS curve is as follows

$$\mathbf{C}(\xi) = \sum_{i=1}^n R_i^p(\xi) \mathbf{B}_i, \quad (2.15)$$

where \mathbf{B} is the set of control points. For the surface and solid expansion of this equation, see Equations (2.6) and (2.7) as it follows the same strategy. It quickly becomes complex to visually present the differences between NURBS and the B-splines when working with surfaces or solids.

All the aforementioned properties about the basis function used to construct B-spline entities applies also to the NURBS basis functions. The B-spline basis functions constitute a partition of unity, are pointwise nonnegative, and have compact support. As these attributes remain intact after the projective transformation, they assure that the NURBS object adheres to a strong convex hull rule.

2.2.3 Derivatives of NURBS

In an isogeometric formulation of a Kirchhoff-Love shell element, the derivatives of the geometry functions are of great importance, and for this reason a thorough review of the cumbersome procedure to generate its derivatives will be performed. The derivatives of the NURBS basis functions $\frac{d}{d\xi}\mathbf{R}$ are dependent on two quantities: the basis functions \mathbf{N} and the weighting function W . Note that the derivative of the rational function is defined in terms of its non-rational substructure. An important feature of the basis functions which is also the case for the B-spline basis functions, is that the basis functions will have $p - k$ continuous derivatives over a knot with a multiplicity of k . Hence it is beneficial to derive an expression for the t^{th} derivative. First, the quotient rule is applied to Equation (2.12) to attain the first derivative of the basis function

$$\frac{d}{d\xi}R_i^p(\xi) = w_i \frac{W(\xi) \frac{d}{d\xi}N_{i,p}(\xi) - N_{i,p}(\xi) \frac{d}{d\xi}W(\xi)}{(W(\xi))^2}. \quad (2.16)$$

An efficient algorithm to calculate the higher order derivatives is

$$\frac{d^t}{d\xi^t}R_i^p(\xi) = \frac{w_i \frac{d^t}{d\xi^t}N_{i,p}(\xi) - \sum_{j=1}^t \binom{t}{j} \frac{d^t}{d\xi^t}W(\xi) \frac{d^{t-j}}{d\xi^{t-j}}R_i^p(\xi)}{W(\xi)}. \quad (2.17)$$

2.2.4 Defining circle segments with NURBS

In structural engineering, circles and arcs are frequently used, and full circles will be used as benchmark examples in this thesis. Therefore, it is appropriate to address the potential challenge they hold. This difficulty is easiest to present by first examining a 60° circle segment. In Figure 2.5, a NURBS formulated circle segment with its control points and corresponding weights are presented. The outer control points are located on the curve, but the remaining control point, \mathbf{B}_2 ,

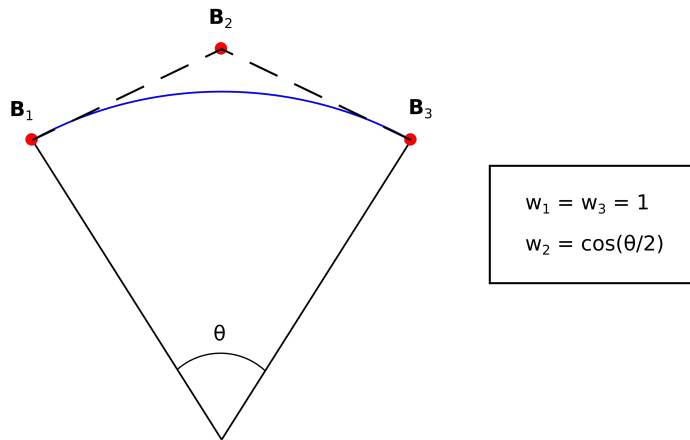


Figure 2.5: A NURBS circle segment. \mathbf{B}_2 lies where control point \mathbf{B}_1 and \mathbf{B}_3 tangent lines cross. When the tangent lines no longer cross each other, which is the case if $\theta \geq 180^\circ$, this method will not produce the desired geometry.

has to be located where the two tangent lines from \mathbf{B}_1 and \mathbf{B}_3 cross. This manner of constructing a circle segment works well for segments $< 180^\circ$, but this method breaks as soon as the segment is larger than 180° . By examining Figure 2.5 it becomes evident that the reason for this is that the two tangent lines will never cross paths. For this reason, it is common practice to either use multiple patches or introduce multiplicities. The former will use multiple patches, preferably a new patch for every 90° , to model the arc. This has the disadvantage of introducing redundant control points where each patch meet. The latter approach will introduce a multiplicity of $k = p$ for every 90° , which will force the corresponding control point to be located on the curve (C^0 -continuity). This will also in a minor way change the problem as this procedure lowers the continuity, which in turn gives a slightly altered geometry. Note that in case of a full circle, the first and the last control points will have the exact same coordinates. A third way which will fully circumvent this obstacle is to use symmetry boundary conditions. Symmetry can be used on an arc with $\theta = 90^\circ$ to construct a full circle. This approach has only the drawback of not being applicable to all geometries.

Chapter 3

NURBS-based isogeometric analysis

In order to understand what differentiates IGA from FEA, this chapter will present the similarities and the differences of the two analysis methods. The main difference between the two methods are that IGA (in the form that it is described and employed in this thesis) applies NURBS as basis functions, while FEA operates with Lagrange or Hermite polynomials. This change of basis has several consequences, like computational efficiency and accuracy of the solution. Table 3.1 presents some of the essential differences between IGA and FEA, as proposed by Cottrell *et al.* [8]. Most of the entries in this table are directly linked to the associated basis functions.

The essence of IGA is that it employs the same basis functions to describe the geometry of the problem in both the design stage and in the analysis stage of the process, where in this thesis NURBS are applied. This results in an analysis where the exact geometry is reviewed. Whereas in FEA, the geometry description used in design are meshed to finite elements with Lagrange or Hermite polynomials as basis and this meshing process may not replicate the geometry exactly, and in addition this process is computationally expensive.

The differences that are relevant for this thesis is discussed further in this chapter, such as refinements of NURBS and analysis spaces, as well as the cases that are similar in both methods, like code architecture for establishing the system equation and numerical methods.

Table 3.1: Differences between IGA and FEA, as proposed by Cottrell *et al.* [8].

IGA	FEA
Exact geometry	Approximate geometry
Control points	Nodal points
Control variables	Nodal variables
Basis does not interpolate control points and variables	Basis interpolates nodal points and variables
NURBS basis	Polynomial basis
High, easily controlled continuity	C^0 -continuity, always fixed
hpk -refinement space	hp -refinement space
Pointwise positive basis	Basis not necessarily positive
Convex hull property	No convex hull property
Variation diminishing in the presence of discontinuous data	Oscillatory in the presence of discontinuous data

3.1 Refinements of NURBS

An advantage of using NURBS as basis functions is that refinements in terms of number of elements or polynomial degree does not change the basis' geometry or parametrization. The refinements of NURBS resemble the standard FEA refinements, namely h - and p -refinements, in the way that they both control the element size and the polynomial degree of the basis functions, but they vary in the way that refinements of NURBS can also control the continuity of the basis function. There are three different methods of refinements in IGA: knot insertion, order elevation and k -refinements, which are explained in more detail in this section. All examples presented in this section are refinements of NURBS curves, but the process of refining NURBS surfaces or solids is performed analogously.

3.1.1 Knot insertion

Knot insertion is the IGA equivalent to classical h -refinement in FEA. By adding a new entry to the knot vector, the basis function gains a new knot, and this can be done without changing the geometry or the parametrization of the curve. The inserted knot may be either a new knot value which divide an existing knot span into two (creating a new element), or be a repetition of a knot

value already present in the knot vector to increase the multiplicity of the knot and decrease the continuity of the basis at this point. The latter will not be further discussed in this thesis because in general a high continuity is desired.

Given the knot vector $\Xi = \{\xi_1, \xi_2, \dots, \xi_{n+p+1}\}$, then the refined knot vector, after inserting the new knot, becomes $\bar{\Xi} = \{\bar{\xi}_1 = \xi_1, \bar{\xi}_2, \dots, \bar{\xi}_{m+p+1} = \xi_{n+p+1}\}$, where $m = n + 1$ and Ξ is a subset of $\bar{\Xi}$. The new knot is defined as $\bar{\xi} \in \langle \xi_k, \xi_{k+1} \rangle$, where $\xi_k \neq \xi_{k+1}$. The new refined knot vector has now m basis functions, that must be formed again as in Equations 2.2 and 2.3. To preserve the original geometry and continuity of the curve, the new m control points, denoted $\bar{\mathbf{B}}$, can be calculated from linear combinations of the n original control points \mathbf{B} [20],

$$\bar{\mathbf{B}}_i = \begin{cases} \mathbf{B}_1 & i = 1, \\ \alpha_i \mathbf{B}_i + (1 - \alpha_i) \mathbf{B}_{i-1} & 1 < i < m, \\ \mathbf{B}_n & i = m, \end{cases} \quad (3.1)$$

where

$$\alpha_i = \begin{cases} 1 & i \leq k - p, \\ \frac{\bar{\xi} - \xi_i}{\bar{\xi}_{i+p} - \xi_i} & k - p + 1 \leq i \leq k, \\ 0 & i \geq k + 1. \end{cases} \quad (3.2)$$

A simple example of knot insertion are presented in Figure 3.1. The left column of the figure presents a quadratic B-spline curve with one element and knot vector $\Xi = \{0, 0, 0, 1, 1, 1\}$, with its associated basis functions plotted underneath. The right column of the figure presents the refined curve with two elements and knot vector $\bar{\Xi} = \{0, 0, 0, 0.5, 1, 1, 1\}$. Here, the new knot $\bar{\xi} = 0.5$ is inserted to make a refined mesh of two elements, and the new control points are redefined, as in Equations 3.1 and 3.2, to preserve the geometry of the curve.

As seen in this section, there are similarities with h -refinements in FEA and knot insertion, like refining the mesh with more elements from the ones that already exist. It varies from FEA in how many new basis functions that are established, and in the continuity across the new element, where NURBS have C^{p-k} -continuity while Lagrange polynomials has C^0 -continuity across all elements. Therefore, a knot value must be inserted p times to replicate h -refinement.

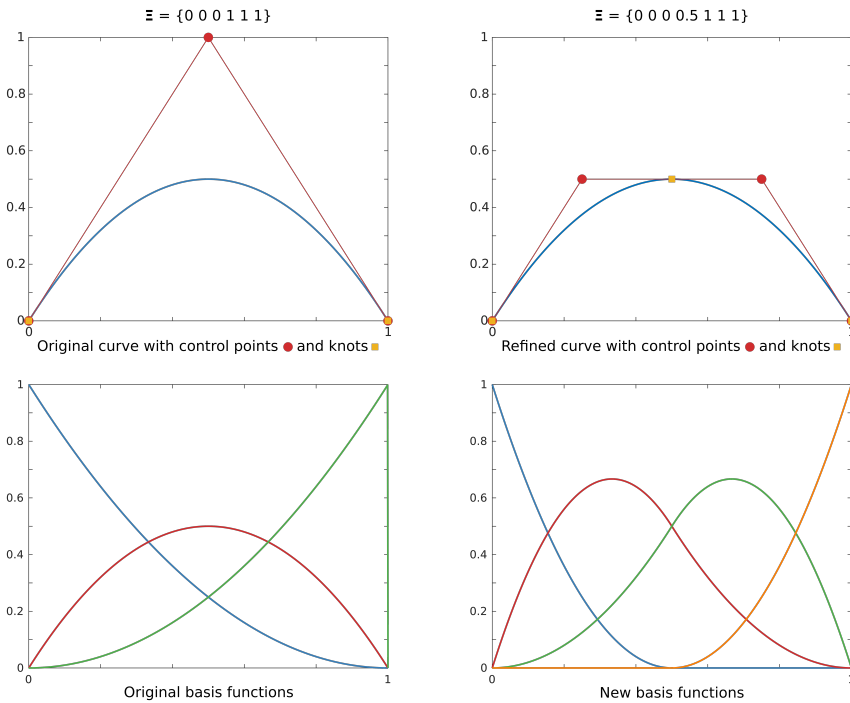


Figure 3.1: An example of knot insertion, with the B-spline curve and its basis functions prior to knot insertion (left column) and after knot insertion (right column).

3.1.2 Order elevation

The method of order elevation improves the basis functions by raising the polynomial degree. This is done by increasing the multiplicity of each knot value. A NURBS basis have $p - k$ continuous derivatives over the elements, which means that to raise the polynomial degree p , the multiplicity k has to be raised likewise. This means that the knots must be repeated until the desired polynomial degree is reached. When performing an order elevation, the number of control points and basis functions are increased, while the number of elements remains the same. Just like in knot insertion, neither the parametrization nor the geometry is changed after order elevation.

A simple example of order elevation are presented in Figure 3.2. The left column in the figure shows the same curve as in Figure 3.1. The curve in the right column of the figure are the order elevated curve with knot vector, $\bar{\Xi} = \{0, 0, 0, 0, 1, 1, 1, 1\}$. The two intermediate knots are added to the knot vector to raise the polynomial degree from $p = 2$ to $p = 3$. The figure shows the new basis functions, and the redefined control points.

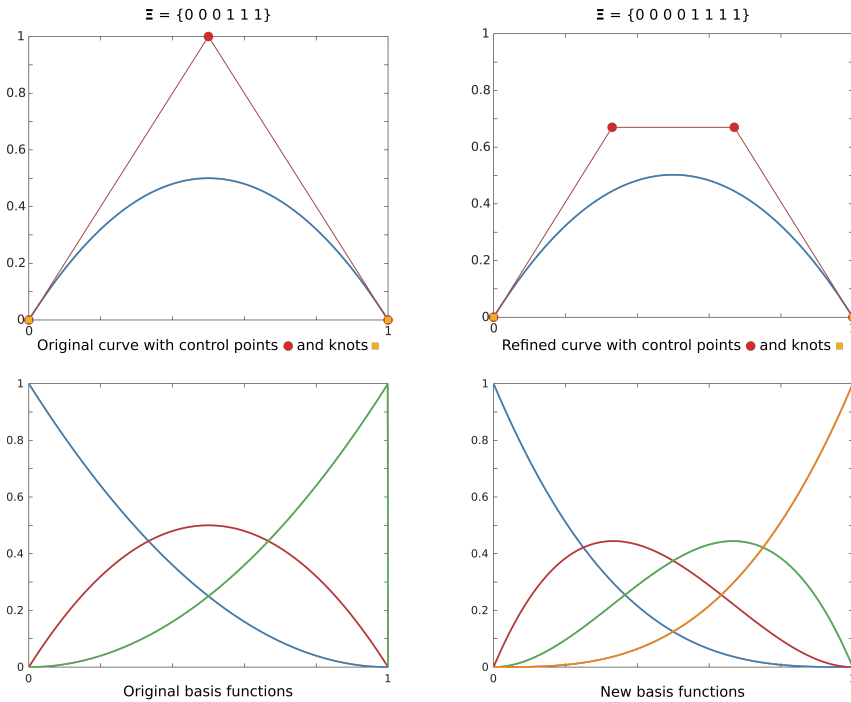


Figure 3.2: An example of order elevation, with the B-spline curve and its basis function prior to order elevation (to the left) and after order elevation (to the right).

Order elevation in IGA is similar to p -refinement in FEA, as both methods raise the polynomial degree of the associated basis. Where FEA increases the polynomial degree by inserting new nodes within the element, order elevation raises the polynomial degree over the whole patch. It differs in the way that FEA has C^0 -continuity over the basis, while in order elevation the continuity C^{p-k} remains the same as before the order elevation, as both p and k are raised by one.

3.1.3 k -refinement

k -refinement in IGA is based on a combination of both knot insertion and order elevation, and FEA has no analogous method. In general, k -refinement is like order elevation, but instead of increasing the multiplicity of all the knot values, only the first and last knot value has its multiplicity increased. This means that the continuity across the knot values are increased to C^{p-1} , instead of remaining the same, as in order elevation.

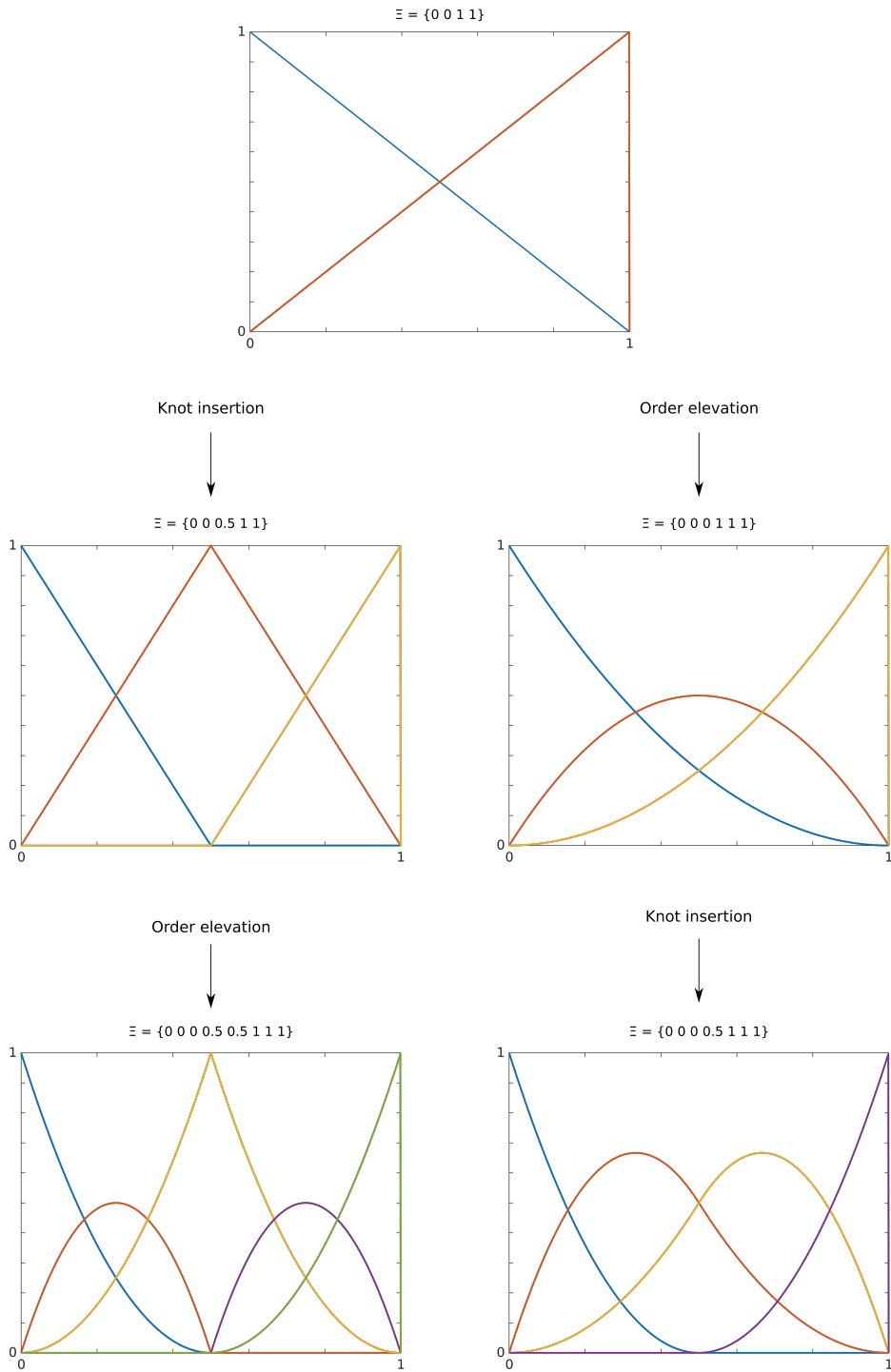


Figure 3.3: Illustration of how knot insertion and order elevation are not commutative. The right side of the figure is a demonstration of k -refinement.

The process of k -refinement is to start with a one-element curve, and then order elevate until the desired polynomial degree is reached, and finally do knot insertion to reach the desired number of elements. If the process had been reversed by starting with a knot insertion, and then perform order elevation, the continuity of the intermediate knot values would maintain the same. This is due to knot insertion and order elevation is a non-commutative processes, which is shown in Figure 3.3. The right side of the figure depicts a k -refinement, where there are no C^0 -continuities across the patch, while the left side, which is a knot insertion followed by an order elevation, has a C^0 -continuity across the patch.

k -refinement has several advantages compared to order elevation. For example, the fact that k -refinements has a higher continuity and that it has fewer number of basis functions, which results in higher efficiency and lower computational costs.

3.2 Analysis spaces

NURBS employs three spaces when used as a basis in IGA. These three spaces are: parametric space, physical space and parent space. Classical FEA on the other hand uses only two analysis spaces, namely physical space and parent space. Thus, the parametric space is unique for IGA. Figure 3.4 illustrates the three spaces, and the mapping between them for an arbitrary NURBS surface. The formulations used in this section are as proposed by Nguyen *et al.* [21] and they represent a surface, but formulations for curves and solids can be derived analogously.

Parametric space

The parametric space, denoted $\hat{\Omega}$, represent a patch, which is partitioned into elements defined by unique knot values. If the patch is normalized, the coordinates of the parametric space are defined on the interval $\xi, \eta \in [0, 1]$, and $\hat{\Omega} = [0, 1]^2$. The majority of the analysis are performed in this space, this is because NURBS are defined over a patch and thus all calculations associated with the basis functions are performed here. This space is not used in FEA because the shape functions are defined over the elements, and not patches.

Physical space

The parametric space is mapped into the physical space, denoted Ω , which represent the physical problem of interest and is defined by the physical coordinates $x, y \in \mathbb{R}$.

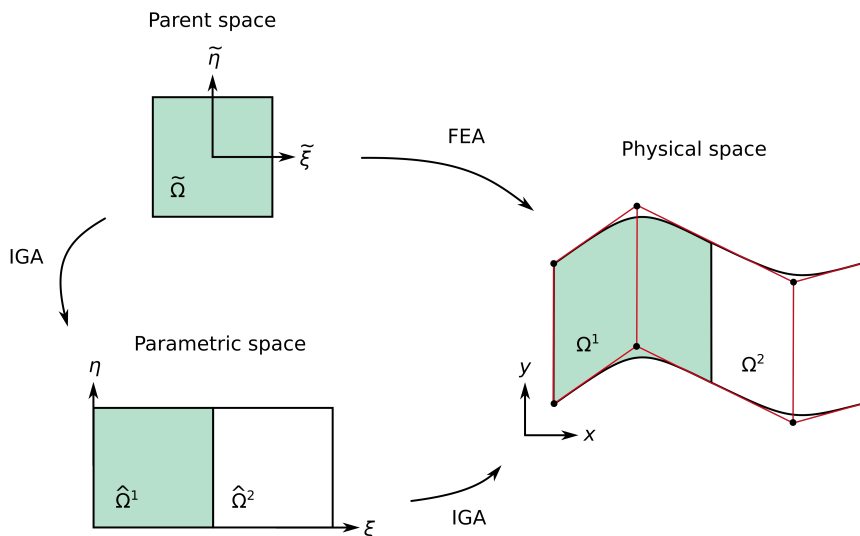


Figure 3.4: An illustration of the three different spaces for an arbitrary NURBS surface, and the mapping between them. Both the mapping used in IGA and FEA are presented.

Parent space

The parent space, denoted $\tilde{\Omega}$, is used to perform numerical integration (Gaussian quadrature) and the space represents an element in the patch. Gaussian quadrature integration requires a function in the range $[-1, 1]$, hence the parent space are defined in the space $[-1, 1]^2$, with coordinates $\tilde{\xi}, \tilde{\eta} \in [-1, 1]$.

As Figure 3.4 depicts, there are two mappings in IGA. First a mapping from parent space to parametric space, where the elements are joined together forming a patch. Then a mapping from parametric space to physical space. The remaining mapping on the figure illustrates the mapping in FEA from parent space to physical space. The colored area in the figure represents an element in the patch.

3.3 Code architecture

The code architecture applied in IGA to solve the linear system equation $\mathbf{K}\mathbf{d} = \mathbf{f}$ is equivalent to the procedure in FEA, where \mathbf{K} is the stiffness matrix, \mathbf{d} is the displacement vector and \mathbf{f} is the force vector. Figure 3.5 presents a flow chart of the code architecture. The only way that IGA differ from FEA in this procedure is due to the different basis functions used by the two

methods. This means that the steps where the basis function are evaluated, the code must be altered according to the basis that is applied. These steps are marked with a green color in the figure. The code architecture for solving the system equation can be divided into three steps: pre-processing, system assembly and post-processing.

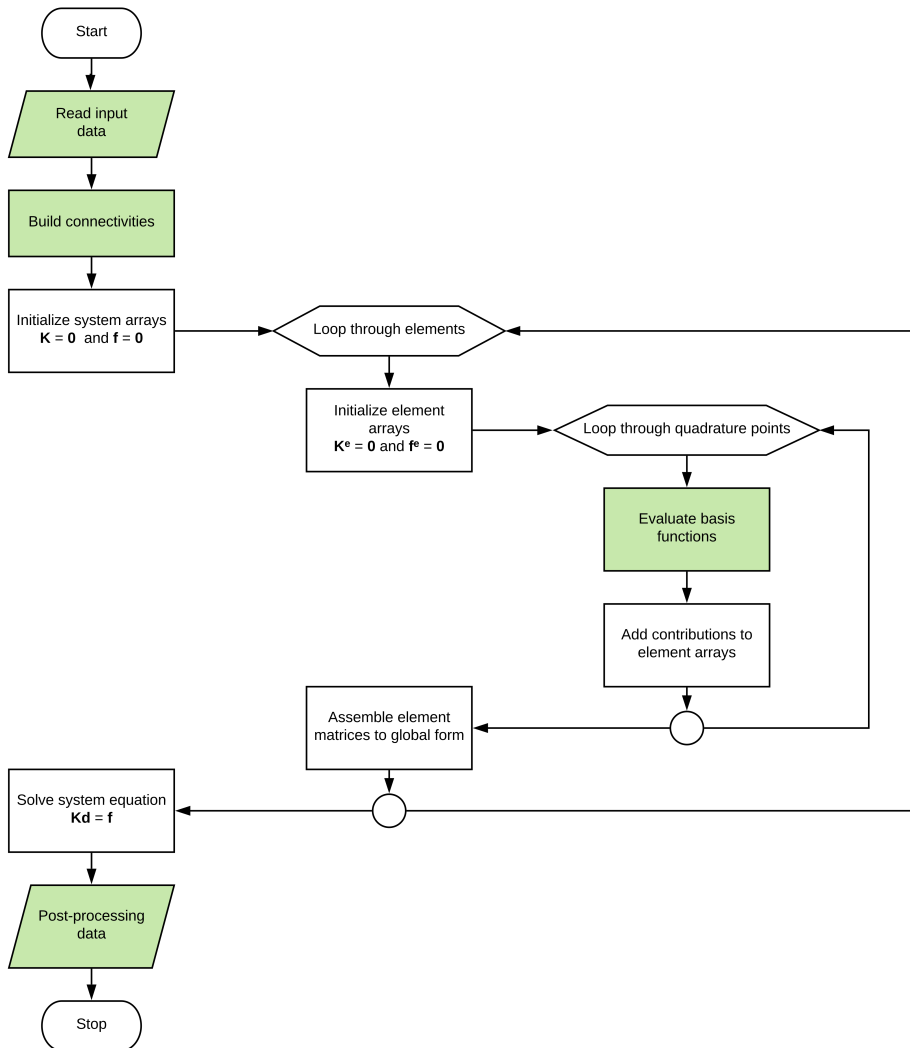


Figure 3.5: A flow chart of how to solve a linear system equation $\mathbf{Kd} = \mathbf{f}$ in both IGA and FEA. The green boxes represent the steps where the code must be altered according to which basis function that is applied.

Pre-processing

Before the calculations can be performed, the input data must be read. The input data consists of geometry definitions in terms of either NURBS or Lagrange polynomials, boundary conditions, loading and material properties. The next step is to establish the connectivities between nodes and elements, where nodes in IGA refers to the control points. The last step in the pre-processing step is to initialize the global system arrays, i.e. the stiffness matrix \mathbf{K} and the element force vector \mathbf{f} .

System assembly

In order to establish the stiffness matrix and the element force vector, each element in the patch has to be evaluated separately, hence a loop through all the elements. For each element, the local stiffness matrix and force vector are initialized and then each quadrature point is evaluated separately to establish the contributions to the system arrays from the basis functions and their derivatives.

Post-processing

After the system arrays are established, the system equation $\mathbf{Kd} = \mathbf{f}$ can be solved with respect to the displacements \mathbf{d} . And at last other desired quantities are established from the displacements and its derivatives, like stress distribution, internal energy and moment distribution.

3.4 Numerical methods

In common for both FEA and IGA is the need to convert a system of partial differential equations that cannot be solved analytically (defining the physical problem in a strong form) into a discrete problem that can be solved numerically, called the weak form of the problem. Some of the numerical methods that can be used to solve the weak form with IGA are: Galerkin's method, collocation, least-squares method and meshless method. In this thesis, Galerkin's method is applied, hence only this method will be described further. See Cottrell *et al.* [8] for a more detailed description of the three other methods.

The weak form of the problem is an equation consisting of integrals, thus a numerical method for solving integrals are needed. In this thesis the Gauss quadrature method is applied, which is also common for both IGA and FEA.

3.4.1 Galerkin's method

The Galerkin method is described as the foundation of FEA, but it is also applicable with IGA. The method is a weighted residual method used to calculate the global stiffness matrix. The method begins with defining the weak form of the problem based on a boundary value problem of differential equations that must satisfy the boundary conditions. The weak form consists of integrals that are multiplied with a weight to ensure that the residual equals zero. The displacements are the only unknown in the weak form, hence the set of integrals can be rearranged such that the system can be solved with respect to the displacements.

The weak form approximates the strong form, and it is defined such that the value of the integral of the weak form equals the integral of the strong form. This results in an approximated solution where the average over the element is exact, but pointwise the weak form is not necessarily equal to the strong form. Because of this, it is important to evaluate the approximated solution in the points where the solution of the weak form and the strong form are most alike.

3.4.2 Gauss quadrature integration

Gauss quadrature is a numerical method used to approximate the integral of a function defined on the interval $[-1, 1]$. In structural mechanics it is used to approximate the stiffness matrix, defined as

$$\mathbf{K} = \int_V \mathbf{B}^T \mathbf{D} \mathbf{B} dV, \quad (3.3)$$

where \mathbf{B} is the strain-displacement matrix and \mathbf{D} is the constitutive relationship matrix.

In general terms, an n -point Gaussian quadrature approximation of the function $f(x)$ can be derived in the following way

$$\int_{-1}^1 f(x) dx = \sum_{i=1}^n w_i f(x_i), \quad (3.4)$$

where w_i are the weight associated with the point x_i . With a higher number of evaluation points n , the better the approximated solution gets. In FEA, an integral can be calculated exactly with the Gauss quadrature rule if the number of Gauss points are determined by $p = 2n - 1$, where p is the polynomial degree. However, in IGA this formula cannot be used. For this thesis, the number of Gauss points have been set equal to $p + 1$, as proposed by Kiendl [15].

Chapter 4

The NURBS-based rotation-free Kirchhoff-Love shell

This chapter reviews the structural mechanics of shell elements in general and the Kirchhoff-Love shell element formulation, which is the element used in the simulations for this thesis. All equations and derivations are based off the work of Kiendl [15] and Coradello [7].

In structural mechanics, shell elements are defined as elements with three spatial dimensions where one of the three dimensions are substantially smaller, referred to as the thickness. Shells are able to represent both pure bending and axial (membrane) deformations, and a combination of the two. It is conventional to differentiate between two types of shells: thick and thin shells. The thin shell formulations are only applicable to geometries which satisfies the following slenderness criteria: $\frac{R}{t} > 20$, where R is the curvature radius, and t is the thickness [4]. In case of planar shell surfaces, as for a cantilever, the curvature radius can be replaced by its length in order to calculate the slenderness ratio.

The thick shell formulation is based on the Mindlin-Reissner equations, and is often referred to as a Mindlin-Reissner element. This shell accounts for shear deformations in the thickness direction caused by bending - which the thin shell formulation does not. The element is conventionally used for geometries which is not classified as thin, even though the element is applicable in all cases. This is due to its sensitivity to mesh distortion. The thin shell element on the other hand, uses what is called a direct approach and follows a Kirchhoff application [4]. In the direct approach the shell is viewed as a two-dimensional surface and is prescribed a thickness which substitutes the need of an explicit third dimension. This simplification causes the shell to

neglect shear deformations, which is a reasonable simplification in slender structures where shear deformations barely impacts the response. In non-slender structures shear deformations are not neglectable w.r.t. the response, thus the shear forces inflict a validity restriction on to the thin shell formulation. This is formulated as a slenderness criterion because slender structures suffer close to no shear deformations. When the shear deformations are neglected, the cross sections of the element will remain straight in the deformed state.

The advantage of using shell structures, is that shells are efficient in terms of material used versus load capacity, due to its ability to carry load through its shape. The disadvantage is that it requires a mathematical description of its shape and curvature and it is therefore more complex to describe compared to a three-dimensional continuum model, even though a shell is only described by two dimensions.

4.1 Mathematical notation

To efficiently and precisely lay out the mathematical derivations used in this thesis, and to make it into an easier read, all the notational conventions used will now be presented. The derivatives of a given function $X(j)$ w.r.t. to the parameter j , will be written in a compact format as $X_{,j}$. The Einstein summation convention is used along with the convention to let Latin indices $\in \{1,2,3\}$ and Greek indices $\in \{1,2\}$, to efficiently write out a summation as

$$n^i = \sum_{i=1}^3 n^i = n^1 + n^2 + n^3, \quad (4.1)$$

$$\theta_\alpha = \sum_{\alpha=1}^2 \theta_\alpha = \theta_1 + \theta_2. \quad (4.2)$$

Furthermore, upper case letters refer to quantities in the reference configuration and small case letters refer to quantities in the deformed configuration. This enables the displacements of a material point x to be written in a very compact format:

$$\mathbf{u} = \mathbf{x} - \mathbf{X} \quad (4.3)$$

Lastly, superscript indices are reserved for contravariant quantities, and subscript indices are reserved for covariant quantities, e.g. \mathbf{g}_i refers to the i^{th} covariant base vector and \mathbf{g}^i refers to the i^{th} contravariant base vector.

4.2 Differential geometry of surfaces

Curved surfaces can be cumbersome to deal with in a normal Cartesian coordinate system, even if the curvature is constant. However, if the curvature is constant, the surfaces can effortlessly be described in a curvilinear coordinate system. This is due to that curvilinear coordinate system has curved coordinate axes, thus a single or doubly curved surface will be precisely rendered with ease. The coordinate system can through locale invertible transformations be converted into a normal Cartesian system. This enables for an analysis where each element firstly is described with a local curvilinear coordinate system and the element's properties will be evaluated, then it gets swiftly converted into a normal global Cartesian coordinate system.

In a Cartesian coordinate system each material point \mathbf{x} , of an arbitrary surface can be represented in the following manner

$$\mathbf{x} = x^1 \mathbf{e}_1 + x^2 \mathbf{e}_2 + x^3 \mathbf{e}_3 = x^i \mathbf{e}_i, \quad (4.4)$$

where \mathbf{e}_i is the Cartesian base vectors and x^i is the length. In order to be able to use curvilinear coordinates, its local covariant basis has to be introduced, namely \mathbf{g}_i . The basis describes the coordinate differentials:

$$\mathbf{g}_i = \frac{\delta \mathbf{x}}{\delta \theta^i} = \mathbf{x}_{,i}, \quad (4.5)$$

where the dot product of the contravariant and covariant base vectors are defined as the Kronecker-Delta [1], where the Kronecker-Delta is identical to the identity matrix \mathbf{I}

$$\mathbf{g}_i \cdot \mathbf{g}^j = \delta_i^j = \begin{bmatrix} \delta_1^1 & \delta_1^2 & \delta_1^3 \\ \delta_2^1 & \delta_2^2 & \delta_2^3 \\ \delta_3^1 & \delta_3^2 & \delta_3^3 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}. \quad (4.6)$$

The contravariant base vector, as stated in Equation (4.5), describes how the axes changes with respect to the curvature, and the covariant base vector is an element of the dual space. See Figure 4.1 for a spatial illustration.

To obtain the contravariant base vectors, \mathbf{G}^α and \mathbf{g}^α , the covariant and contravariant metric coefficients of the surface are needed [7]. The coefficients are scalar products, and are defined as

$$G_{\alpha\beta} = \mathbf{G}_\alpha \cdot \mathbf{G}_\beta, \quad g_{\alpha\beta} = \mathbf{g}_\alpha \cdot \mathbf{g}_\beta. \quad (4.7)$$

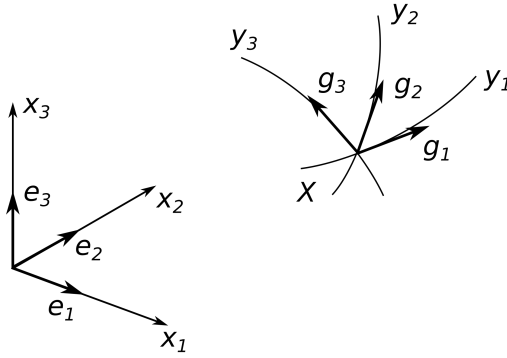


Figure 4.1: The normal Cartesian coordinate system to the left with Cartesian base vectors \mathbf{e}_i , and the curvilinear coordinate system to the right with the covariant bases \mathbf{g}_i .

Now, the contravariant base vectors can be written as the product of the inverse covariant metric coefficients and the covariant metric basis:

$$\mathbf{G}^\alpha = G^{\alpha\beta} \cdot \mathbf{G}_\beta, \quad \text{where} \quad G^{\alpha\beta} = [G_{\alpha\beta}]^{-1}. \quad (4.8)$$

The curvilinear equivalent to Equation (4.4) to represent a point on the surface is given as

$$\mathbf{x} = \theta^i \mathbf{g}_i = \theta_i \mathbf{g}^i. \quad (4.9)$$

Note that the point's coordinates will be given in a normal Cartesian format.

In a curvilinear coordinate system each point of the shell surface can be unambiguously represented by two curvilinear coordinates θ_1 and θ_2 . A consequence of this property is that \mathbf{g}_3 cannot be defined as formerly stated in Equation (4.5) while working in a curvilinear coordinate system. The simple way to remedy this is to use the definition of \mathbf{g}_3 , which states that \mathbf{g}_3 is defined as the normalized orthogonal vector to \mathbf{g}_1 and \mathbf{g}_2 . Hence, \mathbf{g}_3 is in algebraic terms defined as in Equation (4.10). Another decisive attribute of \mathbf{g}_3 , which follows from Equation (4.6), is that the third contravariant base vector is equal to the third covariant base vector, i.e. \mathbf{g}_3 is equal to \mathbf{g}^3 .

$$\mathbf{g}_3 = \frac{\mathbf{g}_1 \times \mathbf{g}_2}{|\mathbf{g}_1 \times \mathbf{g}_2|} \quad (4.10)$$

4.3 Structural mechanics of shells

Shells carries load in a distinctive and efficient way, and this attribute has made the curved layout ever-present. When transversal loads, for example gravity, are applied on to its curved surface, the loads are mostly carried by compression and tension [4]. This causes the occurring bending moments to be of a comparatively small intensity, and allows for larger spans or more complex geometries. In engineering terms, shells are sturdy because it carries the load induced forces as a combination of membrane and bending forces. Hence, when deriving the equations which lay the foundation of the shell's behavior, it is necessary to separate the membrane actions from the bending actions. The two modes of load bearing may reveal for the keen engineer that both plates and membranes can be declared as special cases, or incomplete formulations, of a shell [15]. Prior to embarking onto the specific formulation of the Kirchhoff-Love shell, some of the fundamentals of continuum mechanics will promptly be reviewed. Here, a Lagrangian description is used and the equations are based on the assumption of small strains and large displacements.

The kinematic equation in Equation (4.3) describes the deformation \mathbf{u} of an arbitrary point on the surface Ω , recall lower case letters refers to quantities in the deformed configuration and upper case letters to quantities in the reference configuration. As the incremental displacement is of interest, a relationship between $d\mathbf{x}$ and $d\mathbf{X}$ need to be established. Through considering the deformation gradient along a line in $d\mathbf{X}$ into $d\mathbf{x}$, the deformation pattern can be expressed in terms of a deformation gradient \mathbf{F} according to Holzapfel [11]:

$$\mathbf{F} = \frac{d\mathbf{X}}{d\mathbf{x}}. \quad (4.11)$$

This gradient describes all components of the deformation, and as it includes rigid body motions, it should not be a measurement for strains in itself. Rather, it can be used to define the Green-Lagrange strain tensor \mathbf{E} , which is a much better fit when rigid body motions are present. The Green-Lagrange strain is formulated in terms of \mathbf{F} and \mathbf{I} , and gives a nonlinear correlation between the strain and displacements as

$$\mathbf{E} = \frac{1}{2}(\mathbf{F}^T \cdot \mathbf{F} - \mathbf{I}). \quad (4.12)$$

Through tensor operations Equation (4.12) can be rewritten into a more convenient expression where \mathbf{E} will be defined in terms of the metric coefficients and the contravariant bases as [2][15]

$$\mathbf{E} = E_{ij} \mathbf{G}^i \otimes \mathbf{G}^j, \quad (4.13)$$

where

$$E_{ij} = \frac{1}{2}(g_{ij} - G_{ij}). \quad (4.14)$$

4.3.1 Kirchhoff-Love shell element

The strain tensor E will for a Kirchhoff-Love shell formulation be reduced to only account for the in-plane strain coefficients, as the transverse shear strain is neglected. Thus, Equation (4.13) is replaced by

$$E = E_{\alpha\beta} \mathbf{G}^\alpha \otimes \mathbf{G}^\beta \quad (4.15)$$

where

$$E_{\alpha\beta} = \frac{1}{2} (\mathbf{g}_\alpha \cdot \mathbf{g}_\beta - \mathbf{G}_\alpha \cdot \mathbf{G}_\beta) \mathbf{G}^\alpha \otimes \mathbf{G}^\beta. \quad (4.16)$$

From the assumptions made for the Kirchhoff-Love shell, all points on the surface can be described by its normal vector and the middle surface. This enables for a description where the basis vectors \mathbf{g} are replaced by the middle surface basis vectors, \mathbf{a} . The relationship between the two basis vectors is

$$\mathbf{g}_\alpha = \mathbf{a}_\alpha - \theta^3 \mathbf{a}_{3,\alpha}, \quad (4.17)$$

where $\theta^3 \in [-\frac{1}{2}t, \frac{1}{2}t]$ is the coordinate in the thickness direction. The curvature coefficients $b_{\alpha\beta}$, describing the bending occurring at the surface is defined as

$$b_{\alpha\beta} = \mathbf{a}_{\alpha,\beta} \cdot \mathbf{a}_3. \quad (4.18)$$

The motivation for this alteration lies in that the membrane strain and the bending strain components of the Green-Lagrange strain tensor can now be written explicitly as

$$E_{\alpha\beta} = \epsilon_{\alpha\beta} + \theta^3 \kappa_{\alpha\beta}, \quad (4.19)$$

where

$$\epsilon_{\alpha\beta} = \frac{1}{2} (a_{\alpha\beta} - A_{\alpha\beta}), \quad (4.20)$$

and

$$\kappa_{\alpha\beta} = B_{\alpha\beta} - b_{\alpha\beta}. \quad (4.21)$$

With the membrane strain and the bending strain separated, the normal and moment forces can now be deduced from the strains. This is conventionally referred to as stress recovery. The constitutive behavior of the material describes the stress-strain relations for the shell, and for this thesis Hooke's law will be used in order to extract the stresses from the Green-Lagrange strain tensor. By Hooke's law, the 2nd order Piola-Kirchhoff stress resultant measurements \mathbf{n} and \mathbf{m} will

be linearly dependent on the membrane and bending strain tensor, denoted $\boldsymbol{\epsilon}$ and $\boldsymbol{\kappa}$ respectively [7][2]. The stress-strain relationship is defined as

$$\mathbf{n} = \frac{Eh}{1-\nu^2} \mathbf{D} : \boldsymbol{\epsilon} = \mathbf{D}_m : \boldsymbol{\epsilon}, \quad (4.22)$$

$$\mathbf{m} = \frac{Eh^3}{12(1-\nu^2)} \mathbf{D} : \boldsymbol{\kappa} = \mathbf{D}_b : \boldsymbol{\kappa}, \quad (4.23)$$

where E is the Young's modulus, ν is Poisson's ratio and \mathbf{D} is the isotropic material tensor defined as

$$\mathbf{D} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \frac{1-\nu}{2} \end{bmatrix}. \quad (4.24)$$

The strain tensors correspond to the Green-Lagrange strain measure, which is the energetically conjugate of the second Piola-Kirchhoff stress measures. As the quantities used to express the strains in Equations (4.15)-(4.21) are derived from both the reference and the actual configuration, this formulation is only valid in a nonlinear setting. Thus, the linear analysis requires linearized versions of Equations (4.20) and (4.21). The strain components $\epsilon_{\alpha\beta}$ and $\kappa_{\alpha\beta}$ will in a linear analysis be defined through the basis vectors \mathbf{G}_i [14]

$$\epsilon_{\alpha\beta} = \frac{1}{2} (\mathbf{G}_\beta \cdot \mathbf{u}_{,\alpha} + \mathbf{G}_\alpha \cdot \mathbf{u}_{,\beta}), \quad (4.25)$$

$$\begin{aligned} \kappa_{\alpha\beta} = & -\mathbf{G}_3 \cdot \mathbf{u}_{,\alpha\beta} + \mathbf{G}_{\alpha,\beta} \cdot \mathbf{G}_3 \frac{1}{J} ((\mathbf{G}_2 \times \mathbf{G}_3) \cdot \mathbf{u}_{,1} - (\mathbf{G}_1 \times \mathbf{G}_3) \cdot \mathbf{u}_{,2}) \\ & + \frac{1}{J} ((\mathbf{G}_{\alpha,\beta} \times \mathbf{G}_2) \cdot \mathbf{u}_{,1} - (\mathbf{G}_{\alpha,\beta} \times \mathbf{G}_1) \cdot \mathbf{u}_{,2}), \end{aligned} \quad (4.26)$$

where J is the L_2 -norm of G_3

$$J = \|\mathbf{G}_3\|_2. \quad (4.27)$$

4.3.2 NURBS discretized Kirchhoff-Love shell

The stiffness matrix is derived from the equilibrium condition of virtual work, also known as the weak form of the problem. The weak form of the problem satisfies the strong form of the Kirchhoff-Love boundary value problem (BVP) in an integrated manner only, and not in each point. This shortcoming is however not only present for the Kirchhoff-Love shell, but the very purpose for the finite element method discretization. For a review of the derivation of the weak form from the BVP, the reader is referred to Coradello [7].

Shell elements are prone to locking phenomena which will severely impact its precision. There exist multiple ways to remedy this, or at least reduce the effect of locking. The phenomena

occur when the basis functions are not able to properly model the physical behavior, and is caused by low polynomial orders of the basis functions. With the use of NURBS basis functions, which is higher order functions, the locking behavior of elements in IGA are precluded.

The equilibrium condition of virtual work states that for a virtual displacement of an infinitely small size $d\delta$ the external virtual work done by the forces are opposite and equal in magnitude to that of the internal virtual work. This condition is stated in mathematical terms as

$$\delta W = \delta W_{int} + \delta W_{ext} = 0, \quad (4.28)$$

where δW is the virtual work. Thus, if Equation (4.28) is not satisfied, the system is not in equilibrium. The virtual work are defined as

$$\delta W_{int} = - \int_{\Omega} \mathbf{S} : \delta \mathbf{E} d\Omega, = - \int_A (\mathbf{n} : \delta \boldsymbol{\epsilon} + \mathbf{m} : \delta \boldsymbol{\kappa}) dA, \quad (4.29)$$

$$\delta W_{ext} = \int_{\Omega_b} \mathbf{T} \cdot \delta \mathbf{u} d\Omega_b + \int_{\Omega} \rho \mathbf{B} \cdot \delta \mathbf{u} d\Omega, \quad (4.30)$$

where \mathbf{S} is the PK2 stress tensor, \mathbf{T} is the boundary forces, \mathbf{u} is the displacements, Ω_b is the domain boundary in the reference configuration, ρ is the material density, \mathbf{B} is the body forces and Ω denotes the domain.

Equation (4.28) has to be true for an arbitrary virtual displacement δu_r which yields that the incremental virtual work needs to be equal to zero

$$\delta W = \frac{\partial W}{\partial u_r} \delta u_r = 0, \quad (4.31)$$

$$\frac{\partial W}{\partial u_r} = 0. \quad (4.32)$$

The derivative of the internal virtual work with respect to a displacement is the internal force vector F^{int} , and vice versa for the external force vector F^{ext} . The sum of the internal and external force yields the residual force vector \mathbf{R} . Thus, Equation (4.32) can be expressed as

$$R_r = \left(\frac{\partial W_{int}}{\partial u_r} + \frac{\partial W_{ext}}{\partial u_r} \right) = F_r^{int} + F_r^{ext}. \quad (4.33)$$

By combining Equation (4.33) and (4.29), the internal nodal forces can be expressed by the membrane and bending strains

$$F_r^{int} = - \int_A \left(\mathbf{n} : \frac{\partial \boldsymbol{\epsilon}}{\partial u_r} + \mathbf{m} : \frac{\partial \boldsymbol{\kappa}}{\partial u_r} \right) dA. \quad (4.34)$$

The stiffness matrix \mathbf{K} is the second derivative of the virtual work, and can be split into its two components: the internal stiffness matrix \mathbf{K}^{int} and the external stiffness matrix \mathbf{K}^{ext} . By taking the derivative of the residual vector w.r.t. u_s , the following equation is attained

$$K_{rs} = - \left(\frac{\partial^2 W_{int}}{\partial u_r \partial u_s} + \frac{\partial^2 W_{ext}}{\partial u_r \partial u_s} \right) = K_{rs}^{int} + K_{rs}^{ext}. \quad (4.35)$$

The definition of the external stiffness matrix \mathbf{K}^{ext} , states that the matrix is only relevant in case of displacement-dependent loads, which is not in the scope in this thesis, hence it will be neglected.

The internal stiffness can be defined by inserting Equation (4.29) into Equation (4.35), which yields

$$K_{rs}^{int} = \int_A \left(\frac{\partial \mathbf{n}}{\partial u_s} : \frac{\partial \boldsymbol{\epsilon}}{\partial u_r} + \mathbf{n} : \frac{\partial^2 \boldsymbol{\epsilon}}{\partial u_r \partial u_s} + \frac{\partial \mathbf{m}}{\partial u_s} : \frac{\partial \boldsymbol{\kappa}}{\partial u_r} + \mathbf{m} : \frac{\partial^2 \boldsymbol{\kappa}}{\partial u_r \partial u_s} \right) dA. \quad (4.36)$$

Here, the membrane stiffness is represented in the two first terms, and the membrane stiffness is represented in the last two. The second and last term account for the nonlinear geometric stiffness contributions, and is not relevant for a linear analysis. If a linear analysis is sufficient for the analysis, Equation (4.36) can be used in a nonlinear equation system if only one iteration is performed. This is however a comparatively inefficient way to do the linear analysis. Thus, it is convenient to alter the equations slightly for a linear analysis. The quantities computed from the differences between the actual and reference configuration will now become equal to zero. Furthermore, the virtual internal force will be identical to zero. The consequences from this linearization is that the equation system now is reduced to

$$\mathbf{K}^{lin} \mathbf{u} = \mathbf{F}^{ext}, \quad (4.37)$$

where

$$K_{rs}^{int,lin} = \int_A \left(\frac{\partial \mathbf{n}}{\partial u_s} : \frac{\partial \boldsymbol{\epsilon}}{\partial u_r} + \frac{\partial \mathbf{m}}{\partial u_s} : \frac{\partial \boldsymbol{\kappa}}{\partial u_r} \right) dA, \quad (4.38)$$

which is the typical way to describe a geometrically linear system. An interesting remark is that the nonlinear stiffness matrix in Equation (4.36) contains second derivatives, while the linear stiffness matrix does not. This difference causes the linear version to have a comparatively exponentially increasing efficiency w.r.t. the number of DOFs compared to the nonlinear version.

4.3.3 Discretization

All derivations and resulting equations are valid in general for a Kirchhoff-Love shell, and is not specific for the Kirchhoff-Love shell element. The NURBS discretization provides an exact

description of the geometry, and will with ease produce continuous second derivatives which is highly beneficial when curvatures are to be evaluated. The first step is to establish the position vector, see Equation (4.4), from the shape functions N^i and the discrete nodal values $\hat{\mathbf{x}}$

$$\mathbf{x} = \sum_i N^i \hat{\mathbf{x}}^i. \quad (4.39)$$

Discrete nodal values are denoted by a circumflex. To obtain the derivative of the displacement variables, the derivative of the deformation gradient \mathbf{u} has to be used in the following manner

$$\mathbf{x}_{,r} = \sum_i N^i (\hat{\mathbf{x}}^i + \hat{\mathbf{u}}^i)_{,r} = \sum_i N^i \hat{\mathbf{u}}_{,r}^i, \quad (4.40)$$

where all quantities of the reference configuration are invariant to the displacement variations, thus becoming redundant in the equation. The remaining quantities which need to be discretized must all be derived through the same summation process as the two former equations. For this purpose, a general equation will be provided, which is applicable for all the needed sizes. The discretization for an arbitrary quantity \mathbf{M} differentiated with respect to α, β, r and/or s , the quantity can be found by

$$\mathbf{M}_{\alpha, \beta, r s} = \sum_i N^i_{, \alpha \beta} \hat{\mathbf{u}}_{, r s}^i, \quad (4.41)$$

where the Greek subscripts of \mathbf{M} corresponds to the derivatives of N^i , and the Latin subscripts corresponds to the derivatives of $\hat{\mathbf{u}}^i$. Here, the subscripts may be left empty, hence the equation may be used to find $\mathbf{M}_{\alpha, r}$. Equation (4.41) states how to attain the second derivatives, which are required in order to calculate the bending strain tensor $\boldsymbol{\kappa}$ as the variational index is two. For a NURBS discretization, this requirement will never present itself as a major challenge, as the trans-elemental continuity easily can be elevated. However, this is a weakness of the Kirchhoff-Love shell element from a standard FEA viewpoint because C^1 -continuity cannot be guaranteed for an arbitrary surface. This obstacle is the reason why the Kirchhoff-Love shell formulation is rarely used in FEA, and the Mindlin-Reissner shell formulation is often used instead even if it is appropriate to consider the element as a thin shell.

An interesting property to point out is that when large deformations are present, the Green strain tensor is used and is the reason for why the variational index is two. In linear small displacement analysis however, the variational index will be one, as the Green strain tensor is not needed.

Chapter 5

Implementation of the IGA shell element

This chapter discusses the implementational aspects of this thesis. The first part of this chapter is a review of the implementation, with a presentation of the classes and methods used in the object-oriented environment, and how these are related to each other. The last part of this chapter discusses some of the common implementational issues that appears when dealing with IGA or in general system equation solving.

5.1 Implementation

The purpose of this thesis is to implement the Kirchhoff-Love shell theory in to the open-source software IFEM Elasticity [24], which applies IGA to solve both linear and nonlinear differential equations for elasticity problems in structural mechanics. IFEM Elasticity is a module of IFEM developed by SINTEF Digital together in cooperation NTNU as a part of the ICADA project. The implementation that is performed for this thesis was developed using the programming language C++.

Classes for both linear and nonlinear solving of the integral defining the stiffness matrix were implemented, while classes used for system assembly were already established in the program. Hence, only the Kirchhoff-Love shell specific classes were implemented for this thesis. These classes are based on a MATLAB implementation made by Kiendl [15] as a part of his PhD thesis, which is not an open-source code and therefore it cannot be shared in this thesis.

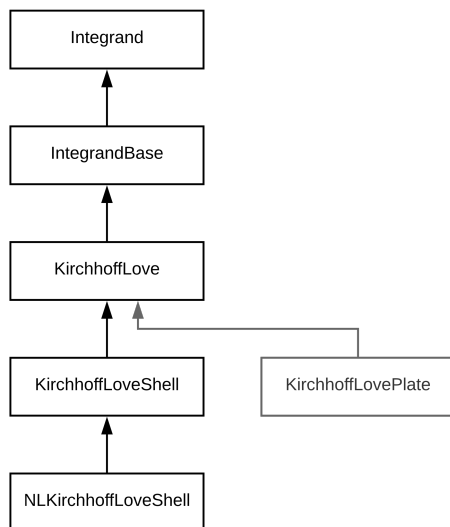


Figure 5.1: Inheritance diagram of the integrand classes in IFEM Elasticity used to establish the stiffness matrix of the Kirchhoff-Love shell element.

5.1.1 Classes

The two specific classes implemented for this thesis are named *KirchhoffLoveShell* and *NLKirchhoffLoveShell*, which represents the integrand used to evaluate the stiffness matrix of a thin shell problem with Kirchhoff-Love shell theory for both linear and nonlinear systems, respectively. Figure 5.1 presents an inheritance diagram that illustrates the relationship between the two classes that are implemented, and the classes that these inherit from. As stated in the figure, the nonlinear class inherits from the linear class.

The class named *Integrand* is at the top of the inheritance hierarchy. This is an abstract class that represents an integrated quantity, which in the case of this thesis is the stiffness matrix. The main purpose of this class is to construct the interface connecting the finite element assembly classes with the problem specific classes, which in this case are the implemented classes containing the Kirchhoff-Love shell theory. The assembly classes loop through the elements and call on the integrand classes to evaluate the local stiffness matrix at each element. The assembly classes are not implemented for this thesis; hence they will not be discussed further. The next class in the hierarchy is the base class that implements the abstract methods of the class *Integrand*, which is named *IntegrandBase*.

The integrand class *KirchhoffLove* is an interface defining the methods and parameters that are in common for both shell and plate Kirchhoff-Love theory, but the actual implementation of the methods is located in the sub-classes, except for the methods that evaluated the mass matrix and the body force vector, as they are in common for both shells and plates. The class *KirchhoffLovePlate* is an integrand class that represent the Kirchhoff-Love plate element and it inherits from the Kirchhoff-Love interface, but the class is not implemented for this thesis and therefore it is not further discussed.

The two last classes, *KirchhoffLoveShell* and *NLKirchhoffLoveShell*, are the ones that are implemented for this thesis. They both represent the integrand of thin shell problems based on the Kirchhoff-Love shell theory with linear and nonlinear solvers, respectively. The nonlinear class inherits from the linear as it extends the linear class with implementation that is needed to perform the nonlinear analysis.

5.1.2 Methods

This section presents the methods that are implemented in the linear class and a description of how the nonlinear class is extended from the linear to deal with nonlinear effects. There exist more methods in the classes, but as they are not implemented or important for this thesis, they will remain undisclosed.

Linear implementation

The methods that are implemented in the linear class *KirchhoffLoveShell* are presented in Figure 5.2. The figure illustrates the relationships between the methods and states all the parameters, both input and output, associated with the method together with a brief explanation. The two public methods, *evalInt* and *evalSol*, are the ones that are called from the assembly classes. These methods subsequently call on the private methods *evalK*, *formBmatrix* and *formDmatrix*.

The method *evalInt* is inherited from the class *Integrand*, and it evaluates the integrand at an interior point. The output of this method is a local integral object, that receives contributions to the stiffness matrix, the mass matrix and the body force vector. This method calls on three methods to establish these contributions: *evalK*, *formMassMatrix* and *formBodyForce*. The two latter methods establish the mass matrix and the body force vector contributions at the current integration point, respectively. These methods are common for all Kirchhoff-Love integrand classes and are therefore implemented in the class *KirchhoffLove*.

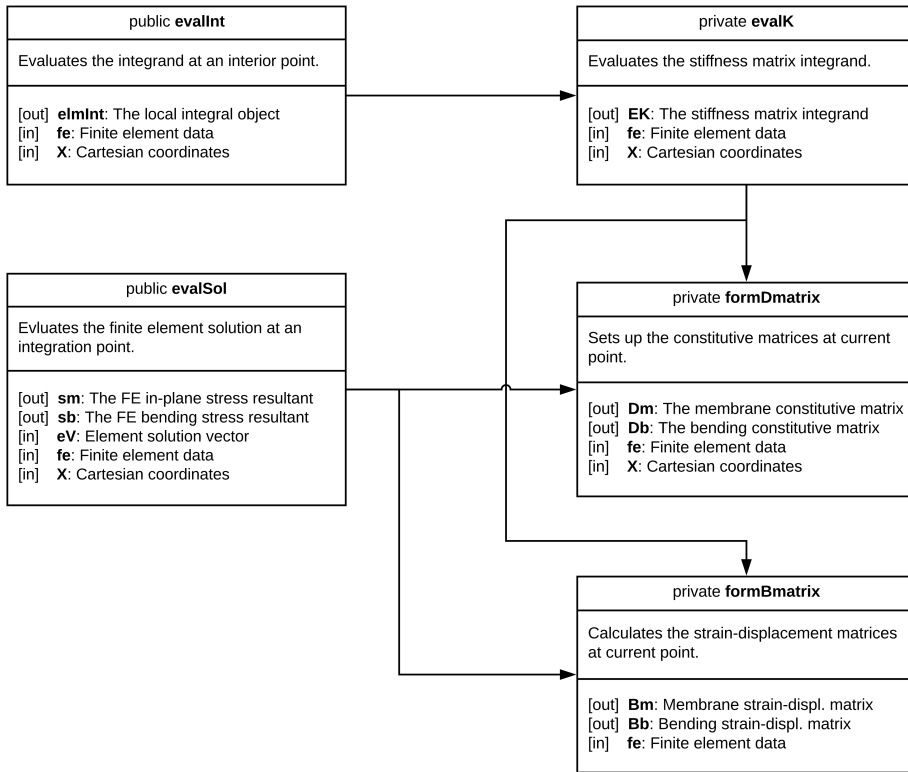


Figure 5.2: Method diagram of the implemented methods in the integrand class *KirchhoffLoveShell* that establish the local integral object at the current integration point.

The method *evalK* evaluates the stiffness matrix integrand. The method begins by defining the constitutive matrices, \mathbf{D}_m and \mathbf{D}_b , and the strain-displacement matrices, \mathbf{B}_m and \mathbf{B}_b , where subscript m represents the contributions from membrane forces and subscript b the contributions from bending moment. These matrices are established by calling the methods *formDmatrix* and *formBmatrix*, respectively. When the constitutive and strain-displacement matrices are established, the calculations of the stiffness matrix can be performed. In order to do so, the local stiffness matrices with contributions from membrane forces and bending moment, \mathbf{k}_m and \mathbf{k}_b , must be established first as

$$\mathbf{k}_i = \mathbf{B}_i^T \cdot \mathbf{D}_i \cdot \mathbf{B}_i \cdot J_w, \quad (5.1)$$

where i is either m or b and J_w is the weighted Jacobian determinant of the coordinate mapping, which is a variable of the finite element class. When the strain tensors are established, they are both added as contributions to the global stiffness matrix \mathbf{K} .

The method *evalSol* evaluates the finite element solution at the current integration point and it returns the stress resultant tensors. The method begins by establishing the strain-displacement matrices and the constitutive matrices, as in *evalK*. Then it evaluates the membrane strain $\boldsymbol{\epsilon}$ and curvature tensor $\boldsymbol{\kappa}$ by the definitions

$$\boldsymbol{\epsilon} = \mathbf{B}_m \cdot \mathbf{eV}, \quad (5.2)$$

$$\boldsymbol{\kappa} = \mathbf{B}_b \cdot \mathbf{eV}, \quad (5.3)$$

where \mathbf{eV} is the element solution vector. After these are established, the stress resultant tensors from membrane strain and curvature, \mathbf{sm} and \mathbf{sb} , can be evaluated as

$$\mathbf{sm} = \mathbf{D}_m \cdot \boldsymbol{\epsilon}, \quad (5.4)$$

$$\mathbf{sb} = -\mathbf{D}_b \cdot \boldsymbol{\kappa}. \quad (5.5)$$

Finally, the stress resultant tensors are transformed to local coordinate system.

Nonlinear implementation

The class *NLKirchhoffLoveShell* holds implementation specific for a nonlinear analysis. The methods that are changed in this class from the linear class are: *evalK*, *formBmatrix* and *evalSol*. The main difference is the fact that the nonlinear calculations has to differentiate between the initial configuration and the deformed configuration of the element. In addition to this, the method *evalK* has additional implementation to establish the internal forces at the current integration point. The method *evalSol* includes implementation to calculate nonlinear strains and stress resultants, based on initial and deformed configuration.

5.2 Implementational Issues

This section presents some of the issues that were relevant for the implementation of the thin shell element. The topics that are discussed in this section are: connectivity matrices, edge collapse, loading, boundary conditions together with symmetry conditions and aspects specific to the nonlinear implementation, such as the Newton-Raphson iteration method.

5.2.1 Connectivity matrices

Establishing the global stiffness matrix \mathbf{K} can be done in two different ways. One option is to loop through all entries in the stiffness matrix and calculate it directly by using global shape functions. The other option is to calculate the element stiffness matrices and add the local matrices into the global stiffness matrix. The latter strategy will be the preferred one because it will save time as it will not evaluate the stiffness entries which will be equal to zero [8]. In order to successfully implement the method which will establish the global stiffness matrix from the local ones, a connectivity array, denoted IEN , is a prerequisite. The notation used here is as proposed by Cottrell *et. al.* [8]. The connectivity array will be the bookkeeper of every local shape function w.r.t. the global shape functions, and will be defined in such a manner that for an element e and for a local function a , the IEN will give the corresponding global shape function A . Said in mathematical terms: $IEN(a,e) = A$. Here, $e \in [1, \dots, n_{el}]$, $a \in [1, \dots, n_{en}]$ and $A \in [1, \dots, n_{eq}]$, where n_{el} is the number of elements, n_{en} is the number of local shape functions, and n_{eq} is the number of global shape functions. This process is simultaneously and analogously done for the force vector \mathbf{f} .

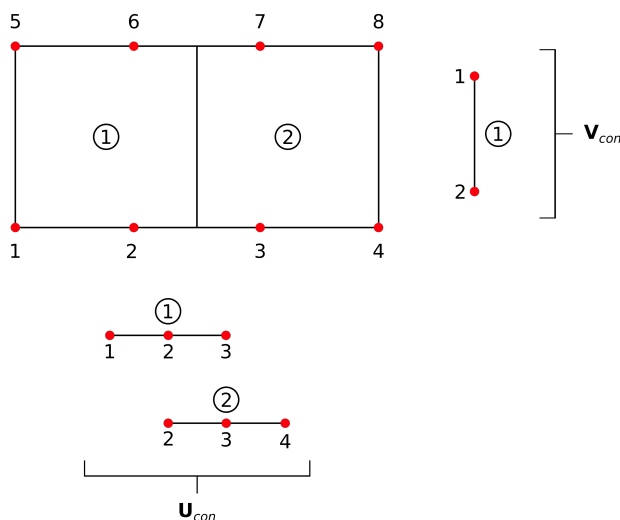


Figure 5.3: Bi-quadratic NURBS with two elements. The basis functions are derived from the knot vectors, and plotted along the corresponding axis in order to better illustrate how the local basis functions influence each element.

This process may turn out to be quite intricate, and increasingly so if the geometry is refined by adding elements in multiple directions. Thus, the process of establishing a connectivity array will be conducted once for a demonstrative purpose, where the notation used will be as proposed by Nguyen [21]. The IEN matrix will have the dimension $n_{el} \times (p+1)(q+1)$, where p and q is the polynomial degree in ξ and η direction, respectively.

In Figure 5.3, the knot vectors and basis functions for a bi-quadratic NURBS surface are presented. From the knot vectors, the surface is defined to have two elements in the ξ -direction, and only one in the η -direction. The corresponding control points are included in the figure, where each element has four control points. In this example, each element has contributions from three basis functions in the ξ -direction and one in the η -direction. Here, the node pattern is given as

$$\text{node pattern} = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \end{bmatrix},$$

and the two separate connectivity matrices in two directions are given as

$$\mathbf{U}_{con} = \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 4 \end{bmatrix}, \quad \text{and} \quad \mathbf{V}_{con} = \begin{bmatrix} 1 & 2 \end{bmatrix},$$

where each entry refers to the row of control points along each axis. Then the full connectivity matrix for this example will be

$$IEN = \begin{bmatrix} 1 & 2 & 3 & 5 & 6 & 7 \\ 2 & 3 & 4 & 6 & 7 & 8 \end{bmatrix}.$$

5.2.2 Edge collapse

Edge collapse is a way to degenerate a quadrilateral into a triangle, which is used when the physical surface has a triangular shape. The process will collapse one of the edges into one vertex, which is illustrated in Figure 5.4. This will cause multiple redundant nodes at this point. The redundant nodes will then effectively be treated with a master-slave technique which purpose is to sum all contributions from the nodes along the collapsed edge and add them to the master node's DOFs. The slave nodes will then be removed from the system matrices, which will result in a decrease of the total number of nodes, and the node numbering will be changed accordingly.

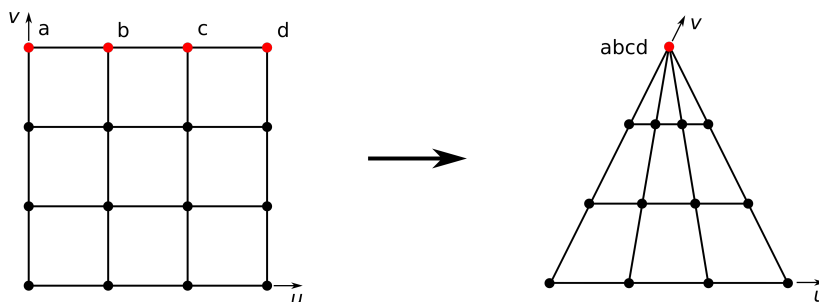


Figure 5.4: Illustration of an edge collapse when mapping the quadratic parametric space into a triangular shape.

5.2.3 Loading

In the simulations done for this thesis, the load was applied as either a point load or a distributed load. In the case of a point load, it has to be located in a control point. For most cases, the basis functions do not interpolate the control points, except for at the start and end knot. Therefore, a point load must be placed in a control point that is interpolated with the basis function, which are at the corners of the patch. In the case of a single-patch analysis, this is often not sufficient as some examples have point loads placed not only in the corners. This problem can in some cases be solved by using symmetry conditions.

In the case of a distributed or gravitational load the forces will span over multiple nodes. The load is first computed into a pressure field with three directional components, where one or more of the inputs may be zero, and multiplied with the basis function's value at each node and the weighted determinant.

5.2.4 Boundary conditions

Boundary conditions are vital in any analysis, and is part of a problem description. The boundary conditions, here referred to as BCs, define nodes or edges where translation or rotation is fixed. Every problem definition is required to include BCs which restrain the geometry in such a way that rigid body motions cannot occur when loads are applied onto the system. The BCs are categorized into two main categories: Neumann BCs and Dirichlet BCs. The Dirichlet BCs are

often referred to as the *first-type* BCs as they impose requirements for the solution's value at a set of points or along an edge. Fixed translations and fixed rotations are examples of Dirichlet BCs. The Neumann BCs on the other hand impose requirements for the solution's derivative at discrete points, and is often referred to as *second-type* BCs. As Dirichlet are the only BC used in this thesis, Neumann conditions will therefore not be discussed any further.

Translational BCs are easily treated in a NURBS setting by simply defining the corresponding DOFs at the chosen nodes to take on the value specified by the Dirichlet BC. Rotational BCs, like clamped BCs, demand a slightly more nuanced method as the element in question is formulated as a rotation-free element. This is achieved by controlling the slope of the surface along the edge, and as the slope is determined by the two rows of control points closest to the edge, the interior control points must be restrained from translation. For example, in the case of a clamped hemisphere, the first two rows of control points will be restrained from translation in all three directions, and in the case of a cantilevered beam, the interior row of control points have to be fixed in vertical direction. It may seem like clamped boundaries will introduce supports at the second row of control points, but this however is not the case as the procedure will only affect the nodes along the boundary.

Symmetry

Symmetry boundaries can be applied onto a geometry along its symmetry planes in order to alleviate the necessity of using the entire geometry in an analysis. This enables for example for an analysis of a hemispherical surface by only examining one quarter of the hemisphere. The advantage of using symmetry boundaries is that the total number of elements can be reduced, thus increasing the computational efficiency. In addition, circular problems can be undertaken without using multiple patches or creating arches with lowered continuity.

The symmetry plane boundaries are imposed by a master-slave technique where the slope between the master and the slave node remains constant throughout the analysis. This is achieved, in general terms, by restricting the translation in the normal direction of the plane, and keeping the rotations along the surface normal and boundary directions equal to zero. In a NURBS setting with a rotation-free element, this requirement is fulfilled by setting the translation of the slave node equal to that of the master node in the two in symmetry plane directions and setting the remaining translation equal to zero. In Figure 5.5, an example of symmetry boundary in the xz -plane is provided. Here, the longitudinal translation is zero, and the slave node's translation in the x and z direction is equal to the master node's.

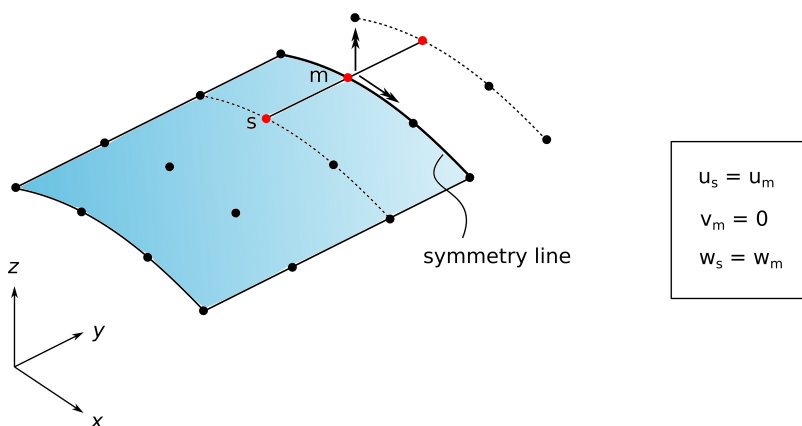


Figure 5.5: Symmetry boundary condition. The slave node in the interior will be equal to that of the exterior. Note that the rotational DOFs in the figure is not relevant for this implementation as the shell is formulated without rotational DOFs. Here, subscript s and m denotes slave and master, respectively.

5.2.5 Nonlinear implementation

In a nonlinear system the Galerkin method may still be used, but the system equation $\mathbf{K}\mathbf{d} = \mathbf{f}$ becomes a system of nonlinear algebraic equations. This means that instead of solving the system equation directly, as in linear analysis, the system has to be solved with an iterative equation solver for each load increment. For this thesis the Newton-Raphson iteration method is used.

Newton-Raphson method

The Newton-Raphson method, referred to as the NR method in this thesis, is an iterative method used to find roots of nonlinear equations. Let $\mathbf{R}(\mathbf{d}) = \mathbf{K}\mathbf{d} - \mathbf{f} = 0$ be a set of nonlinear equations, which represents the residual of the system equations, with exact solution $\bar{\mathbf{d}}$. Then $\bar{\mathbf{d}} = \mathbf{d}_i + \delta\mathbf{d}_i$, where \mathbf{d}_i is an approximated solution and $\delta\mathbf{d}_i$ is the unknown error, referred to as the iterative displacement.

The first step of the method is to start with an trial displacement, denoted \mathbf{d}_0 , chosen by the analyst, then estimate the iterative displacement $\delta\mathbf{d}_0$ to obtain a new (and better) approximated displacement $\mathbf{d}_1 = \mathbf{d}_0 + \delta\mathbf{d}_0$. This is an iterative process where a new approximated solution are

estimated from the previous solution on the form $\mathbf{d}_{i+1} = \mathbf{d}_i + \delta\mathbf{d}_i$ until a convergence requirement $\mathbf{R}(\mathbf{d}_i) \leq \epsilon$ is fulfilled, where ϵ is the convergence criteria and is determined by the analyst.

In order to perform the NR method, an approximation of $\delta\mathbf{d}_i$ must be determined. This is found by Taylor expansion of the residual about the points \mathbf{d}_i

$$\mathbf{R}(\bar{\mathbf{d}}) = \mathbf{R}(\mathbf{d}_i + \delta\mathbf{d}_i) \approx \mathbf{R}(\mathbf{d}_i) - \mathbf{K}_T(\mathbf{d}_i)\delta\mathbf{d}_i, \quad (5.6)$$

$$\Rightarrow \delta\mathbf{d}_i \approx \frac{\mathbf{R}(\mathbf{d}_i)}{\mathbf{K}_T(\mathbf{d}_i)}, \quad (5.7)$$

where $\mathbf{K}_T = -\frac{d\mathbf{R}(\mathbf{d}_i)}{d\mathbf{d}}$ is the tangent stiffness.

In the nonlinear IGA the NR method is used to determine the equilibrium path, hence it will be employed at each load step. The NR method will in its zeroth iteration, the predictor step, check how well the initial result performed with respect to the convergence criteria, and will perform multiple iterations until the criteria is met. The use of the Newton-Raphson method enforces a restriction onto the analysis, because it can only track a monotonic equilibrium path. Thus, examples where the equilibrium path has critical points (limit, bifurcation, failure or turning points), it will diverge. This problem can be circumvented by implementing an enhanced iteration method, like the arc length method, but this is not in the scope of this thesis.

The convergence criteria used in this thesis and in IFEM Elasticity is formulated as

$$\Delta E = \Delta\mathbf{u} \cdot \mathbf{R} \leq \epsilon_{tol} E_{ref}, \quad (5.8)$$

where ΔE is the energy norm, $\Delta\mathbf{u}$ is the incremental solution vector, \mathbf{R} is the residual force vector, ϵ_{tol} is the convergence tolerance, and E_{ref} is the reference energy norm. The reference energy norm is set equal to the highest energy norm found in the zeroth iteration. The convergence results reported will be: the incremental L_2 -norm of the solution vector denoted u_{norm} , the L_2 -norm of the residual force vector denoted R_{norm} , the energy norm ΔE , and the convergence ratio of the energy norm compared to the reference energy norm denoted ϵ_i .

Chapter 6

Verification of the IGA shell element

This chapter verifies the implementation of the Kirchhoff-Love shell element discussed in chapter 5 and the associated shell formulation, discussed in chapter 4, with benchmark examples. The first part of this chapter verifies the linear implementation of the shell element, while the last part deals with the nonlinear verification. The linearly implemented element will be tested against five acknowledged linear shell benchmark examples, and the shell's performance will be evaluated in terms of the precision and convergence rate of the solution, and its stress distribution. The nonlinear Kirchhoff-Love shell element will be tested against three benchmark examples, where the results will be assessed in terms of the precision of the solution, convergence of the NR iterations and its stress distribution.

6.1 Benchmark examples with geometric linearity

To verify the linear implementation of the shell element, a convergence study has been conducted. For all the linear benchmark problems, the relative error of the quantity of interest, denoted δx , used in the convergence plots is established as

$$\delta x = \frac{\bar{x} - x}{x}, \quad (6.1)$$

where x is the reference solution and \bar{x} is the calculated solution. The reference solutions are found with multimesh extrapolation, which is explained in more detail in Appendix B. This is done to get a solution with a higher numerical precision than the solutions associated with the benchmark examples. In addition, a different element and analysis method is used, therefore the

solutions will not converge to exactly the same values as proposed by the benchmark problems. This convergence study is done for a set of mesh refinements and different polynomial degrees and it is evaluated at the critical point where the maximum displacement occurs. For problems subjected to a distributed load, the quantities of interest for the convergence study are displacement, internal energy and von Mises stresses. While for the problems subjected to point loads, only the displacements are evaluated, because the energy is proportional to the displacements and the stress-value under a point load will diverge, due to it being a singular point.

A total of five cases have been considered in this thesis to verify the implemented IGA element with geometric linearity. The first three cases are the shell obstacle course proposed by Belytschko *et al.* [5], where the first case is the Scordelis-Lo roof subjected to gravity load. The second case is a pinched cylinder and the third case is a pinched hemisphere, both subjected to point loads. The fourth case is a hemisphere with a cut at its pole, but with the same loading conditions as the third case [22]. The last case is a clamped hemisphere subjected to periodic loading [3]. These examples are assumed to obtain small displacements and thus geometric linearity is appropriate.

6.1.1 Scordelis-Lo roof

The first problem is the Scordelis-Lo roof, which is part of a cylindrical shell. An illustration of the problem is given in Figure 6.1, together with all relevant parameters. As seen in this figure, two of the edges are constrained in the x - and z -direction, while the two other edges are free to move. The roof is subjected to gravity load in the negative z -direction, with a magnitude of 90.0 per unit area. The maximum displacement will occur in the vertical direction at the midpoint of the free edges. According to Belytschko *et al.* [5], the reference solution of this displacement is $w = 0.3024$. This problem can be evaluated as both a full model, and a reduced model where the double symmetry of the roof is used. The reduced model is one fourth of the roof, and this is marked as the shaded area in the problem setup. In the simulation of the problem a uniform mesh is evaluated first, then a non-uniform mesh is evaluated with refinements where the maximum displacements occur, which in this case is at the boundaries. A plot of the vertical displacements is presented in Figure 6.2, where a polynomial degree $p = 4$ is used and the number of elements per edge N_{els} is set to 32.

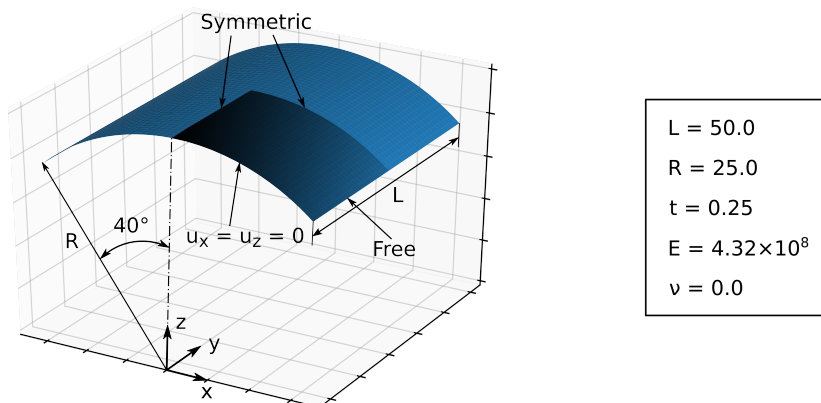


Figure 6.1: Setup for the Scordelis-Lo roof with all relevant parameters. The shaded area depicts the reduced model.

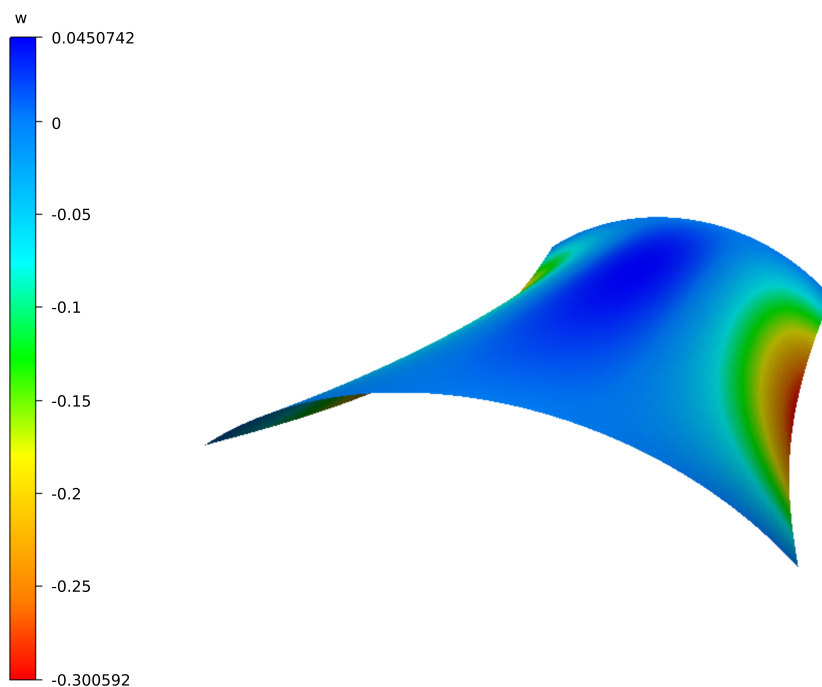


Figure 6.2: Vertical displacement w plotted on the deformed geometry of the Scordelis-Lo roof. The result is based on an analysis performed on the reduced model with a uniform mesh where $N_{els} = 32$ and $p = 4$. The deformation is scaled with a factor of 10, and the geometry has been mirrored about the x - and y -axis.

The Scordelis-Lo problem has a complex distribution of membrane strains and is therefore suitable to determine if the implemented Kirchhoff-Love shell element is able to solve such problems accurately. According to Belytschko *et al.* [5], the Scordelis-Lo problem will converge even in cases of membrane locking, but if the membrane stresses are not accurate enough the solution may not converge.

The convergence study for the Scordelis-Lo problem were evaluated at the critical point of maximum vertical displacement, which is the midpoint of the free edges. The quantities of interest for this problem are maximum vertical displacement, internal energy and maximum von Mises stress. As mentioned, a reference solution for the convergence plots were calculated from mesh extrapolation. The solutions were estimated from three simulations of the reduced model with $p = 5$ and $N_{els} = 64, 128$ and 256 . The maximum vertical displacement was estimated to $w = 0.300592457$, and the maximum von Mises stress $\sigma = 364813.55$. The internal energy is in the simulations presented as the energy norm $|u^h|$, which is related to the internal energy U in the following manner

$$U = \frac{1}{2} \int \sigma \epsilon \, dV = 2 \cdot |u^h|^2. \quad (6.2)$$

The internal energy, in the terms of the energy norm $|u^h|$, were estimated to $|u^h| = 49.125234$, for the reduced model. The internal energy of the full model will be twice the size of the reduced model.

Uniform mesh

For the convergence study with a uniform mesh, the relative error of the quantity of interest is plotted against the total number of unknowns. For all the polynomial degrees there has been a uniform mesh refinement where $N_{els} = 2^n$ for $n = 0, 1, \dots, 5$. For comparative purposes, the Scordelis-Lo roof has been analyzed in the analysis program ABAQUS, with the FEA elements: S4R5 and S9R5. For more information about the FEA elements, see Appendix C.

The convergence plots for the vertical displacement of the reduced and the full model are given in Figures 6.3 and 6.4, respectively. As seen in the plot, the IGA-based element converges faster than the FEA-based for all polynomial degrees and for $p \geq 4$ the IGA solutions are significantly better than the FEA solutions. The convergence results for the reduced and the full model are almost equal, as expected. For $p > 4$ the convergence order of the solutions does not significantly improve. The convergence plots for the internal energy and the maximum von Mises stress of both the reduced and the full model are given in Figures 6.5 - 6.8. The same observations as for

the vertical displacement applies for both these cases: it is evident that the IGA-based element converges faster than the FEA-based.

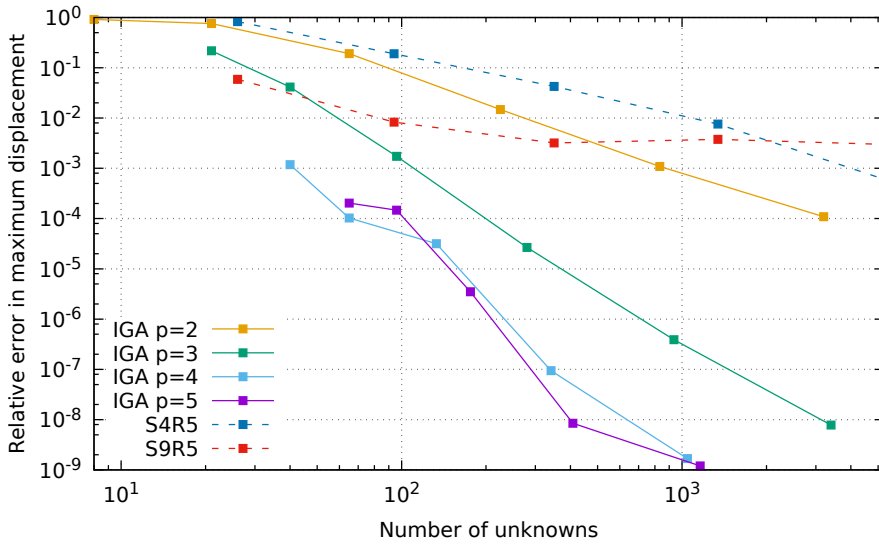


Figure 6.3: Convergence plot of the maximum vertical displacement for the reduced Scordelis-Lo roof for multiple polynomial degrees.

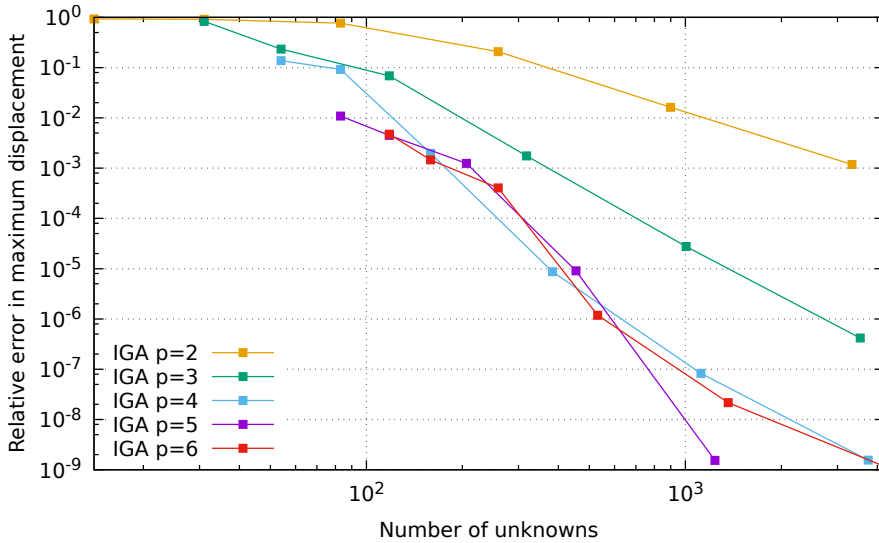


Figure 6.4: Convergence plot of the maximum vertical displacement for the full Scordelis-Lo roof for multiple polynomial degrees.

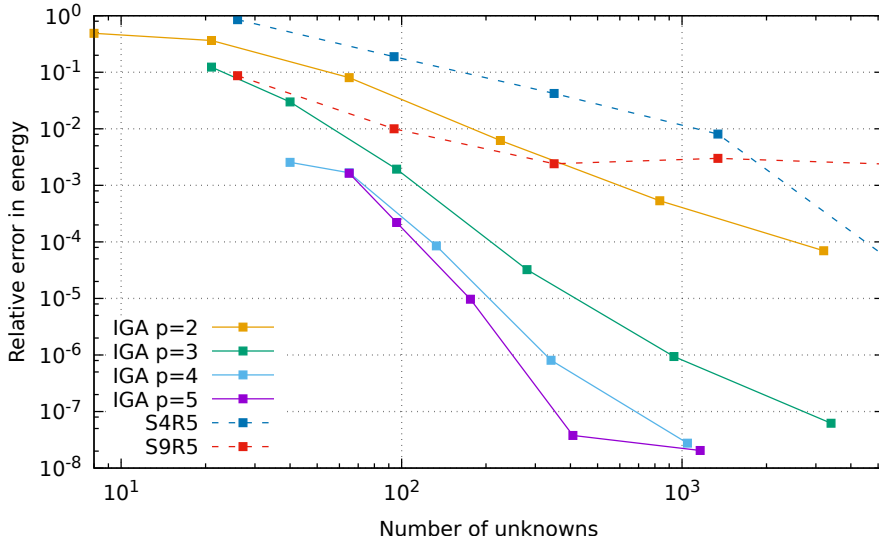


Figure 6.5: Convergence plot of the internal energy for the reduced Scordelis-Lo roof for multiple polynomial degrees.

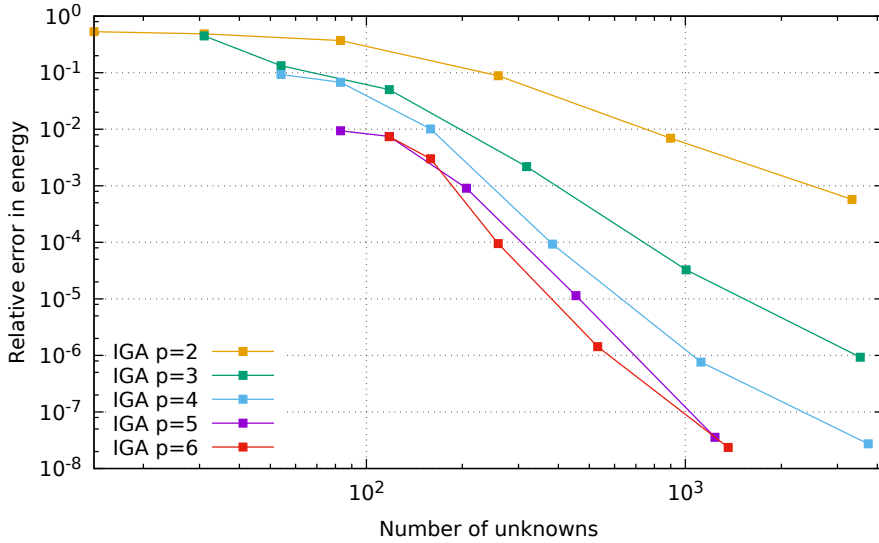


Figure 6.6: Convergence plot of the internal energy for the full Scordelis-Lo roof for multiple polynomial degrees.

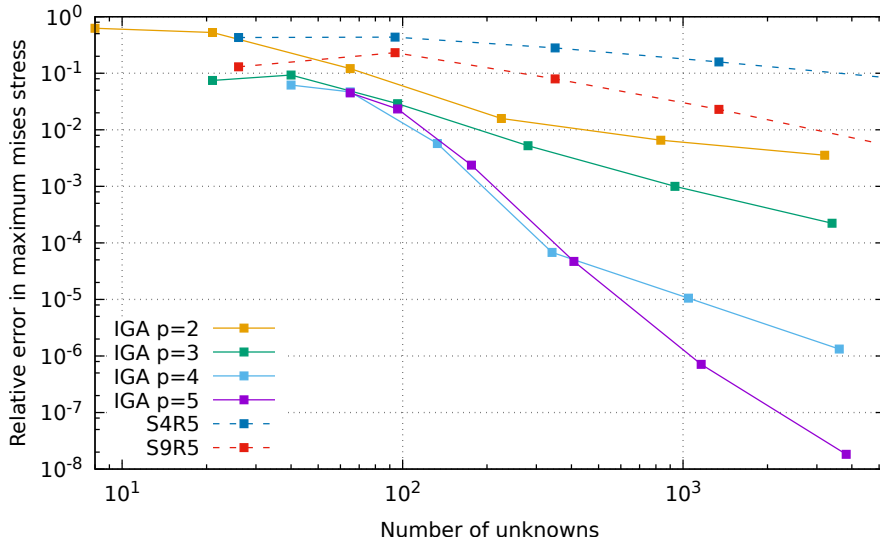


Figure 6.7: Convergence plot of the maximum von Mises stress for the reduced Scordelis-Lo roof for multiple polynomial degrees.

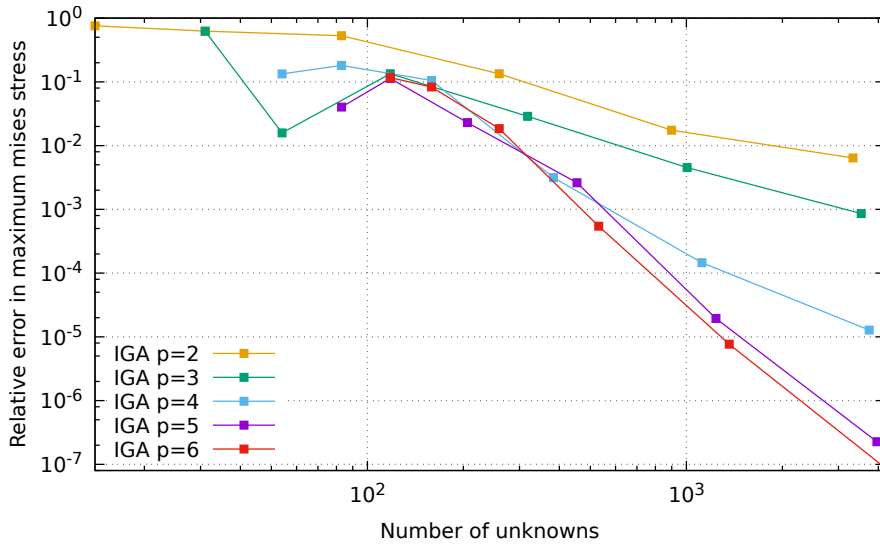


Figure 6.8: Convergence plot of the maximum von Mises stress for the full Scordelis-Lo roof for multiple polynomial degrees.

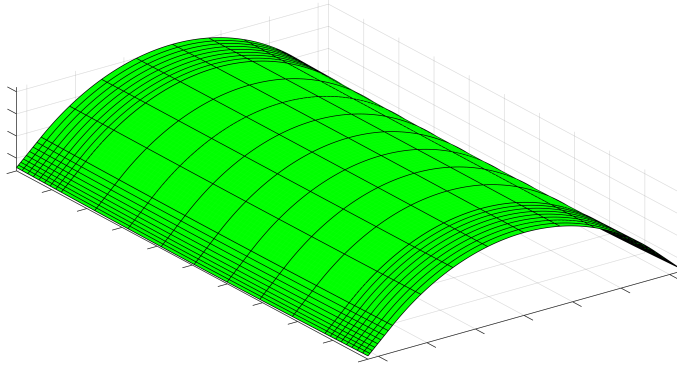


Figure 6.9: The Scordelis-Lo roof problem with a boundary refined mesh with $N_{els} = 24$

Boundary refined mesh

The Scordelis-Lo roof as a full model was also evaluated with a boundary refined mesh, where the mesh is extensively refined along its edges while the interior has a comparatively coarse mesh, as illustrated in Figure 6.9. For this thesis, the refinements were done by first inserting additional knots at u and $v = 1/8$ or $7/8$, where $u, v \in [0, 1]$ are the coordinates in the parametric space, and then uniform mesh refinements were performed.

The convergence plot for the boundary refined Scordelis-Lo roof is given in Figure 6.10, where the boundary refined mesh is compared with the uniform mesh. As seen in this figure, the boundary refined mesh does not yield faster convergence than the uniform mesh. This is an unexpected result, as the maximum displacements occur at the boundaries. The results also contradicts the findings of Kiendl *et al.* [17], where a Mindlin-Reissner element was evaluated and the analysis was based on the collocation method. The different element and analysis method may explain the contrasting results. Based on the results, the boundary refined mesh is deemed to be inferior to a uniform mesh. Therefore, the boundary refined mesh method will not be pursued any further in this thesis.

Stress distributions

In order to verify the implemented stress recovery described in Section 4.3.1, both the membrane stresses, denoted n_{xx} , and the bending moment, denoted m_{xx} , are evaluated for several polynomial degrees and different meshes, and then compared with the results obtained by Kiendl [15].

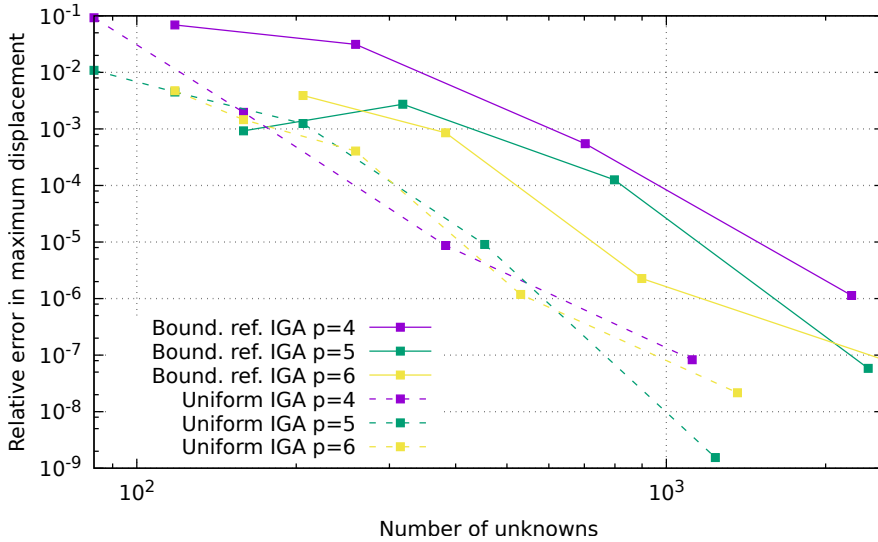


Figure 6.10: Convergence plot of the relative error of the maximum displacement for the boundary refined Scordelis-Lo roof for multiple polynomial degrees plotted together with the results from the uniformly refined Scordelis-Lo roof.

Figure 6.11 presents three plots of the membrane stresses with various meshes, denoted in the figure description. The plots clearly show that the values converge towards the distribution found by Kiendl [15], even for these rather coarse meshes. The three plots in this figure have two visualization points uniformly distributed per element in each parameter direction. However, if the number of visualization points are raised to four, the plots of n_{xx} obtain an oscillating pattern, while the L_2 -projection of the stresses results in an expected distribution, as seen in Figure 6.12. An explanation for these oscillations is the fact that a uniform distribution of visualization points is not necessarily an optimal distribution in terms of where the solution is the most exact. Oscillations were found for several meshes with four visualization points.

Plots of the bending moment m_{xx} are presented in Figure 6.13 with the same meshes as for the membrane stresses. The finest mesh clearly shows a convergence towards what was presented by Kiendl [15]. Even though the two other meshes also have the same pattern, they are not symmetric. This is due to the polynomial degree being $p = 2$, and at this polynomial degree the derivative of the evaluated knot is constant and may yield a different value depending on which side of the knot the derivative is evaluated.

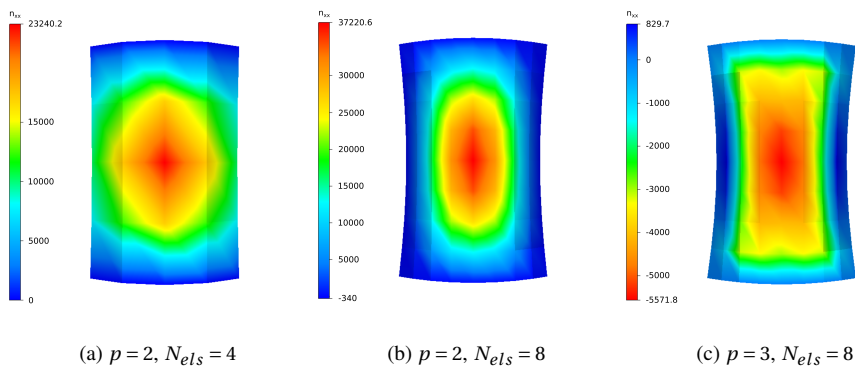


Figure 6.11: Membrane stresses, n_{xx} , plotted on the deformed geometry for three different meshes, in the xy -plane. The plots have two visualization points per element in each parameter direction. The deformation is scaled with a factor of 10.

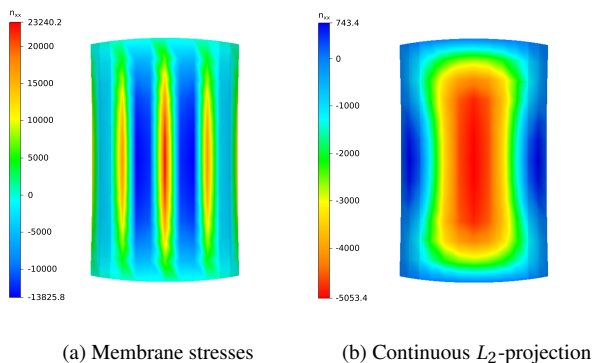


Figure 6.12: Membrane stresses, n_{xx} , plotted with four visualization points per element in each parameter direction. The deformation is scaled with a factor of 10. Here, a mesh with $N_{els} = 4$ and $p = 2$ was used.

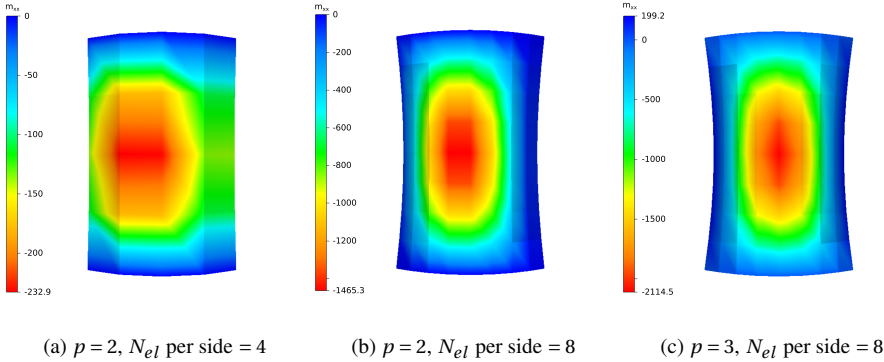


Figure 6.13: Bending moment, m_{xx} , plotted on the deformed geometry for three different meshes, in the xy -plane. The plots have two visualization points per element in each parameter direction. The deformation is scaled with a factor of 10.

6.1.2 Pinched cylinder

The second problem of the shell obstacle course is a pinched cylinder. Both edges of the cylinder is constrained by a rigid diaphragm and it is subjected to two point loads opposite of each other, as illustrated in Figure 6.14. The critical points of interest are located under the point loads, but because of symmetry only one of these are regarded. The reference solution of the vertical displacement in this point is $w = 1.8248 \cdot 10^{-5}$, according to Belytschko *et al.* [5]. The simulation of this problem is performed on a reduced model, where only one eighth of the model is considered due to symmetry. A full model of this problem is not possible without adding capabilities to assume convergence, because of the challenge circle segments with $\theta \geq 180^\circ$ introduces in a NURBS setting, which has been discussed in section 2.2.4. The deformed geometry is presented in Figure 6.15. Here, the reduced model has been mirrored about two of the symmetry boundaries, such that half of the cylinder is shown.

The convergence study for the pinched cylinder examines only the convergence of the maximum vertical displacement, as mentioned. The reference solution calculated by Richardson's extrapolation is $w = 1.828195 \cdot 10^{-5}$. Figure 6.16 plots the relative error in maximum displacement versus number of unknowns, which shows how the IGA element converges compared to two FEA elements evaluated in ABAQUS, namely S4R5 and S9R5. As seen in this plot, the IGA element converges faster than the FEA elements for all polynomial degrees. The second plot of the convergence study, Figure 6.17, plots the relative error in maximum displacement versus number

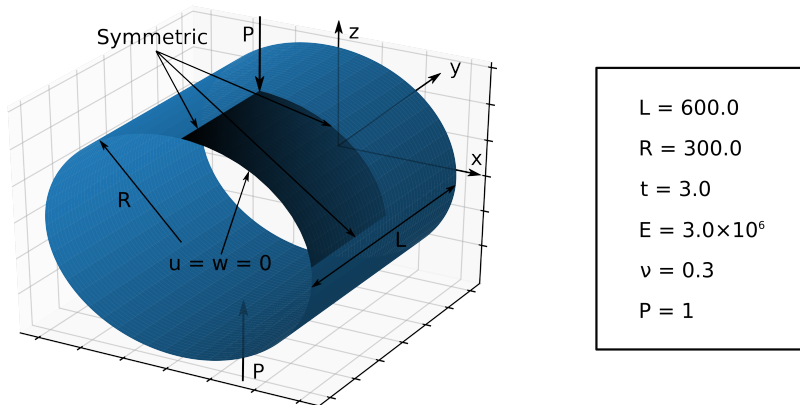


Figure 6.14: Setup for the pinched cylinder with all relevant parameters. The shaded area depicts the reduced model

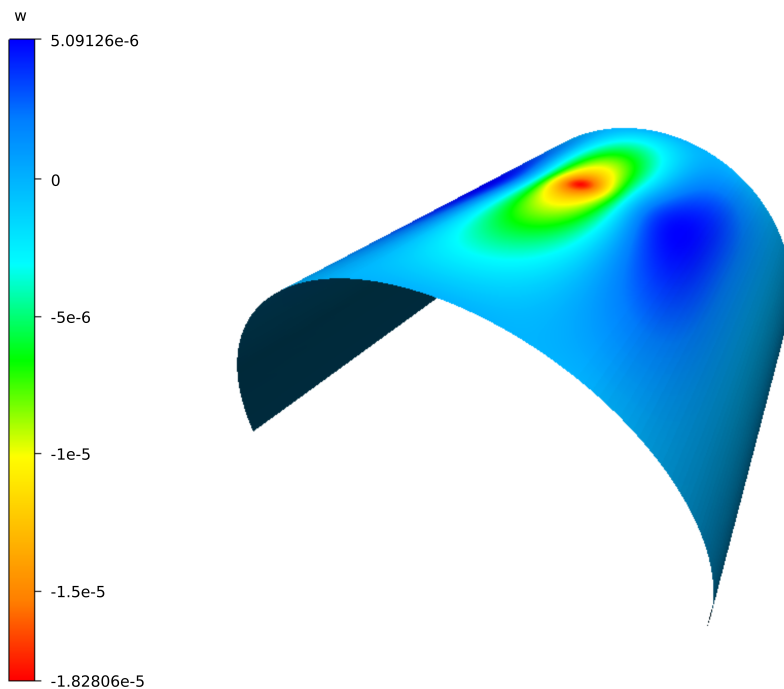


Figure 6.15: Vertical displacement w plotted on the deformed geometry of half the pinched cylinder, which is based on an analysis with $N_{els} = 64$ and $p = 6$.

of elements per side, which also present the convergence of the higher polynomial degrees (up to $p = 9$). Even though higher polynomial degrees yield a slightly faster convergence, the profits gained passes a point of diminishing returns. Which in this case, applies to $p \geq 5$. The reason for this is that the convergence for this problem is monitored by the singular points, while theoretical convergence is only obtained for highly regular problems.

6.1.3 Pinched hemisphere

The hemispherical shell is the third and last problem in the shell obstacle course proposed by Belytschko *et al.* [5]. The problem setup and all relevant parameters are presented in Figure 6.18. The hemisphere is subjected to four opposing radial point loads along its circumferential edge. The boundary conditions are modelled as free along the circumferential edge, and fixed at the hemisphere's pole. The primary unknown for this example is the radial displacements at the loaded points, and Belytschko *et al.* [5] reports the reference solution to be $u = 0.0924$ and $v = -0.0924$ at these points.

This is chosen as a benchmarking example as it is a challenging case. The hemispherical shell tests the element's ability to handle rigid body rotations as relatively large sections of the hemisphere rotates in a fashion resembling that of the rigid body rotations. As the hemisphere exhibits close to no membrane stresses it further complicates the problem as this will require the element to represent inextensional modes. The resulting deformed geometry is presented in Figure 6.19 with a contour plot of the total deformation $u_{tot} = \sqrt{u^2 + v^2 + w^2}$.

In order to evaluate the element's performance in terms of convergence it is here sufficient to track only the displacement. This is due to the energy measurement will be proportional to the displacement, and the stresses will diverge towards infinity because of the point loads. The convergence study is conducted with a reference solution of the deformation in x -direction determined to $u = 9.2413089 \cdot 10^{-2}$, which is found with multimesh extrapolation. All results in this section are derived from the reduced model.

The convergence plot presented in Figure 6.20 shows the differences in convergence rate for the element for different polynomial degrees. From the plot it becomes clear that the only significant change in convergence rate is between $p = 2$ and $p = 3$ for this case. The convergence rate is approximately identical for $p \geq 3$, and the results generated with $p > 6$ yielded no improvement and was thus neglected. However, it is worth noting that even though there is clear diminishing return in terms of accuracy when the unknowns are increased, for equal numbers of unknowns, higher polynomial degrees accomplish slightly better results.

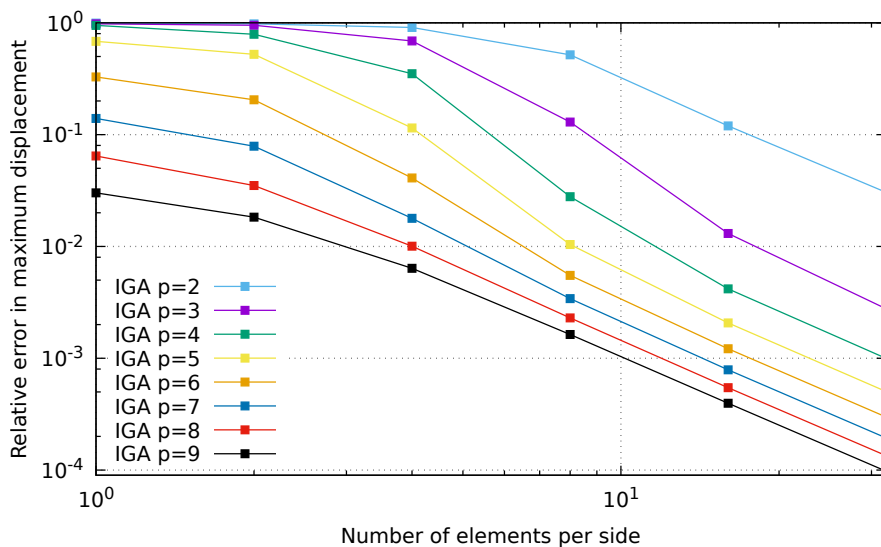


Figure 6.17: Convergence plot of the displacement versus number of elements per edge for the pinched cylinder for multiple polynomial degrees.

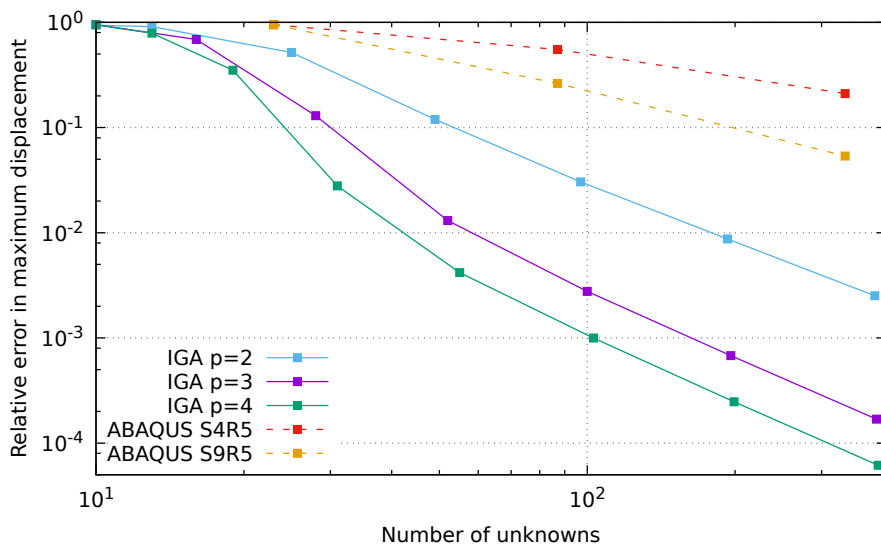


Figure 6.16: Convergence plot of the displacement versus number of unknowns for the pinched cylinder for multiple polynomial degrees.

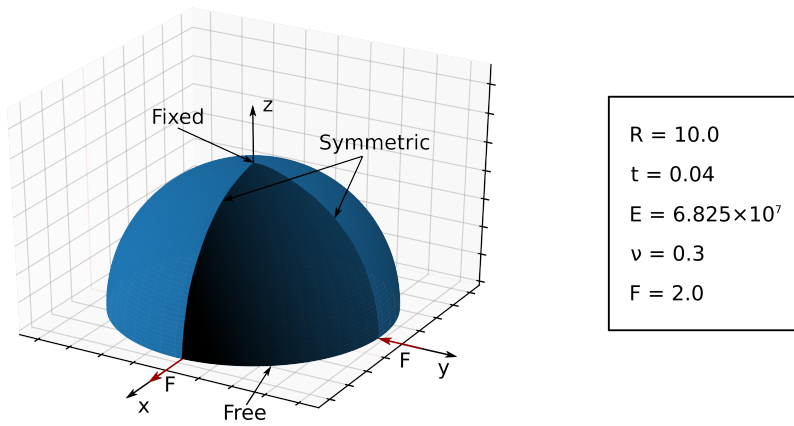


Figure 6.18: Setup for the hemispherical shell with all relevant parameters. The shaded area is the reduced model which all results are generated from.

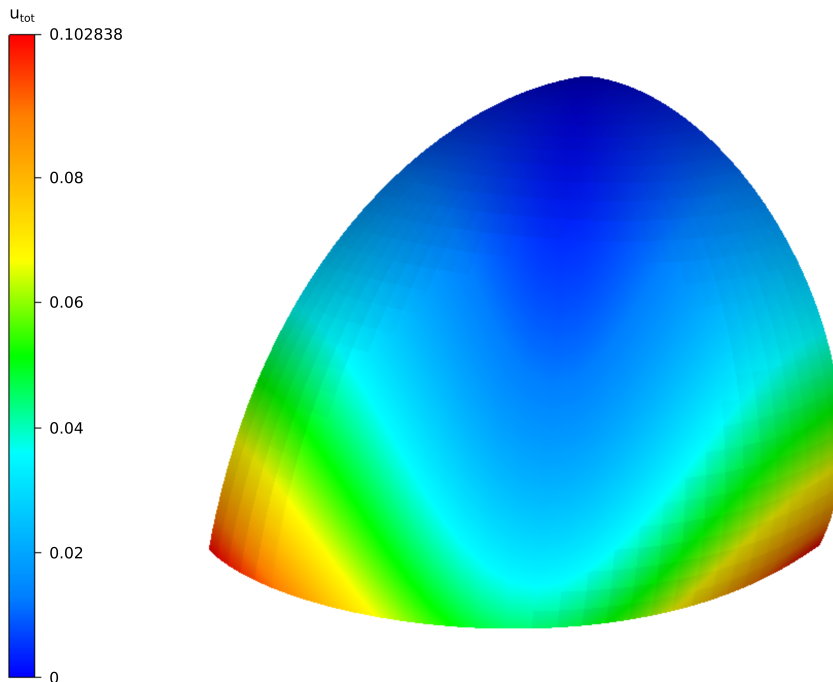


Figure 6.19: Deformed geometry for the hemispherical shell scaled with a factor of 20. The contour plot tracks the total deformation u_{tot} . Only the reduced model is shown. A mesh with $N_{els} = 32$ and $p = 4$ is used to generate the plot.

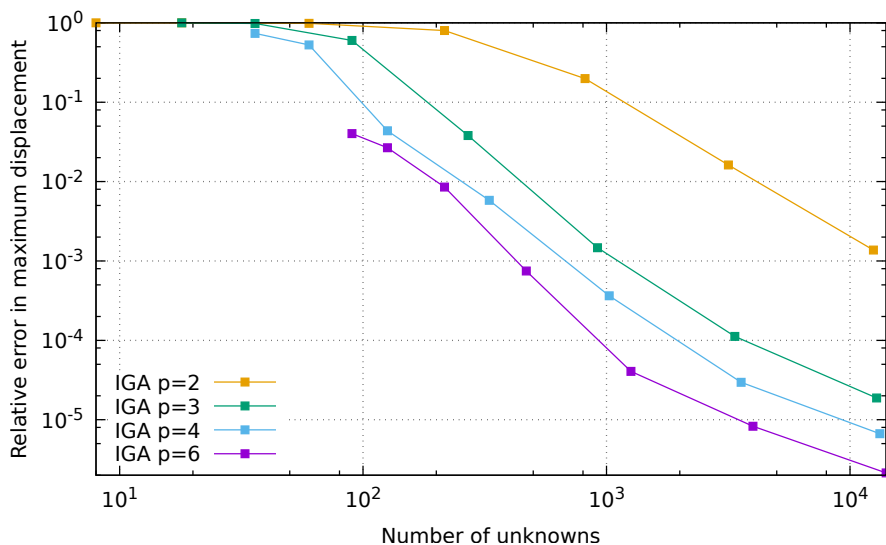


Figure 6.20: Convergence plot of the maximum displacement in x -direction for the reduced model of the pinched hemisphere for multiple polynomial degrees.

6.1.4 Pinched hemisphere with cut

The hemisphere with a 18° cut at the top is another common benchmark example. The problem setup and all relevant parameters are presented in Figure 6.21. The hemisphere is subjected to the same loads as the former example (Section 6.1.3), and the boundary condition along the circumferential edge is free. Through applying symmetry boundaries and restraining one of the corners from vertical displacement, all rigid body motions are suppressed. This example exhibits the same challenges as the full hemisphere, but is a reoccurring example as the cut often circumvents the necessity of triangular mesh segments around the axis of revolution [19][18]. Like the full hemisphere, the primary unknown is the displacement at the points where the loads are applied. With the Richardson's extrapolation method a reference solution of the displacement in x -direction were estimated to $u = 9.3019296 \cdot 10^{-2}$. Once again it is sufficient to only examine the displacement of this example to evaluate the element's performance. The resulting deformed geometry is presented in Figure 6.22 with a contour plot of the total deformation.

Figure 6.23 presents the convergence plot of the hemisphere with cut, where the relative error in maximum displacement is plotted against the number of unknowns. From this plot no new conclusions can be made.

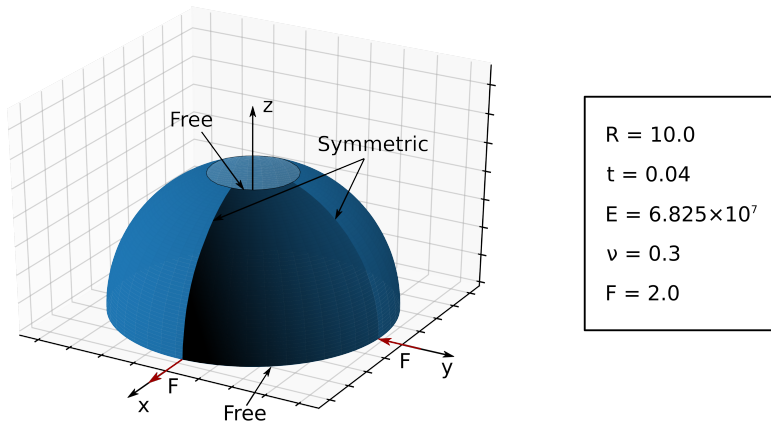


Figure 6.21: Setup for the pinched hemisphere with a cut at its pole with all relevant parameters. The shaded area depicts the reduced model which all results are generated from.

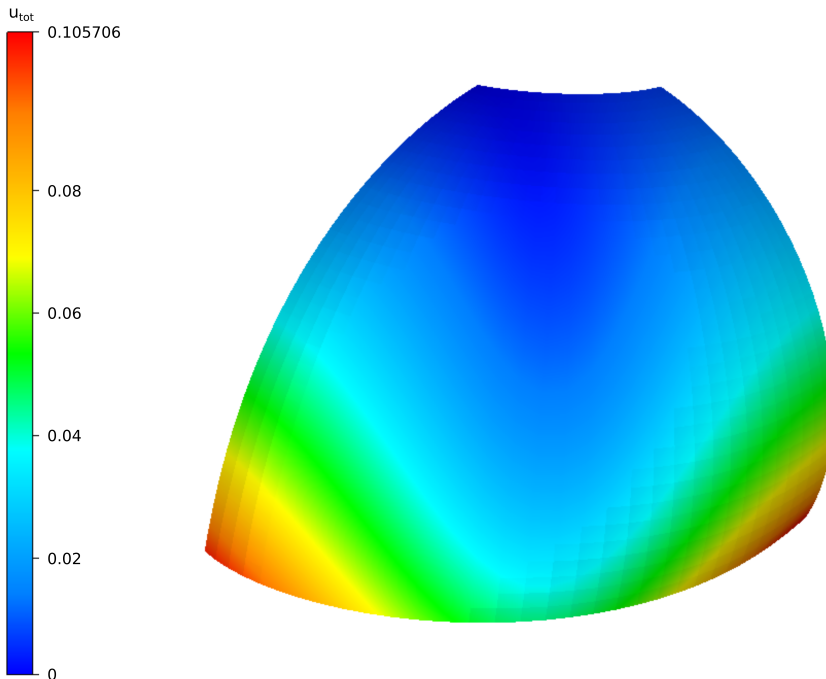


Figure 6.22: Deformed geometry for the hemispherical shell with cut scaled with a factor of 20. The contour plot tracks the total deformation u_{tot} . Only the reduced model is shown. A mesh with $N_{els} = 32$ and $p = 4$ is used to generate the plot.

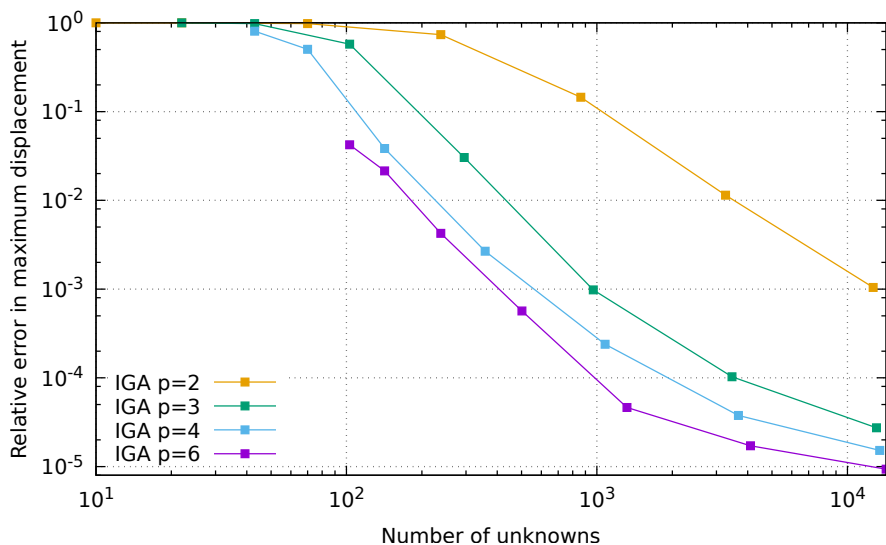


Figure 6.23: Convergence plot of the maximum displacement in x -direction for the reduced model of the pinched hemisphere with cut for multiple polynomial degrees.

6.1.5 Clamped hemisphere with periodic loading

The clamped hemisphere with periodic loading is the last benchmark example used to verify the linear IGA element. All analyzes of this problem are performed by using the reduced model, with the same symmetry conditions as for the other hemispherical benchmarks. The problem setup and all relevant parameters are presented in Figure 6.24. The critical point of interest for this problem is the vertical displacement at the pole of the hemisphere. This example exhibits the same set of challenges as the hemisphere in Section 6.1.3, but differs in one important way: here the hemisphere is exposed to a distributed periodic load. This alteration cures the problem for singular points, thus allows for a convergence rate free of affections from singularities. The resulting deformed geometry is displayed in Figure 6.25.

In order to evaluate the element's performance, it will suffice for this example to study the vertical displacements and the internal energy at the pole. The reference solutions used in the convergence study are the ones reported from Bathe *et al.* [3]: $w = -8.19934 \cdot 10^{-6}$ and $U = 7.88885 \cdot 10^{-4}$. In Figures 6.26 and 6.27 the convergence study for the displacement is presented for the different polynomial degrees $p = 2, \dots, 6$. The convergence plots show a higher convergence rate compared to examples where singularities are present. In Figure 6.28 the convergence study w.r.t. the internal energy study is presented, where the highest polynomials exhibit one impor-

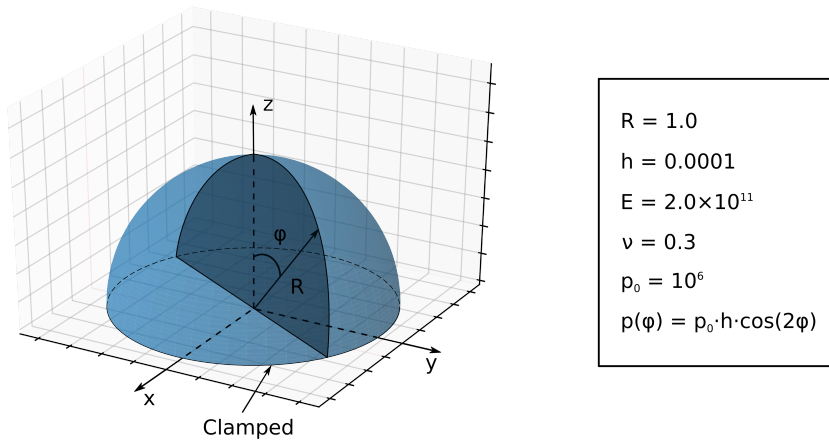


Figure 6.24: Setup for the clamped hemisphere with periodic loading with all relevant parameters.

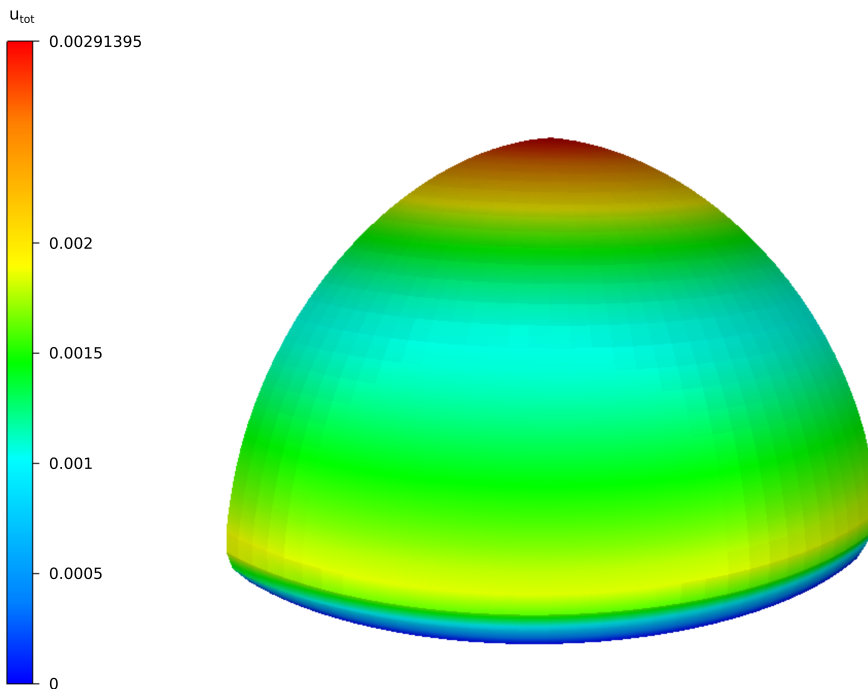


Figure 6.25: Deformation plot of the clamped hemisphere, where the contour plot tracks the total deformation u_{tot} . The plot is based off an analysis with $N_{els} = 32$ and $p = 4$ with the reduced model. To visualise the deformations, a scaling factor of 250 has been applied.

tant property: they achieve a quadratic convergence rate earlier, which makes their performance significantly better in terms of n_{els} .

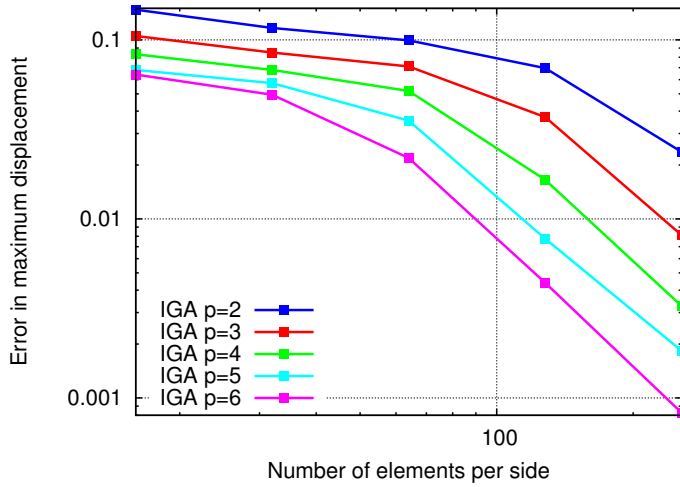


Figure 6.26: Convergence plot of the displacement results in terms number of elements per side. The highest polynomials perform best, as expected.

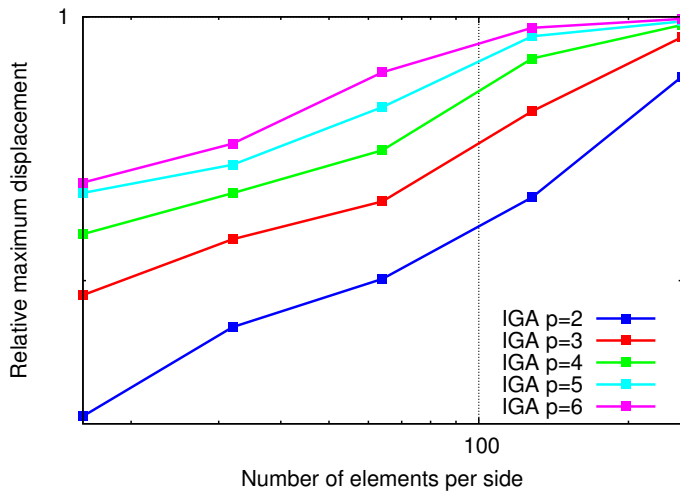


Figure 6.27: Convergence plot of the relative maximum displacement plotted against number of element per side, the highest polynomials perform best, and the lowest polynomial need an extravagant amount of elements per side in order to have an error less than 10 %.

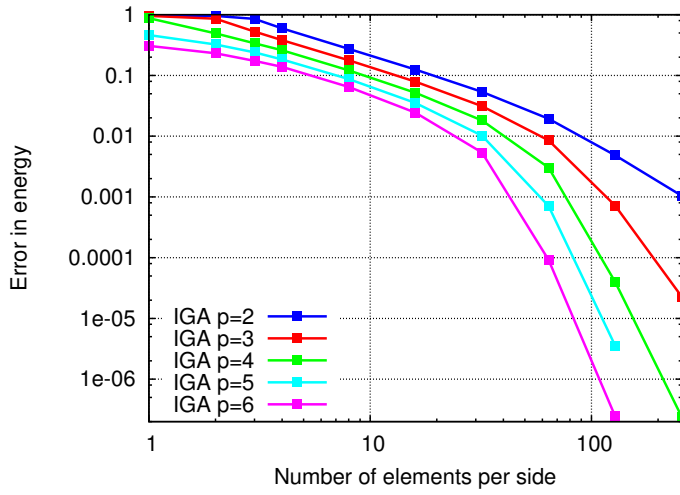


Figure 6.28: Convergence plot of the error in energy plotted against the number of elements per side. Here, the convergence rate is in accordance with the theory as no singularities are present.

6.2 Benchmark examples with geometric nonlinearity

To verify the implemented Kirchhoff-Love shell element with nonlinearity, some of the benchmark examples proposed by Sze *et al.* [25] are tested in this section. A total of three cases are studied: a cantilever subjected to a line load, a pinched cantilever cylinder with prescribed deflection and a hemispherical shell subjected to point loads. All these benchmark examples are assumed to obtain large displacements and thus a nonlinear analysis must be performed. To verify the nonlinear implementation the membrane stresses and bending moment of the pinched cantilever cylinder are compared to a FEA model made in the analysis program ABAQUS.

6.2.1 Cantilever beam

The first nonlinear example is a cantilever beam subjected to a line load with a magnitude of -100 per length unit in the vertical direction. The beam geometry has the dimensions of $L_x = 2$ and $L_y = 1$, and is restrained from rotation about the y -axis on one of the short edges. The material parameters are defined as the following: $E = 1.0 \cdot 10^7$, $\nu = 0.0$, and $t = 0.05$. In this example the cantilever beam will be subjected to both bending and membrane strains, and a quadratic convergence rate w.r.t. the NR iterations is expected.

In the load-stepping curve displayed in Figure 6.29, both the displacement in x and y is tracked as a function of the load steps in terms of P/P_{max} , where P is the incremental load. The figure tracks the absolute displacement, and as can be seen in the deformation plot presented in Figure 6.30, the displacements occur in the negative z -direction and positive x -direction. The results presented from the analysis gives the minimum bending stress at the tip, and maximum at the fixed end, which irrefutably will be the case.

The Figure 6.29 is based off the results presented in Table 6.1, where the number of NR iterations are included. For each load step, the analysis has performed multiple NR iterations, and for this example the convergence criterion was set equal to $1.0 \cdot 10^{-8}$. The convergence plot is displayed in Figure 6.31, where the convergence rate approaches a quadratic rate in the last iterations, as expected. The plot also shows that the first load steps require more iterations than the later ones. This can be explained by examining the two Figure 6.29 which show that the equilibrium path has a steeper curve at the end, which in turns gives a more efficient NR method.

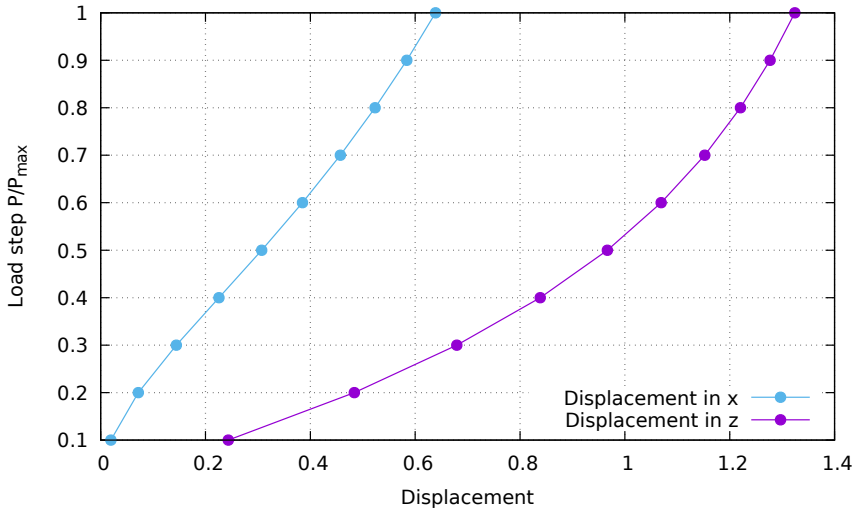


Figure 6.29: The displacements in the x - and y -direction for each load step. Note that the figure tracks the absolute displacements. The monotonic equilibrium path is successfully tracked with the normal NR method.

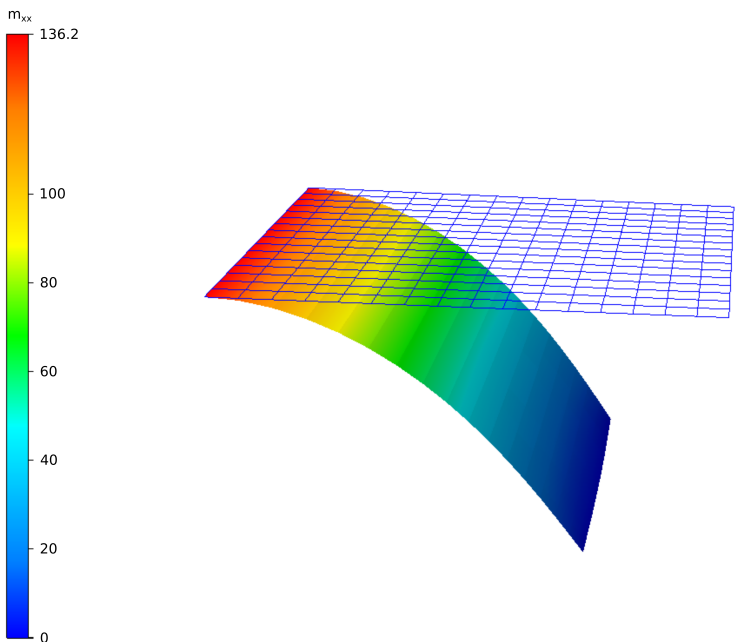


Figure 6.30: Deformed geometry of the cantilever beam. The contour plot shows the bending stresses plotted onto the deformed geometry. The plot is based on a mesh with $N_{els} = 16$ and $p = 4$. The reference configuration is shown with the blue mesh. The deformations are not scaled.

Table 6.1: The results from step 10 at $t = 1.0$ of the nonlinear analysis of the cantilever beam. 5 iterations was necessary to meet the convergence criterion of $1.0 \cdot 10^{-8}$. The simulation was run with $N_{els} = 32$ and $p = 4$.

Iteration	ϵ_i	ΔE	R_{norm}	ΔU_{norm}
0	1.998e-01	5.150e-01	7.071e+00	1.488e+00
1	3.833e-01	9.880e-01	1.193e+02	3.712e-02
2	4.760e-04	1.227e-03	2.680e-01	6.506e-02
3	2.586e-06	6.664e-06	3.699e-01	5.444e-04
4	4.197e-12	1.082e-11	3.270e-05	5.792e-06

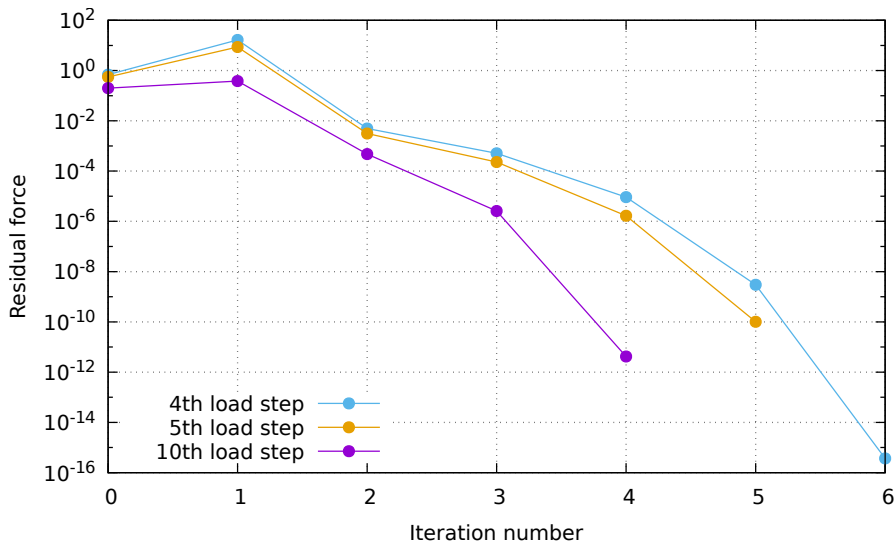


Figure 6.31: The convergence results from the NR iterations from selected load steps from the mesh with $N_{els} = 32$ and $p = 4$. As can be seen, a quadratic convergence rate is achieved.

6.2.2 Pinched cantilever cylinder

The second nonlinear benchmark of this thesis is the pinched cantilever cylinder. The problem setup is illustrated in Figure 6.32, where only half of the cylinder is shown. In the problem setup the cylinder is subjected to two point loads opposite of each other on the free end. In order to improve the convergence, the simulation of the problem will instead have a prescribed vertical deflection of $w = -1.6R$ where the load is placed. Only a quarter of the cylinder is used to simulate this problem, due to the symmetry and because the deformations go beyond the radius of the cylinder. Thus, it is not physically possible, but it is still a relevant theoretical benchmark example [22]. A contour plot of the vertical deformation is given in Figure 6.33, where the nonlinear deformations are clearly shown.

To verify that the implementation of the nonlinear Kirchhoff-Love shell element is correct, an ABAQUS model of the same problem has been evaluated for comparison. The ABAQUS model was compiled with the FEA element S4R, which is described in Appendix C. A comparison of the membrane stresses n_{xx} is given in Figure 6.34. Here, the L_2 -projection of the IGA solution is given next to the FEA solution established in ABAQUS. The two plots in the figure clearly shows that the two cases have similar distributions. In order to compare the two plots, the legend-values has been adjusted so that they show the same range, because of the singular point under the load. Figure 6.35 and 6.36 depicts the same two models (IGA versus FEA), but with the bending moment distribution m_{xx} and the von Mises stresses σ_v , respectively. Also, the plots show the same distributions. By these three figures, the implementation can be assumed to be correct.

Figure 6.37 presents a plot of the prescribed normalized vertical displacement w/R versus the reaction force R at the tip of the cylinder, where a mesh with $N_{els} = 32$ are evaluated with three different polynomial degrees, $p = 2, 3, 4$. As seen in this plot, the three cases are indistinguishable from each other, hence convergence can be assumed. Table 6.2 presents the NR iterations of the last load step of the simulation with $p = 3$. This plot yields a reaction force of $R_z = 840.6$ for the IGA model, whereas the FEA model has a reaction force of $R_z = 827.4$ at negative vertical direction at the midpoint of the tip of the cantilevered cylinder.

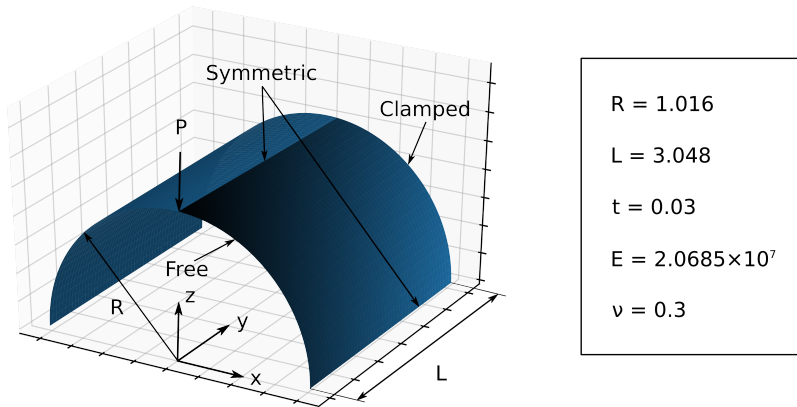


Figure 6.32: Setup for the pinched cantilever cylinder with all relevant parameters. Only half of the cylinder is illustrated. The shaded area depicts the reduced model used in simulations.

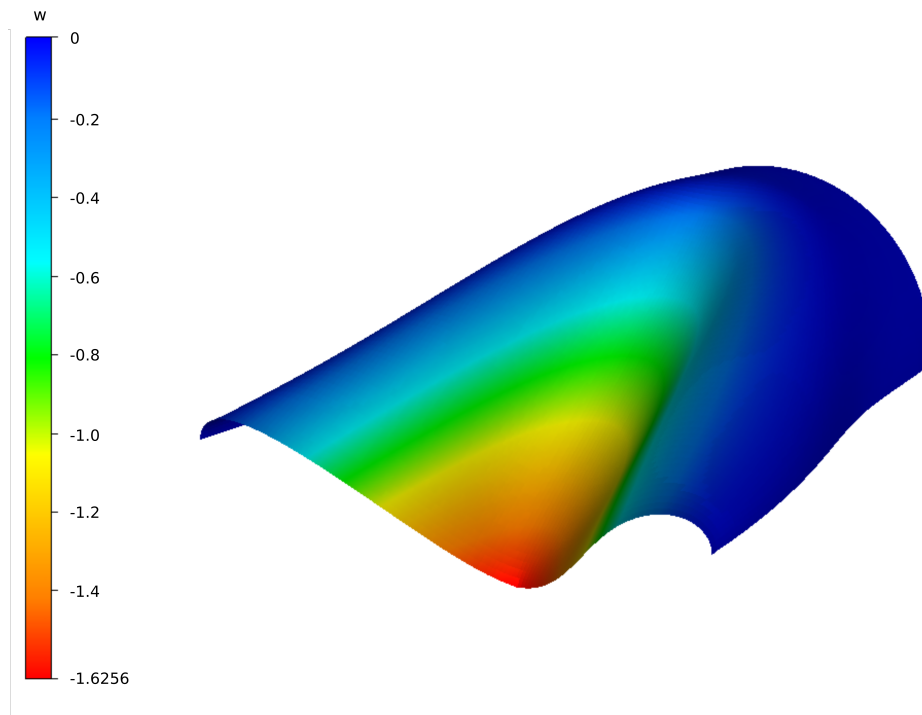


Figure 6.33: Deformed geometry for the pinched cantilever cylinder. The contour plot tracks the vertical displacement w .

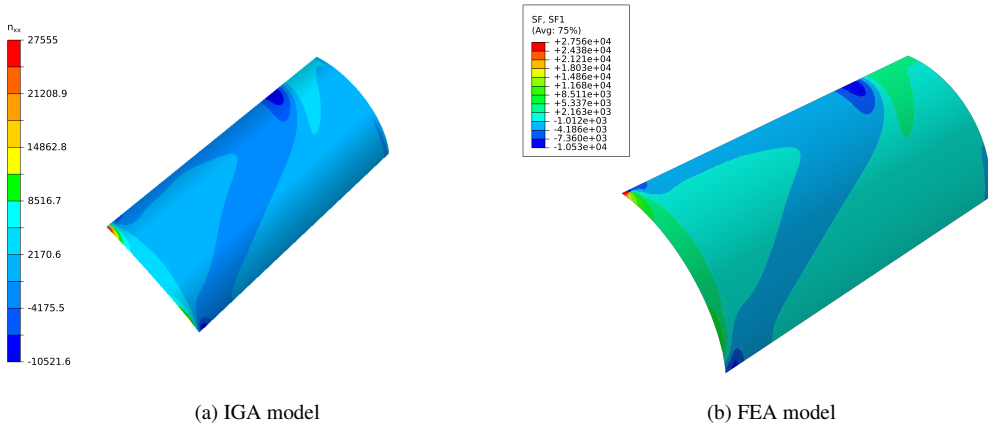


Figure 6.34: A comparison of the membrane stresses n_{xx} of the IGA model and the FEA model made in ABAQUS. The legend range of the IGA model has been adjusted for visualization purposes.

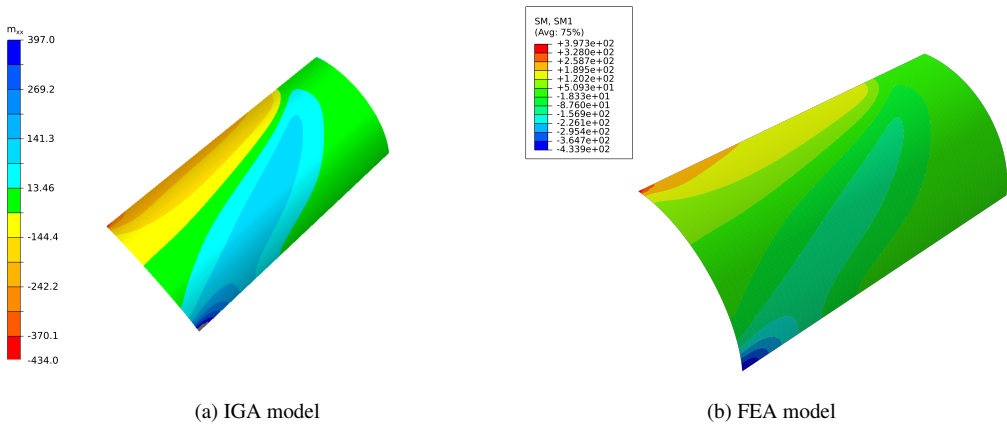


Figure 6.35: A comparison of the bending moment m_{xx} of the IGA model and the FEA model made in ABAQUS. The legend range of the IGA model has been adjusted for visualization purposes.

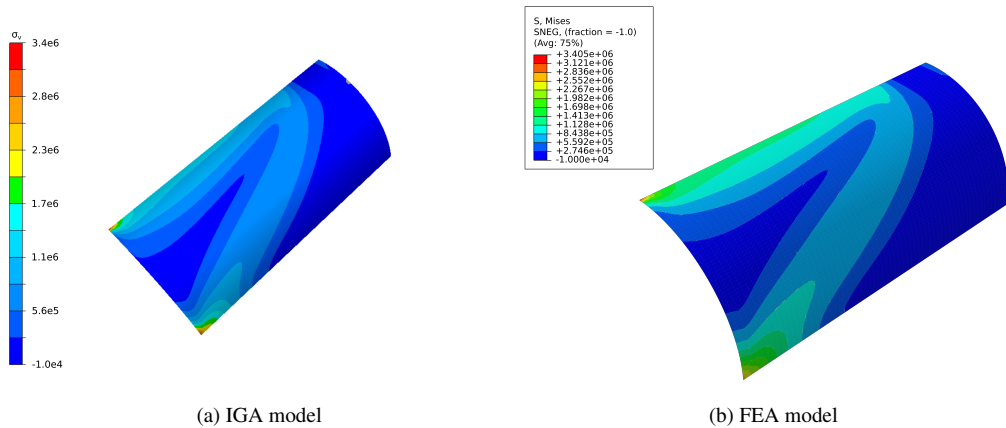


Figure 6.36: A comparison of the von Mises stress σ_v of the IGA model and the FEA model made in ABAQUS. The legend range of the IGA model has been adjusted for visualization purposes.

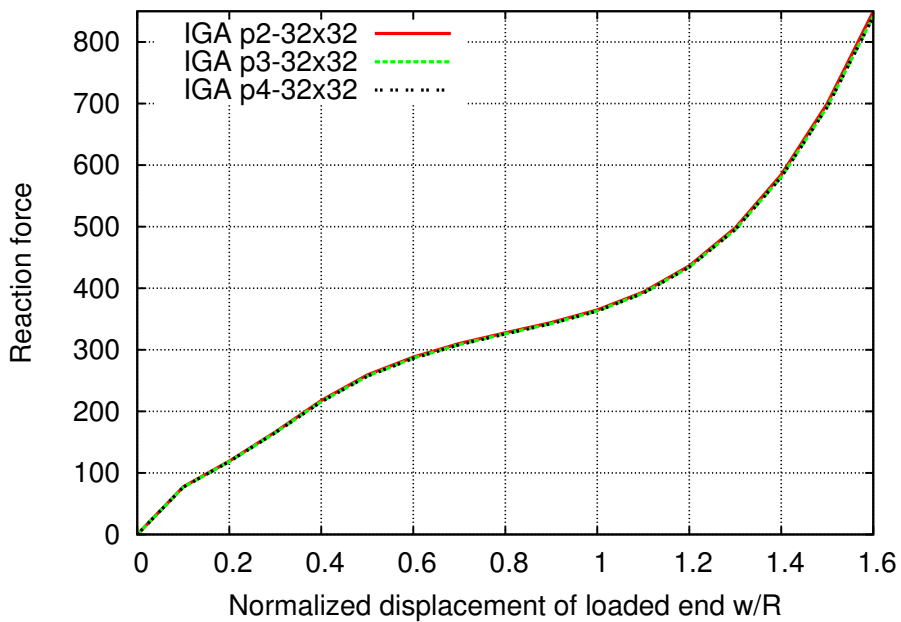


Figure 6.37: Normalized displacement plotted against the reaction force at the tip of the cantilever cylinder. For all three polynomial degrees a mesh of 32 elements per edge is used.

Table 6.2: The results from the NR iterations at step 16 at $t = 1.6$ of the nonlinear analysis of the pinched cantilever cylinder. 10 iterations was necessary to meet the convergence criterion of $\epsilon = 1.0 \cdot 10^{-16}$.

Iteration	ϵ_i	ΔE	R_{norm}	ΔU_{norm}
0	1.000e+00	5.500e+03	3.713e+04	2.247e+00
1	1.830e-01	1.006e+03	1.301e+04	2.807e-01
2	5.404e-03	2.972e+01	1.338e+03	1.194e-01
3	5.200e-04	2.860e+00	3.455e+02	9.064e-02
4	4.534e-05	2.494e-01	7.471e+01	1.067e-01
5	1.524e-05	8.382e-02	8.959e+01	4.433e-02
6	3.745e-06	2.059e-02	1.477e+01	4.769e-02
7	3.826e-07	2.104e-03	1.581e+01	8.213e-03
8	4.643e-09	2.553e-05	4.799e-01	1.873e-03
9	8.621e-13	4.741e-09	2.404e-02	1.308e-05
10	2.753e-20	1.514e-16	1.197e-06	4.579e-09

6.2.3 Hemispherical shell with cut

The hemispherical shell with a cut from Section 6.1.4 is chosen as the final benchmark case for the nonlinear implementation. The geometrical properties remain the same, and the following parameters has been changed: $E = 6.825 \cdot 10^7$, $t = 0.04$ and $F = 400$. As the forces are applied along the x and y axis, the deformations of main interest will be the radial deformations at the points where the load is applied, that is u and v .

This example is a doubly curved surface with the same set of challenges as before, but it will now be subjected to forces with such a magnitude that large deformations are inevitable. This example is a prevalent example for nonlinear analysis, and Sze *et al.* [25] reports the following displacement results at the points of the loads: $u = 4.067$ and $v = -8.178$.

The analysis has been performed on the reduced model with a mesh of 16 elements per edge, and three different polynomial degrees: $p = 3, 4, 5$. As this example will produce large deformations, 20 load steps and a convergence criterion of $1.0 \cdot 10^{-14}$ has been employed to prevent the analysis from diverging. The resulting displacements as a function of the load steps are presented in Figure 6.38, and the final displacements of the points where the loads were applied are as follows: $u = 4.073$ and $v = -8.147$. The presented displacements are generated with a poly-

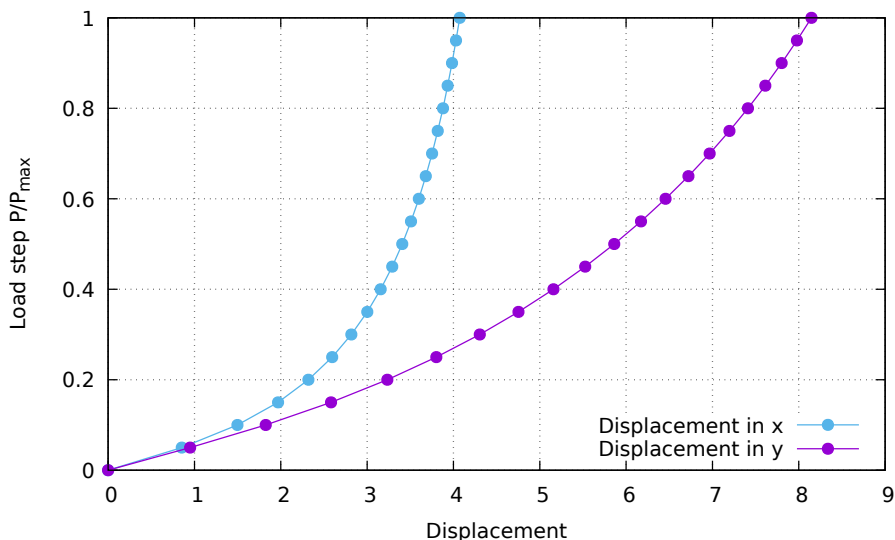


Figure 6.38: The displacements of the two points for each load step for the hemispherical shell with cut. A polynomial degree of $p = 5$ is used to generate the figure. Note that the figure tracks absolute displacements.

nomial degree of $p = 5$, but for this mesh there was no significant difference between the results generated with the three different polynomial degrees. The results obtained with $p = 4, 5$ yielded results which differed with less than 0.5% from what Sze *et al.* [25] reports, and $p = 3$ performed slightly worse with a discrepancy of 2%. The deformed geometry is presented in Figure 6.39, where undeformed geometry is displayed with the blue mesh.

The strict convergence criterion of $1.0 \cdot 10^{-14}$ was for this example sufficiently rapidly met as the NR iterations achieved a quadratic convergence rate, as can be seen in Table 6.3. The maximum amounts of iterations needed were at the 4th load step, which required eight iterations in order to fulfil the criterion. As only one of the 20 load steps had 8 iterations the number of load steps needs not to be increased.

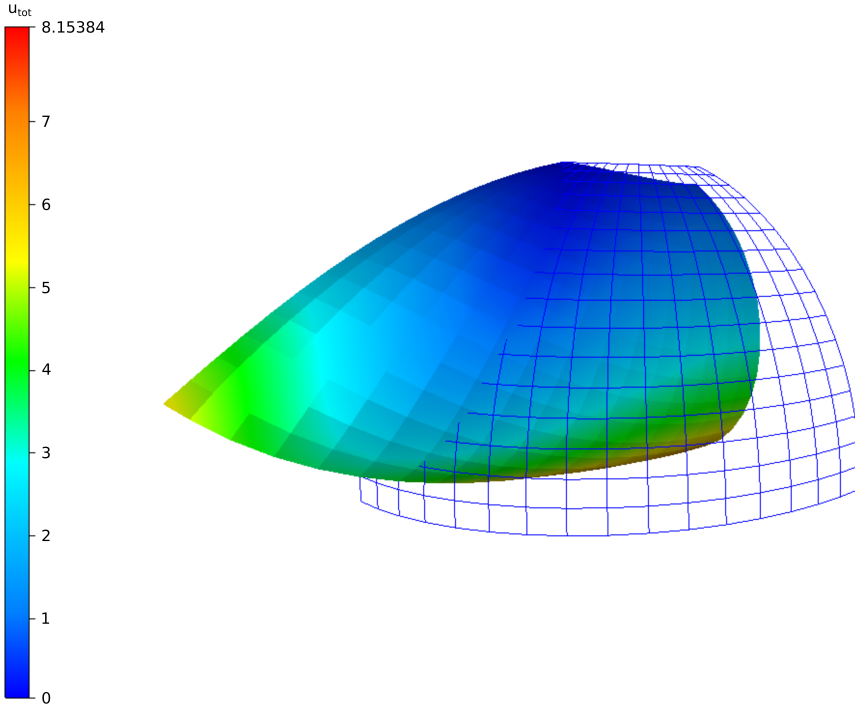


Figure 6.39: The undeformed and deformed geometry. The model is based off the analysis performed with $N_{els} = 16$ and a polynomial degree $p = 5$.

Table 6.3: The results from step 20 at $t = 1.0$ of the nonlinear analysis of the cantilever beam. 5 iterations were necessary to meet the convergence criterion of $\epsilon = 1.0 \cdot 10^{-14}$.

Iteration	ϵ_i	ΔE	R_{norm}	ΔU_{norm}
0	1.161e-01	4.343e+00	2.000e+01	1.702e+00
1	1.034e+00	3.867e+01	6.807e+03	1.835e-02
2	3.430e-05	1.283e-03	5.958e+00	3.316e-02
3	2.827e-07	1.058e-05	3.224e+00	8.131e-04
4	4.008e-12	1.499e-10	1.865e-03	1.227e-05
5	4.856e-21	1.816e-19	4.224e-07	1.090e-10

Chapter 7

Conclusion

In this thesis, the IGA shell element has been implemented in an object-oriented computing environment, namely IFEM Elasticity. The element is based on Kirchhoff-Love kinematics and discretized with NURBS as basis functions. All conclusions made in this thesis are strictly based on the obtained results presented in Chapter 6.

The main purpose of this work is to implement the Kirchhoff-Love shell element. With the objective to verify the implementation, the generated results have been compared with results from ABAQUS and relevant scientific publications. Five linear benchmark cases have been chosen to test the element's performance. The displacement and internal energy results were used to verify the calculations of the stiffness matrix, and the stress distribution plots were used to verify the stress recovery. The linear benchmark examples gave a unified verdict: the implementation of the linear element formulation has been successful based on the obtained results.

The next three benchmark examples exhibited nonlinear geometrical behavior under deformation and are used to verify the nonlinear implementation of the IGA shell element. The stress distributions and the final deformations were compared with ABAQUS in order to verify the nonlinear geometrical contributions to the stiffness matrix, and the stress distribution plots were used to evaluate the stress recovery in the nonlinear analysis. The nonlinear benchmark examples gave a unified verdict: the implementation of the nonlinear element formulation has been successful based on the obtained results.

From the convergence study conducted on the linear examples, one important finding has been documented: the NURBS-based Kirchhoff-Love shell element converges faster than its equivalent in ABAQUS both in terms of unknowns and number of elements per side. Furthermore, the same

results also show that in simulations where the polynomial degree has been elevated a faster convergence rate was reported, both in terms of unknowns and number of elements per side.

Lastly, the boundary refinement method did not yield a more precise nor a faster convergence rate in terms of unknowns or number of elements per side for this Galerkin discretization. This result is unexpected as the findings of Kiendl *et al.* [17] yields a faster convergence rate. However, in this article a Mindlin-Reissner shell element and a collocation discretization was applied. Hence, the result is not fully comparable.

Chapter 8

Further work

A natural next step, from this work's perspective, would be to continue on and finalize the Kirchhoff-Love shell implementation. Because as of now, shear stress calculations have yet to be implemented. This is the first proposed path forward, but a total of four tasks will be presented and briefly described.

The stress recovery for the shear stresses has not been implemented for the shell element, and is therefore the first suggestion of further work. The reason for this being that this requires the third derivative of the basis functions, and the software does not yet support this function. All equations needed are derived and documented by Kiendl [15]. The shear stresses will in many cases be a decisive factor in an engineering setting, thus it is desirable to implement this functions in IFEM Elasticity, and verify its calculations.

A dynamic analysis verification has yet to be done for the implementation of the Kirchhoff-Love shell element. The analysis would include the dynamic response in both the time and frequency domain for the linear part, while it will suffice to examine only the time domain for in a nonlinear dynamic analysis. No further implementation or changes are necessary to perform such an analysis as IFEM Elasticity already supports this kind of analysis. To do said verification, benchmark examples are needed in order to evaluate the elements performance. The examples can either be chosen from relevant publications, or compared to analyses from ABAQUS.

Plastic material behavior has not been taken into account in the implementation of the Kirchhoff-Love shell element. A suggestion of further work for this thesis is to implement plasticity, and if this had been performed the simulations could have been reevaluated. Presumably the solutions would have encountered plastic behavior, especially for the nonlinear cases, and a different result

would be expected.

The last suggestion of further work is to take use of locally refined (LR) B-splines or NURBS, where the elements are not uniformly distributed but locally refined at critical points, such as under a point load or at points with high stress values. This implementation could be expected to result in faster convergence than for a uniformly refined mesh for the same number of unknowns. LR B-splines are further discussed by Johannessen *et al.* [13].

Bibliography

- [1] Basar, Y. and Krätzig, W. (1985). *Mechanik der Flächentragwerke*. Springer Fachmedien Wiesbaden GmbH.
- [2] Basar, Y. and Weichert, D. (2000). *Nonlinear continuum mechanics of solids*. Springer Fachmedien Wiesbaden GmbH.
- [3] Bathe, K., Iosilevich, A., and Chapelle, D. (2000). An evaluation of the MITC shell elements. *Computers & Structures*.
- [4] Bell, K. (2013). *An engineering approach to finite element analysis of linear structural mechanics problem*. Fagbokforlaget.
- [5] Belytschko, T., Stolarski, H., Liu, W., Carpenter, N., and Ong, J.-J. (1985). Stress projection for membrane and shear locking in shell finite elements. *Computer methods in applied mechanics and engineering*.
- [6] Cook, R., Malkus, D., Plesha, M., and Witt, R. (2001). *Concepts and applications of finite element analysis*. John Wiley & Sons, 4th edition.
- [7] Coradello, L. (2016). Implementation of a high-order Kirchhoff-Love shell: a comparison of IGA and p-FEM. Master's thesis, Technical University of Munich.
- [8] Cottrell, J. A., Hughes, T. J. R., and Bazilevs, Y. (2009). *Isogeometric analysis toward integration of CAD and FEA*. John Wiley & Sons.
- [9] Dassault Systèmes (2018). Abaqus Theory Guide. <http://abaqus.software.polimi.it/v2016/books/stm/default.htm>. [Accessed June 6th 2018].
- [10] Diening, L. and Kreuzer, C. (2013). Convex hull property and maximum principle for finite element minimisers of general convex functionals. *Numerische Mathematik*.

- [11] Holzapfel, G. (2000). *Nonlinear solid mechanics: A continuum approach for engineering*. John Wiley & Sons.
- [12] Hughes, T., Cottrell, J., and Bazilevs, Y. (2005). Isogeometric analysis: CAD, finite elements, NURBS, exact geometry and mesh refinement. *Computer methods in applied mechanics and engineering*.
- [13] Johannessen, K. A., Kvamsdal, T., and Dokken, T. (2014). Isogeometric analysis using LR B-splines. *Computer methods in applied mechanics and engineering*.
- [14] Kaufmann, P., Martin, S., Botsch, M., and Gross, M. (2009). Implementation of discontinuous Galerkin Kirchhoff-Love shells. *Technical Report*.
- [15] Kiendl, J. (2017). *Isogeometric analysis and shape optimal design of shell structures*. PhD thesis, Technical University of Munich.
- [16] Kiendl, J., Bletzinger, K., Linhard, J., and Wüchner, R. (2009). Isogeometric shell analysis with Kirchhoff-Love elements. *Computer methods in applied mechanics and engineering*.
- [17] Kiendl, J., Marino, E., and De Lorenzis, L. (2017). Isogeometric collocation for the Reissner-Mindlin shell problem. *Computer methods in applied mechanics and engineering*.
- [18] Kim, J. and Kim, Y. (2002). Three-node macro triangular shell element based on the assumed natural strains. *Computational mechanics*.
- [19] Macneal, R. and Harder, R. (1985). A proposed standard set of problems to test finite element accuracy. *Finite Elements in Analysis and Design*.
- [20] Nguyen, T. (2011). Isogeometric finite element analysis based on Bézier extraction of NURBS and T-Splines. Master's thesis, Norwegian University of Science and Technology.
- [21] Nguyen, V. P., Anitescu, C., Bordas, S. P. A., and Rabczuk, T. (2015). Isogeometric analysis: An overview and computer implementation aspects. *Mathematics and Computers in Simulation*.
- [22] Okstad, K. (1994). *Adaptive methods for non-linear finite element analysis of shell structures*. PhD thesis, NTNU.
- [23] Okstad, K. (2018). Personal communication, May 24th 2018.

- [24] SINTEF Digital (2018). IFEM Elasticity Github repository. <https://github.com/OPM/IFEM-Elasticity>.
- [25] Sze, K., Liu, X., and Lo, S. (2003). Popular benchmark problems for geometric nonlinear analysis of shells. *Finite Elements in Analysis and Design*.

Appendix A

Constructing a B-spline curve

The B-spline curve in this example will be constructed with a knot vector, polynomial degree and control points. For this example, the curve is to be quadratic, hence $p = 2$, and will exist in \mathbb{R}^2 . The knot vector $\Xi = \{0,0,0,1,2,3,4,4,5,5,5\}$ will be used, which is classified as open due to its first and last entry appears $p + 1$ times. Furthermore, as the knot $\xi = 4$ appears two times, this knot vector is said to be non-uniform.

Just by examining the knot vector it is possible to predict a $C^{p-2} = C^0$ -continuity at $\xi = 4$, and a C^{-1} -continuity at the ends. By applying the Cox-de Boor recursion formula, as in Equations (2.2) and (2.3), the basic functions, as shown in Figure A.1, are obtained. By using Cox-de Boor's formula to generate the basis functions for a quadratic curve, it will generate the basis functions for $p = 0, 1, 2$. This because it is a recursive formula, and the three sets of basis functions are displayed in Figure A.1. By using Equation (2.5), the B-spline curve can be calculated. The control points are defined in Table A.1, and the resulting curve is displayed in figure A.2.

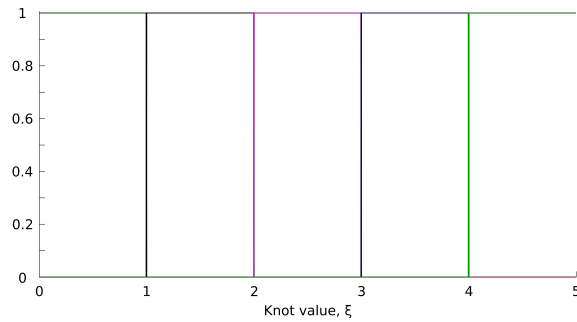
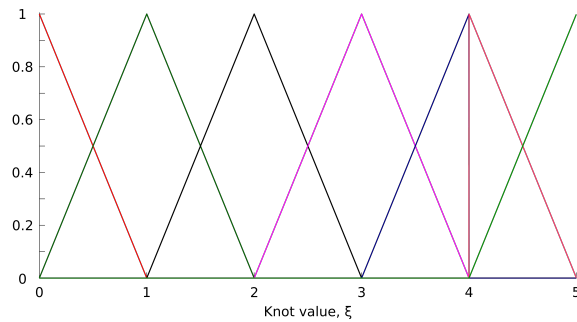
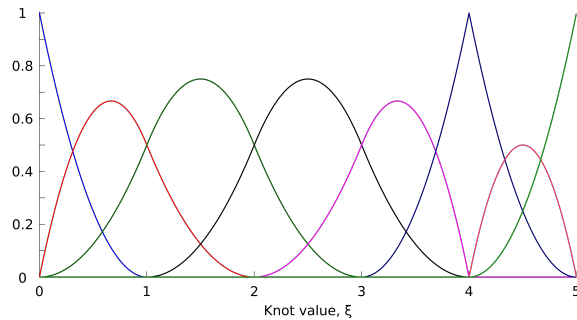
(a) Basis functions for $p = 0$ (b) Basis functions for $p = 1$ (c) Basis functions for $p = 2$

Figure A.1: The basis functions for the open, non-uniform knot vector $\Xi = \{0, 0, 0, 1, 2, 3, 4, 4, 5, 5, 5\}$ obtained by the Cox-de Boor recursion formula. The compact support property is fulfilled for all $N_{i,p}$, which is easiest checked by controlling that $N_{1,2}$ spans from $\xi_1 = 0$ to $\xi_{1+2+1} = \xi_4 = 1$.

Control point i	$[x, y]$
1	[0.30, 0.57]
2	[0.40, 0.74]
3	[0.68, 0.65]
4	[0.59, 0.48]
5	[0.40, 0.48]
6	[0.40, 0.32]
7	[0.30, 0.32]
8	[0.20, 0.40]

Table A.1: Coordinates for all 8 control points.

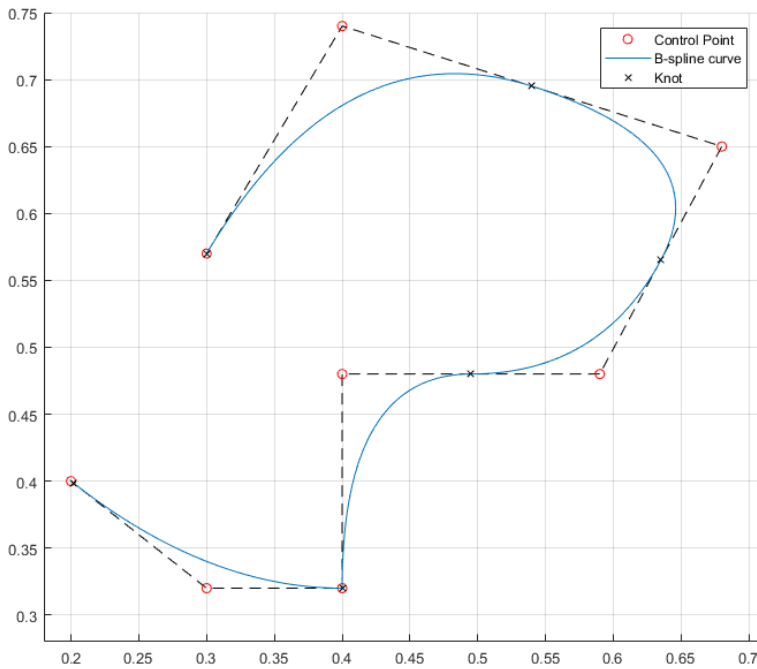


Figure A.2: The resulting B-spline curve based on the given Ξ and \mathbf{B} . The red circles are the control points, the blue curve is the B-spline curve, and the crosses are the knots. The curve has a clear lower order of continuity at the sixth control point, $\xi = 4$, due to its multiplicity.

Appendix B

Multimesh extrapolation

The reference solutions used in the convergence plots are found with multimesh extrapolation, and the method used in this thesis is Richardson's extrapolation [6]. This method predicts a reference solution, which is almost exact, from a set of approximate solutions. In the case of the linear benchmark problems, the values of interest are displacements, internal energy and von Mises stresses at a critical point in the mesh.

Richardson's extrapolation takes known values of the quantity, w_1 and w_2 , calculated with element sizes, h_1 and h_2 , and extrapolate them with the order of error $\mathcal{O}(h^q)$, where q is a constant that is either known or found empirically using the fact that the plot of w versus h^q is a straight line. The extrapolation used to find the reference solution, w_∞ , which is equivalent to a solution where the element size approach zero $h \rightarrow 0$, is

$$w_\infty = \frac{w_1 h_2^q - w_2 h_1^q}{h_2^q - h_1^q}. \quad (\text{B.1})$$

To get an almost exact solution, there are some requirements that the method must meet. The quantity of interest must have a monotonic convergence, the point evaluated in the mesh must be without singularities and the mesh refinements must be such that the mesh corresponding to h_1 is maintained within the mesh corresponding to h_2 . If these requirements are not met, it affects the accuracy of the reference solution in a negative manner.

Appendix C

ABAQUS elements

Following is a description of the ABAQUS elements used in this thesis as they are presented in the documentation of the program [9]. The elements' name tells the following: the first entry "S" stands for "Shell", the second entry is the number of nodes, the third entry stands for "Reduced integration", and if a fourth entry is given it lists the number of DOFs at each node.

S4R

The S4R element is a general purpose three-dimensional shell element with 4 edges and is evaluated with a reduced integration. It is based on the Mindlin-Reissner shell element, and can thus be used in both thin and thick shell problems. This element produces a robust and accurate solution with all types of loading.

S4R5

The S4R5 element is a thin shell element with 4 nodes and 5 DOFs per node. It is based on the Kirchhoff kinematics, which it satisfies in discrete points. It is commonly referred to as a discrete Kirchhoff element, and is not suitable for nonlinear analysis.

S9R5

The same element as S4R5, but with 9 nodes, thus it uses a quadratic basis function to describe the geometry.