Nanna Thoen Frogner and Marte Møkkelgård

A Comparative Study Between the Finite Element Analysis and the Isogeometric Analysis Within a Parametric Environment.

Master's thesis in Civil and Environmental Engineering
Supervisor: First amanuensis Marcin Luczkowski
June 2022

Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Engineering
Department of Civil and Environmental Engineering

NTNU
Kunnskap for en bedre verden

Nanna Thoen Frogner and Marte Møkkelgård

A Comparative Study Between the Finite Element Analysis and the Isogeometric Analysis Within a Parametric Environment.

Master's thesis in Civil and Environmental Engineering
Supervisor: First amanuensis Marcin Luczkowski
June 2022

Norwegian University of Science and Technology
Faculty of Engineering
Department of Civil and Environmental Engineering

NTNU
Norwegian University of
Science and Technology

Department of Structural Engineering
Faculty of Engineering
NTNU – Norwegian University of Science and Technology

ACCESSIBILITY
Open

MASTER'S THESIS 2022

SUBJECT AREA:	DATE:	NO. OF PAGES:
Structural Engineering	10.06.2022	xi + 66 + 1

TITLE:

A Comparative Study Between the Finite Element Analysis and the Isogeometric Analysis Within a Parametric Environment.

Et sammenligningsstudium mellom elementanalyse og isogeometrisk analyse i et parametrisk miljø.

BY:

Nanna Thoen Frogner and Marte Møkkelgård

ABSTRACT:

The common practice for the design process is for architects and engineers to work in separate software programs. However, it is beneficial to have a tight cooperation to create visually pleasing and structurally beneficial designs. In a parametric environment, both the ability to create spatial designs, and run structural analysis are available. This framework enables architects and engineers to work simultaneously in an interactive environment, through Algorithm-Aided Design (AAD). Although, the traditional structural analysis method, the Finite Element Analysis (FEA), causes a workflow in separate models. FEA is the most integrated approach to structural analysis and is based on a discretization of the exact geometry into finite elements (mesh). Discretization is the largest bottleneck for analysis time and can cause modeling errors. With architecture today approaching complex spatial designs and utilizing curved structures, this disadvantage of FEA increases. A new structural analysis method, Isogeometric Analysis (IGA), is developed to analyze the exact geometry from the Computer-Aided Design (CAD). Consequently, architects and engineers can work simultaneously in the same model, the interaction will increase and disadvantages concerning approximate geometry will terminate.

This thesis investigated the state of the art within IGA in the visual programming language Grasshopper and compared it to an FEA within the same parametric environment. The pros and cons of both workflow and accuracy were evaluated as a step towards a more interactive design process between architects and engineers. A FEM plug-in for Grasshopper was developed and compared with an exciting, but rather newly established IGA algorithm, Kiwi. The purpose was to achieve a fair comparison with analyses of similar complexity. Karamba, an established FEA plug-in for Grasshopper, was used to validate the results. In addition, a component generating Non-Uniform Rational Basis-Spline (NURBS) was developed to achieve geometry for IGA and to gain knowledge about this approach to structural analysis.

The thesis concluded that FEA defends its position as leading structural analysis. The execution time associated with the refined mesh is alone the unfortunate aspect. This aspect was present, but not significant even for the complexity of the study cases examined. IGA requires more research to convert the mathematical theory into a numerical and implementable language for an algorithm. This approach to structural analysis has improvements to achieve lower execution time, higher functionality, and more comprehensible output values. A considerable amount of development is required for an IGA to be a reliable tool for structural analysis in Grasshopper compared to a FEA which has been tried and tested for decades. Nevertheless, considerable benefits of bypassing the discretization process required for the FEA should encourage further research and development of IGA in the future. However, the advantageous performance of the FEM is setting high expectations for a future IGA.

RESPONSIBLE TEACHER AND SUPERVISOR:
First amanuensis Marcin Luczkowski

CARRIED OUT AT:
Department of Structural Engineering, Norwegian University of Science and Technology

Acknowledgements

This master's thesis is the concluding work of our Master of Science Degree at the Norwegian University of Science and Technology (NTNU), written at the Department of Structural Engineering. The thesis has given us a greater understanding of the Finite Element Method and Isogeometric Analysis. In addition, we have learned to program in C# and to develop plug-ins for a parametric environment. We are very pleased with what we have learned throughout this master's thesis.

We would like to extend a special thanks to our supervisor Marcin Luczkowski. First and foremost, for sublime guidance throughout the semester, at any time of the day. In addition, he showed leniency and let us go skiing *whenever there is a lot of sun and good snow*. Attraction towards snowy mountains on sunny days ensured our motivation throughout the semester.

Thank you to Sverre Magnus Haakonsen for valuable assistance throughout the semester, especially in the final days.

At last, we would like to thank each other for great support and all the enchanting times we have had during the semester. This master thesis would not have been possible without each other.

Abstract

The common practice for the design process is for architects and engineers to work in separate software programs. However, it is beneficial to have a tight cooperation to create visually pleasing and structurally beneficial designs. In a parametric environment, both the ability to create spatial designs, and run structural analysis are available. This framework enables architects and engineers to work simultaneously in an interactive environment, through Algorithm-Aided Design (AAD). Although, the traditional structural analysis method, the Finite Element Analysis (FEA), causes a workflow in separate models. FEA is the most integrated approach to structural analysis and is based on a discretization of the exact geometry into finite elements (mesh). Discretization is the largest bottleneck for analysis time and can cause modeling errors. With architecture today approaching complex spatial designs and utilizing curved structures, this disadvantage of FEA increases. A new structural analysis method, Isogeometric Analysis (IGA), is developed to analyze the exact geometry from the Computer-Aided Design (CAD). Consequently, architects and engineers can work simultaneously in the same model, the interaction will increase and disadvantages concerning approximate geometry will terminate.

This thesis investigated the state of the art within IGA in the visual programming language Grasshopper and compared it to an FEA within the same parametric environment. The pros and cons of both workflow and accuracy were evaluated as a step towards a more interactive design process between architects and engineers. A FEM plug-in for Grasshopper was developed and compared with an exciting, but rather newly established IGA algorithm, Kiwi. The purpose was to achieve a fair comparison with analyses of similar complexity. Karamba, an established FEA plug-in for Grasshopper, was used to validate the results. In addition, a component generating Non-Uniform Rational Basis-Spline (NURBS) was developed to achieve geometry for IGA and to gain knowledge about this approach to structural analysis.

The thesis concluded that FEA defends its position as leading structural analysis. The execution time associated with the refined mesh is alone the unfortunate aspect. This aspect was present, but not significant even for the complexity of the study cases examined. IGA requires more research to convert the mathematical theory into a numerical and implementable language for an algorithm. This approach to structural analysis has improvements to achieve lower execution time, higher functionality, and more comprehensible output values. A considerable amount of development is required for an IGA to be a reliable tool for structural analysis in Grasshopper compared to a FEA which has been tried and tested for decades. Nevertheless, considerable benefits of bypassing the discretization process required for the FEA should encourage further research and development of IGA in the future. However, the advantageous performance of the FEM is setting high expectations for a future IGA.

Sammendrag

Vanlig praksis for en design prosess i dag, er at arkitekter og ingeniører jobber i ulik programvare. Det er derimot fordelaktig med et tett samarbeid, for å kunne skape visuelt pene og strukturelt smarte design. I et parametrisk rammeverk, har man mulighet til å både lage romlige geometrier og kjøre strukturelle analyser. Dette muliggjør at arkitekter og ingeniører kan jobbe parallelt i et interaktivt miljø, gjennom Algorithm-Aided Design (AAD). Den tradisjonelle tilnærmingen til strukturell analyse, elementmetoden, forhindrer arbeidsflyt i samme modell. Elementmetoden er den mest integrerte tilnærmingen til strukturell analyse, og er basert på å diskretisere en eksakt geometri til *finite elements* (mesh). Diskretisering er det aspektet ved strukturell analyse som tar lengst tid, og det kan oppstå error som resultat av modelleringen. Arkitektur i dag preges av komplekse, romlige design som utnytter seg av kurvet geometri. Ulempene ved diskretisering vil derfor få større betydning. En ny tilnærming til strukturell analyse, isogeometrisk analyse, er utviklet for å kunne analysere eksakt geometri fra en Dataassistert Konstruksjon (DAK) direkte. Dette muliggjør at arkitekter og ingeniører kan jobbe parallelt i samme modell, og reduserer ulempene vedrørende tilnærmet geometri.

Denne masteroppgaven har undersøkt Isogeometrisk analyse i kontekst av det visuelle programmerings språket Grasshopper, og sammenlignet det med elementmetoden innenfor det samme parametriske rammeverket. Fordeler og ulemper tilknyttet arbeidsflyt og nøyaktighet er evaluert, som et steg mot en mer interaktiv design prosess mellom arkitekter og ingeniører. Det er utviklet en tilleggspakke for elementmetoden for Grasshopper, for å sammenligne med en eksisterende, men nyetablert, isogeometrisk algoritme Kiwi. Hensikten var å utføre en mest mulig rettferdig sammenligning, med tilleggspakker av tilsvarende kompleksitet. Karamba er et veletablert program for strukturell analyse i bransjen, og ble derfor brukt til å validere resultatene. I tillegg er det utviklet en algoritme som genererer Non-Uniform Rational Basis-Spline (NURBS) som input geometri til isogeometrisk analyse, og for å opparbeide kunnskap om denne tilnærmingen til strukturell analyse.

Oppgaven konkluderer med at elementmetoden fortjener sin ledende rolle innen strukturell analyse. Utførelsestid tilknyttet diskretisering er den eneste ulempen. Dette aspektet skapte imidlertid ingen problemer for oppgaven, selv for kompleksiteten på studiecasene som ble undersøkt. Det kreves mere forskning for å konvertere den matematiske teorien til isogeometrisk analyse til et numerisk språk tilpasset en algoritme. Denne tilnærmingen til strukturell analyse har et forbedringspotensial innen utførelsestid, funksjonalitet, og presentasjon av resultater. Det kreves en betydelig utvikling for at en isogeometrisk analyse skal bli et pålitelig verktøy for strukturell analyse i Grasshopper, sammenlignet med elementmetoden som har blitt utprøvd og testet over flere tiår. Fordelene ved å omgå diskretiseringsprosessen som kreves for elementmetoden oppmuntrer derimot til ytterligere forskning og utvikling av isogeometrisk analyse. Den fordelaktige ytelsen av elementmetoden setter store forventninger til en eventuell fremtidig isogeometrisk analyse.

Table of Contents

List of Figures

List of Tables

List of Listings

Definitions

AAD Algorithm-Aided Design

C# Programming language

CAD Computer Aided Design

DOF Degree Of Freedom

FEA Finite Element Analysis

FEM Finite Element Method used in FEA

IGA Isogeometric Analysis

NURBS Non-Uniform Rational Basis-Spline

RHS Right Hand System

VS Visual Studios

1 Introduction

In contemporary building projects, it is common practice for the participants to work in separate software. In addition, they are focusing on different areas of design. While the architects work with the aesthetics and expression of a design, the structural engineers strive for safe and efficient load bearing systems. Sharing models between the professions is rather time consuming, and desires from one discipline can be at the expense of the other.

It is beneficial for architects and engineers to have a tight cooperation in an interactive design process. The design should be explored dynamically, to identify the best shape. Parametric design is a design approach that is based on using parameters to describe numerous sets of geometries and gives the advantage of interactively visualizing modifications of the design (Caetano et al., 2020). As the geometry changes, it is convenient to acquire an overall understanding of how it behaves structurally. A structural analysis package for a parametric environment gives the opportunity to quickly assess the work of a design process in a structural manner. This opens for an interface between architects and engineers to connect architectural shape and structural optimization. This can lead to more optimized structures, as the structural behavior will have a more significant influence on the design process.

Today, the Finite Element Analysis (FEA) is the most widespread structural analysis. The Finite Element Method (FEM) was invented and developed through publications and work from several authors in the 1950s. The first textbook popularizing the FEM was published in 1957 by Zienkiewicz (Gupta and Meek, 1996). Characteristics of this method are dividing the structural system into finite, linear parts, referenced to as *finite elements*. This indicates that the geometry for the FEA is an approximation (mesh) of the Computer-Aided Design (CAD), and is time expensive for the structural analysis. In addition, discretization can cause modeling errors, and obstruct the interface between architects and engineers as it prevents a workflow through a shared model.

The architecture today is approaching complex spatial designs, utilizing curved structural elements (Zhang et al., 2016). Non-Uniform Rational Basis-Spline (NURBS) is fundamental for achieving designs with such characteristics, and most CAD systems are based on NURBS. Isogeometric Analysis (IGA) is bridging the gap between the CAD model and the structural model (Zhang et al., 2016). Hughes developed this approach for structural analysis in the early 21st century. He aimed to unify the fields of CAD and FEM to ensure exact geometry for the analysis, and to reduce analysis time and modeling errors (Cotterell et al., 2009). IGA uses NURBS to interpolate the geometry and solution space, which indicates that architects and engineers can work in the same model.

The thesis starts by introducing the necessary theory of both FEA and IGA. Thereafter, a description of the implementation of tools in Grasshopper follows. There are developed a FEA software and NURBS geometry as input for an already existing IGA software, Kiwi. The theory of the FEM has a more significant attachment in the profession. IGA is relatively new, and Kiwi is newly established, compared to the FEA software. Therefore, it seemed necessary to create a FEM solver to compare with Kiwi, to examine more equal and fair. The performance of the tools is then verified before the workflows of the different methods are established and compared through a series of case studies.

The intention was to evaluate the advantage of not having to discretize curved structures into a mesh and instead analyze the exact geometry. Finally, the observations are discussed, and the results are presented in the conclusion.

2 Theory

2.1 Parametric Design

The design process has changed throughout the years. A diversity of computational design methods, caused by an increased computational capacity, have revolutionized the traditional design process (Caetano et al., 2020). Based on Oxford´s definition on parameters and parametric, the definition of parametric design is defined as a design process based on algorithmic thinking that uses parameters and rules to constrain them (Caetano et al., 2020). Parametric design is a computational design method that currently is challenging and renovating the traditionally architectural design process. The characteristics of structures have changed from straight lines, sharp corners and acute angels to sweeping lines, curves and irregular shapes (Engle, 2020). With this new innovative technology one can make more spectacular structures to perform the role as landmarks and increase the attractiveness of an area.

Parametric design is a new innovational way of designing structures by making a parametric framework. With this new method you dynamically explore the structure and have a more iterative process in determining the geometry, based on a finite set of input parameters (Peters and Peters, 2013). It is based on the idea that all elements in the design are interdependent and adaptable (Engle, 2020). Instead of defining a single static representation of a physical design, you make a process description for how the design is created (Peters and Peters, 2013). This process description is an instruction for how the design should be generated, in computer terminology this is called an algorithm.

2.2 Algorithm-Aided Design

Algorithm-Aided Design (AAD) is an approach based on describing computer programs that generate space and form from the rule-based logic inherent in architectural programs, typologies, building code, and language itself (Caetano et al., 2020). Thus, AAD allows designers to incorporate the computational complexity and creative use of computers within the design workflow.

An algorithm is a computational procedure for addressing a problem in a finite number of steps (Terzidis, 2004). It involves deduction, induction, abstraction, generalization, and structured logic. It is the systematic extraction of logical principles and the development of a generic solution plan. Algorithmic strategies utilize the search for repetitive patterns, universal principles, interchangeable modules, and inductive links. The intellectual power of an algorithm lies in its ability to infer new knowledge and to extend certain limits of the human intellect.

2.3 Finite Element Method

2.3.1 General

The FEM is a method for numerical solutions of field problems (Cook et al., 2001). The basic idea is to divide a complex system behaviour into subsystems with 'known' behaviour (Bell, 2013). This method consists of using a simple approximation of unknown variables to transform partial differential equations into algebraic equations (Dhatt et al., 2012). In other words, one should transform a physical model of the geometry into a mathematical model, which, in most cases, needs a numerical model in order to be solved.

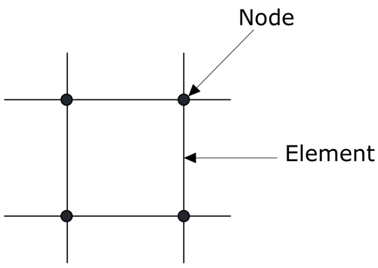


Figure 2.1: A mesh describing nodes and elements

The basis of the FEM is to divide the geometry into finite elements. Individual finite elements can be visualized as small pieces of a structure (Cook et al., 2001). In each finite element a field quantity is allowed to have only a simple spatial variation, described by polynomial terms. This is an approximation to the actual variation of the displacement pattern of the elements. Finite elements are connected at the nodes, which are the points connecting neighbouring elements, presented in Figure 2.1. The assemblage of elements is called a finite element structure, and the specific arrangement of elements is called a mesh. Numerically, an finite element mesh is represented by a system of algebraic equations to be solved for unknowns at the nodes, by the following formula:

$$\mathbf{R} = \mathbf{K}\mathbf{r} \tag{2.1}$$

where $\mathbf{r}$ are the nodal unknowns, $\mathbf{K}$ is the matrix describing the stiffness of the structure, and $\mathbf{R}$ is the load vector. Nodal unknowns are values of the field quantity, and are refereed to as the degrees of freedom (DOFs) of the geometry. Depending on the element type, DOFs can be described by both values of the field quantity and its first derivatives (Cook et al., 2001). The behaviour of the individual elements is expressed by the following relationship:

$$\mathbf{S} = \mathbf{k}_e\mathbf{v} \tag{2.2}$$

between the nodal unknowns $\mathbf{v}$ and the corresponding nodal forces $\mathbf{S}$ (Bell, 2013). $\mathbf{k}_e$ is the matrix describing the stiffness of the element.

The solution for nodal quantities combined with the assumed displacement field, completely determines the spatial variation of displacement behaviour in that element (Cook et al., 2001). The displacement pattern over the entire structure is approximated element by element. Therefore, a FEM solution is not exact, but the result can be improved by using a finer mesh to represent the structure.

To obtain a valid solution of a structural analysis, the following assumptions must be satisfied (Bell, 2011):

- **Static equilibrium** between prescribed loads and internal stresses,
- **Kinematic compatibility** between displacements and strains ($\varepsilon = \frac{du}{dx}$),
- **Material law** the relationship between stress and strain ($\sigma = E\varepsilon$)

2.3.2 Euler-Bernoulli

Euler-Bernoulli is an elementary theory for beam analysis. The theory neglects transverse shear, assuming Navier's hypothesis; plane section that is perpendicular to the normal axis before bending will remain plane and orthogonal after deformation (Bell, 2013).

$$\theta = -w_{,x} \tag{2.3}$$

Equation 2.3 indicates that the rotation angles is equal to the slope of the deflection curve. The slope is the derivative of the deflection curve. This is an approximation and simplification to how beams behave under deformation.

Euler-Bernoulli requires five assumptions to be fulfilled (Bell, 2013);

1. Small displacements,
2. Linear elastic and homogeneous material,
3. The strong form of Navier's hypothesis,
4. The beam is prismatic, meaning the parameters E, A, and I, are constant along the entire beam, where
   - E is the modulus of elasticity (Young's modulus),
   - A is the cross section area and,
   - I is the second moment of the cross section area (moment of inertia).
5. Normal stress in the z- direction (σ_z) is neglected.

Assumption 4 is not strictly necessary considering that the theory gives good results for varying cross section and beams with moderate curvature.

For a straight beam loaded with point load in the transverse direction the deflections pattern is of cubic polynomial (Mathisen, 2012). Therefore, shape functions of third degree provides accurate results for deflection.

2.3.2.1 Shape functions

A shape function N is an assumed field displacement behaviour, that has to be appropriately established for each element (Dhatt et al., 2012). The displacement pattern of the element $\mathbf{u}$ is established by interpolating the shape function between the nodal displacements $\mathbf{v}$ (Bell, 2011). To interpolate is to devise a continuous function that satisfies prescribed conditions at a finite number of points (Cook et al., 2001). In the FEM, the points are nodes of an element, and the prescribed conditions are nodal values $\mathbf{v}$ of a field quantity. Nodal values are rarely exact, and even when they are, interpolation generally provides approximated values at the other locations. In the FEM, it is assumed that the displacement $\mathbf{u}$ within the element can be interpolated between the nodal DOFs $\mathbf{v}$ via the shape functions (Bell, 2013). Displacement can be established by:

$$\mathbf{u} = \mathbf{N_q q} \tag{2.4}$$

where $\mathbf{N_q}$ are *generalized* shape functions and $\mathbf{q}$ are *generalized* parameters for displacement. The generalized shape functions are polynomials. The relationship between the physical DOFs $\mathbf{v}$ and the generalized displacements $\mathbf{q}$, are (Bell, 2013):

$$\mathbf{q} = \mathbf{A}^{-1}\mathbf{v} \tag{2.5}$$

where each row of $\mathbf{A}$ is $\mathbf{N_q}$ evaluated at nodal location of the specific DOF.

$$\mathbf{u} = \mathbf{N_q A}^{-1}\mathbf{v} = \mathbf{Nv} \tag{2.6}$$

$$\mathbf{N} = \mathbf{N_q A}^{-1} \tag{2.7}$$

An individual N_i in matrix $\mathbf{N}$ is called a shape function (Cook et al., 2001).

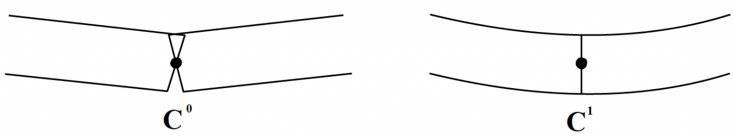


Figure 2.2: C^0 and C^1 continuity.

There are three requirements for the shape functions in order for the elements to converge to the correct solution (Bell, 2013) :

- **Continuity**: shape function and derivatives up to $m-1$ must be continuous along the entire boundary of neighbouring elements to have C^{m-1}-continuity. m is the order of differentiation in the strain- displacement operator Δ. Figure 2.2 is clarifying continuity.
- **Completeness**: the element must be capable of describing rigid body motions and constant strain without producing any stress in the element. This requirement is fulfilled if the shape functions contains complete polynomials of order m.
- **The interpolation requirement**: any shape function N_i must have the value of 1 at the boundary of the element, so for DOF i equal to 1 all other element DOFs is equal to 0. In other words, every arbitrary point in the element the shape functions have to satisfy the following requirement:
$$\sum_i N_i = 1$$

If all the requirements presented above are satisfied, one can know that the element will converge to the correct solution (Bell, 2013). Shape functions describing the elements can be, for various reasons, modified in a way that are in conflict with the basic requirements for convergence. To establish the quality of these elements, performance testing have to be done. One of the most important and useful tests is the *patch test*. For this test one element in an arbitrary mesh should be considered, and the size (h) of the element should be in the limit $h \to 0$. For convergence, the element should satisfy the following requirements:

- correct representation of rigid body movements,
- ability to represent constant strain within the elements,
- sufficient displacement continuity at the element nodes,

as the size (h) of the element is decreasing. The test only requires a state of constant strain within the element, and convergence will occur even though the continuity requirement is not fully satisfied.

2.3.2.2 Isoparametric Element

Isoparametric elements are elements that has irregular shape, and the interpolation function N_q is used to describe both the geometry and the unknown displacement pattern, $N_q = N$ (Bell, 2013). An irregular shaped element covers element that has corner angles that deviate from 90°, curved element sides and midside nodes that are located outside the midpoint.

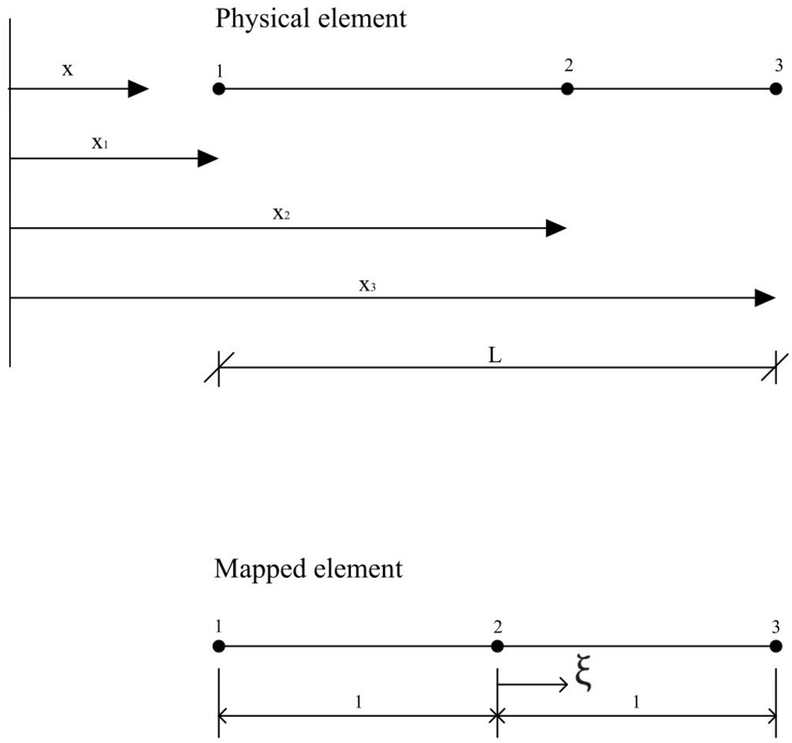


Figure 2.3: Physical and mapped element.

A 1D element operates with ξ, that is non-dimensional between [-1,1]. Each ξ in the reference element have a corresponding x-position in the physical element, see Figure 2.3. The relation is

$$X = X(\xi) \tag{2.8}$$

To integrate over an irregular shaped element with various boundary conditions, the element should be mapped into a natural coordinate system (Bell, 2013). The boundary for the natural coordinate system is $[\xi = -1, \xi = 1]$, this boundary also allows for Gauss quadrature and is easy to implement in computer programs. In order to relate the physical element to the mapped element, the Jacobian is implemented. Mapping is visualized in Figure 2.3. For one-dimensional beam problems, the Jacobian is a scalar, more specific the length of the beam, and is defines as:

$$J = \frac{dx}{d\xi} \tag{2.9}$$

Since the natural coordinates varies from $[\xi = -1, \xi = 1]$, the relationship between ξ and x is the following for a straight structural member:

$$\xi = \frac{2x}{L} - 1 \tag{2.10}$$

2.3.2.3 K-matrix element

A formula for the element stiffness matrix is established from *the principle of virtual work*, stated on the form (Cook et al., 2001):

$$\int \delta\epsilon^T \boldsymbol{\sigma}\, dV = \int \delta\mathbf{u}^T \mathbf{F}\, dV + \int \delta\mathbf{u}^T \boldsymbol{\Phi}\, dS + \mathbf{S}^T \delta\mathbf{v} \tag{2.11}$$

Equation 2.11 must be valid for any $\delta\mathbf{v}$, hence (Bell, 2013):

$$\mathbf{S} = \int_{V_e} \mathbf{B}^T \mathbf{C}\mathbf{B}\, dV\, \mathbf{v} - \int_{V_e} \mathbf{B}^T \mathbf{C}\epsilon_0\, dV - \int_{V_e} \mathbf{N}^T \mathbf{F}\, dV + \int_{S_T} \mathbf{N}^T \boldsymbol{\Phi}\, dS = \mathbf{k}_e\mathbf{v} + \mathbf{S}^0 \tag{2.12}$$

where

$$\mathbf{k}_e = \int_{V_e} \mathbf{B}^T \mathbf{C}\mathbf{B}\, dV \tag{2.13}$$

is the *element stiffness matrix* and

$$\mathbf{S}^0 = -\int_{V_e} \mathbf{B}^T \mathbf{C}\epsilon_0\, dV - \int_{V_e} \mathbf{N}^T \mathbf{F}\, dV + \int_{S_T} \mathbf{N}^T \boldsymbol{\Phi}\, dS \tag{2.14}$$

is the *consistent element load vector*.

A *generalized element stiffness matrix* $\mathbf{k}_{e.q}$ can be derived based on $\mathbf{N}_q$, from section 2.3.2.1 (Bell, 2013). The element stiffness matrix $\mathbf{k}_e$ can be obtained by *transforming* $\mathbf{k}_{e.q}$. The procedure is as follows:

$$\mathbf{u} = \mathbf{N}_q\mathbf{q} \Rightarrow \epsilon_0 \boldsymbol{\Delta}\mathbf{u} = \boldsymbol{\Delta}\mathbf{N}_q\mathbf{q} = \mathbf{B}_q\mathbf{q} = \mathbf{B}_q\mathbf{A}^{-1}\mathbf{v} \Rightarrow \mathbf{B} = \mathbf{B}_q\mathbf{A}^{-1} \tag{2.15}$$

Inserting for $\mathbf{B}$ from equation 2.15 in equation 2.13, the element stiffness matrix $\mathbf{K}_e$ can be expressed as

$$\mathbf{k}_e = \mathbf{A}^{-T} - \int_{V_e} \mathbf{B}_q^T \mathbf{C}\mathbf{B}_q\, dV\, \mathbf{A}^{-1} = \mathbf{A}^{-T}\mathbf{k}_{e.q}\mathbf{A}^{-1} \tag{2.16}$$

where

$$\mathbf{k}_{e.q} = \int_{V_e} \mathbf{B}_q^T \mathbf{C}\mathbf{B}_q\, dV \tag{2.17}$$

2.3.2.4 Transformation matrix

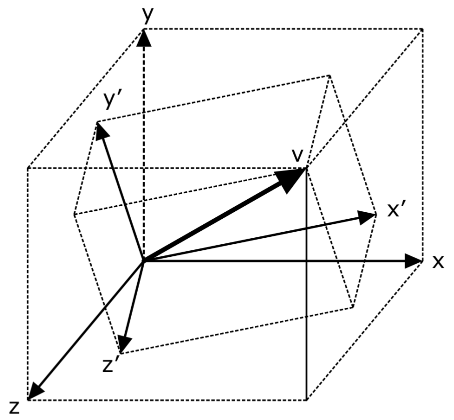


Figure 2.4: Coordinate system explaining the transformation matrix.

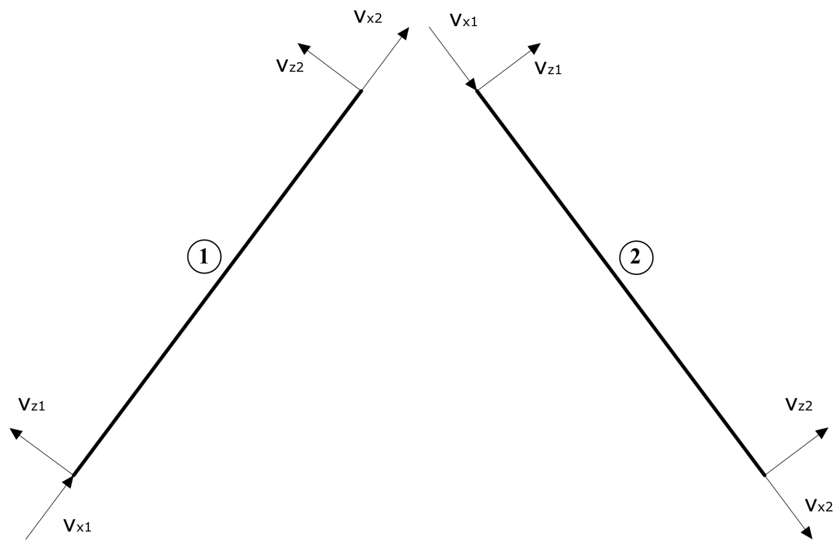


Figure 2.5: Local DOFs related to the element (**v**).

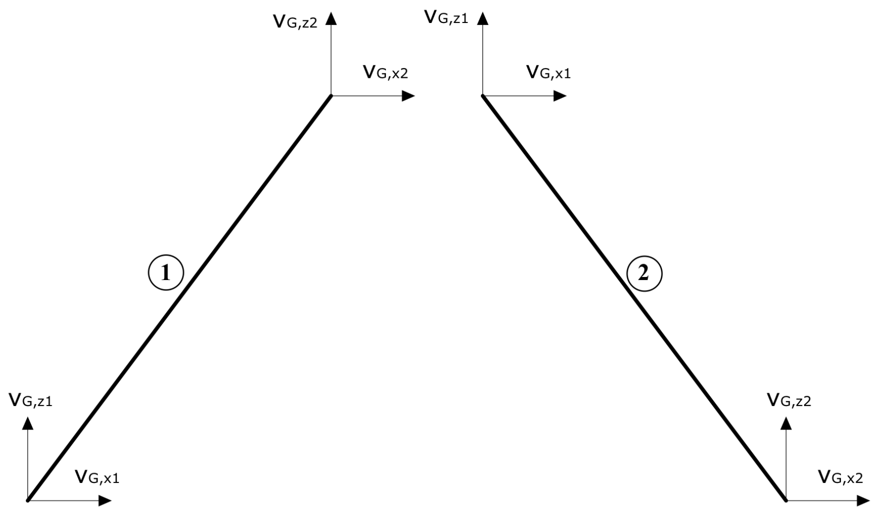


Figure 2.6: Global DOFs related to the element ($\mathbf{v}_G$).

Finite elements must be capable of assuming any orientation in space (Cook et al., 2001). This capability can be provided by doing a simple manipulation of the already derived element matrices. The already derived stiffness matrix $\mathbf{k}_e$ for the element is obtained from the local coordinate system (x', y', z') of the element, see Figure 2.4 and Figure 2.5. This axis is arbitrarily oriented in a global system (x, y, z), see Figure 2.4. To obtain an element matrix $\mathbf{k}_{G.e}$ that operates on DOF refereed to as global coordinates (x, y, z) (see Figure 2.6), a rotational coordinate transformation must be applied to the local stiffness matrix $\mathbf{k}_e$.

The relation between local and global DOF is (Cook et al., 2001):

$$\mathbf{v} = \mathbf{T}\mathbf{v}_G \tag{2.18}$$

Likewise, the relation between local $\mathbf{k}_e$ and global $\mathbf{k}_{G.e}$ element stiffness matrix (Bell, 2011):

$$\mathbf{k}_{G.e} = \mathbf{T}^T \mathbf{k}_e \mathbf{T} \tag{2.19}$$

The transformation matrix $\mathbf{T}$ depends on the characteristic of the element, and will be produced by stacking the Δ -matrix on the diagonal for each node of the element (Cook et al., 2001). For the 3D beam, the transformation of the translations and rotations will be similar, and the Δ -matrix has to be stacked twice for each node. The transformation Δ is:

2D truss:

$$\Delta = \begin{bmatrix} l_1 & m_1 \\ l_2 & m_2 \end{bmatrix}$$

2D beam:

$$\Delta = \begin{bmatrix} l_1 & m_1 & 0 \\ l_2 & m_2 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

3D truss/beam:

$$\Delta = \begin{bmatrix} l_1 & m_1 & n_1 \\ l_2 & m_2 & n_2 \\ l_3 & m_3 & n_3 \end{bmatrix}$$

Where l_1 is the cosine of the angle between the x'- and x-axis, m_1 is the cosine of the angle between the x'- and y- axis, and so on, visualised in Figure 2.4.

2.3.2.5 K-matrix total

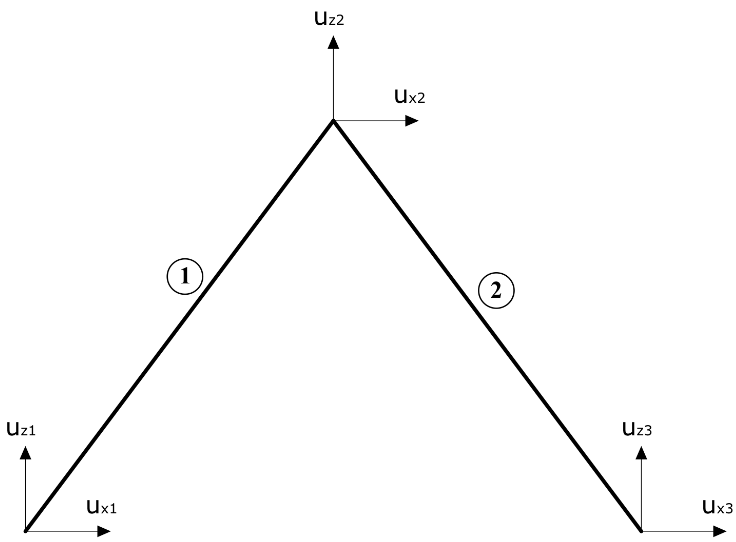


Figure 2.7: Global DOFs related to the system (**r**).

The total stiffness matrix of the system **K** is derived from the fundamental requirements presented in section 2.3.1 (Bell, 2011). The formula is:

$$\mathbf{K} = \sum_{i=1}^{m} (\mathbf{a}^i)^T \mathbf{k}^i_{G.e} \mathbf{a}^i \tag{2.20}$$

where m is the number of elements in the mesh. The kinematics, how the DOFs of the element **v** are related to the DOFs of the system **r** (see Figure 2.7), are defined by the connectivity matrix **a**.

The relation is:

$$\mathbf{v}^i_G = \mathbf{a}^i\mathbf{r} \tag{2.21}$$

2.3.3 Timoshenko beam theory

For beams that have a more severe bending, Timoshenko beam theory present a more accurate representation of how the beam deflect during bending. It accounts for shear deformation and therefor include transverse shear in the stiffness matrix (Bell, 2013). Shear deformation is approximated to average shear strain over the height. The rotation angles (θ) of the deformed beam includes both the slope of the deflection curve ($-w_{,x}$), and shear strain (γ).

$$\theta = \gamma - w_{,x} \tag{2.22}$$

$$\theta = \frac{\kappa}{GA}V - w_{,x} \tag{2.23}$$

$$V = M' = -EIw_{,xxx} \qquad G = \frac{E}{2(1+\nu)} \tag{2.24}$$

κ is a parameter depending on the cross section, V is the shear force from applied load, G is the shear modulus, E is the Young's Modulus and A is the area of the cross section.

Constructing the stiffness matrix in Timoshenko theory is different than for Euler-Bernoulli theory. The stiffness matrix is a product of the generalized stiffness matrix and the A-matrix. The A-matrix is constructed from the generalized shape function, where w is translation and θ is rotation. The generalized shape functions are $\mathbf{N}_q = [1 \quad x \quad x^2 \quad x^3]$ for 2D beam.

$$w = \mathbf{N}_q\mathbf{q} \tag{2.25}$$

$$\mathbf{k}_e = \mathbf{A}^{-T}\mathbf{k}_{e.q}\mathbf{A}^{-1} \tag{2.26}$$

Timoshenko is a beam theory that applies for deep beams. When using this method to analyze slender beams spurious constraints may appear and create shear locking (Bell, 2013). The beam troubles to produces zero strain as it becomes slender and therefore produces shear locking. This will result in underestimation of the deflection field. The most effective and popular method to deal with shear locking is reduced integration. This method uses less Gauss points than needed to integrate exactly.

Assuming reduced integration, the Timoshenko beam is a robust element that gives accurate results for all height to length ratios (Bell, 2013). When the beam is thick is takes into account the shear strain and for slender beams it converges to the Euler-Bernoulli element. A short span Timoshenko beam has a C^0 continuity condition, however for small depth to length ratio, the element relies more on a C^1 continuity. Continuity is clarified in Figure 2.2.

2.4 Isogeometric Analysis

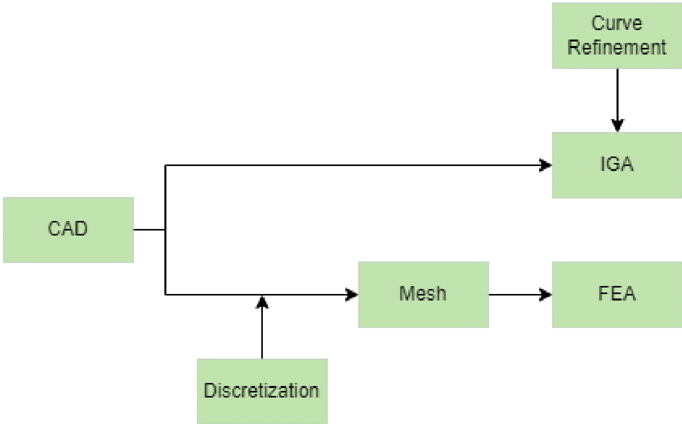


Figure 2.8: Flow chart for FEA and IGA.

Using IGA, the structural analysis is performed directly on the exact geometrical description of curved members given by the CAD model (Zhang et al., 2016). NURBS are used in most CAD program to describe geometry. Hughes based the numerical analysis functions on NURBS to ensure exact geometry. The fundamental different between FEA and IGA in presented in Figure 2.8.

2.4.1 Non-Uniform Rational B-spline Curve

NURBS is a mathematical model for representing curves using B-Splines, which are functions defined piecewise by polynomials. The major strength of NURBS is the way to exactly represent geometry, and the existence of many numerically stable and efficient algorithms to generate NURBS objects (Cotterell et al., 2009). Furthermore, the mathematical properties have the ability to refine through knot insertion (see section 2.4.1.1) and C^{p-1} continuity for NURBS of order p are handy .

The parameter space of NURBS is local to patches rather than elements, which is different from the traditional finite element analysis (Cotterell et al., 2009). As the reference element, or the parameter space, in the FEA is mapped into an element in the physical space, the NURBS mapping takes a patch of multiple elements in the parameter space into the physical space. Therefore, each element in the physical space is an image of the corresponding element in the parameter space, but the mapping is global to the patch and not the elements.

2.4.1.1 Knot vector

A knot vector consists of non-decreasing parameter to describe the NURBS (Cotterell et al., 2009). It has the purpose to regulate the control points in a parameter space. It is expressed as

$$\Xi = [\xi_1, \xi_2, ..., \xi_{n+p+1}] \tag{2.27}$$

The length of the vector is always equal to the sum of the degree (p) of the curve, number of control points (n) plus one. ξ_i is a knot, whereas i is the knot index, i = 1, 2, ..., n + p + 1.

In Isogeometric terminology, the member is considered as a patch while the non-zero knot span is considered an element (Cotterell et al., 2009).

2.4.1.2 Basis functions

NURBS curves are represented by a series of NURBS basis functions, where the basis functions can have different polynomials (Zhang et al., 2016) (Gan, 2018). A basis function is a single curve element represented by one specific polynomial, which is used as a basis of linear combinations to construct a particular curve (Gan, 2018). In the finite element method the basis functions are refereed to as *shape functions*, as presented in chapter 2.3.2.1 (Cotterell et al., 2009).

A NURBS curve is represented by NURBS basis functions $\mathbf{R}_{q,p}(\xi)$ through the following formula (Cotterell et al., 2009):

$$\mathbf{x}(\xi) = \sum_{q=1}^{n} \mathbf{R}_{q,p}(\xi)\mathbf{x}_q \tag{2.28}$$

The curve is parametrized by the scalar ξ, and the position vector of a point on the curve is $\mathbf{x}(\xi) = [x(\xi), y(\xi), z(\xi)]$. $\mathbf{x}_q$ is considered the coordinates in the x,y,z coordinate system. The NURBS basis functions are given by (Cotterell et al., 2009):

$$\mathbf{R}_{q,p}(\xi) = \frac{N_{q,p}(\xi)w_q}{\sum\limits_{j=1}^{n} N_{j,p}(\xi)w_j} \tag{2.29}$$

$N_{q,p}(\xi)$ is the basis function for B-splines of degree p, and q indicates the knot index. The basis functions are constructed recursively (Zhang et al., 2016). They can be obtained by the following formula, which is reffered to as the **Coc-de Boor recursion formula** (Cotterell et al., 2009):

$$N_{q,0}(\xi) = \begin{cases} 1 & \text{if} \quad \xi_q \leq \xi < \xi_{q+1} \\ 0 & \text{otherwise} \end{cases} \quad , \quad \text{p = 0} \tag{2.30, 2.31}$$

$$N_{q,p}(\xi) = \frac{\xi - \xi_q}{\xi_{q+p} - \xi_q} N_{q,p-1}(\xi) + \frac{\xi_{q+p+1} - \xi}{\xi_{q+p+1} - \xi_{q+1}} N_{q+1,p-1}(\xi), \quad \text{p} \geq 1 \tag{2.32}$$

NURBS basis functions are established in a way that all rigid body motions and constant strain states are exactly represented. Standard patch tests are likewise satisfied (Cotterell et al., 2009).

2.4.1.3 Control point

In order to construct a NURBS curve, control points are necessary. The control points are related to the basis functions as the *i*th control point works as a scalar multiplier of the *i*th basis function (Cotterell et al., 2009). Therefore, the control point can be reffered to as a "controller" of a corresponding basis function.

2.4.1.4 Weights

Weights are what differentiate NURBS curves from other splines. NURBS are rational and can have weights different from 1. Changing the weight of a control point will have consequences for the influence of that specific point for the geometry of the curve. The curvature will be "pulled" towards the control point of highest weight (Zhang et al., 2016). Due to this, NURBS curves are more flexible and can describe more geometries than other splines.

2.4.2 Example of NURBS

To visualize the theory, there will be presented three different NURBS.

The first curve is of degree 1 and have three control points (Cpt):

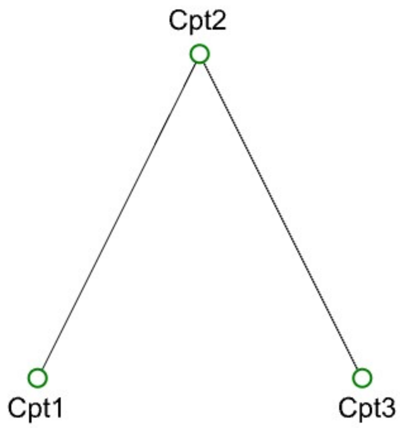


Figure 2.9: NURBS curve with three control points of degree 1.

The knot vector is as follows:

$$\Xi = [0, 0, 0.5, 1, 1]$$

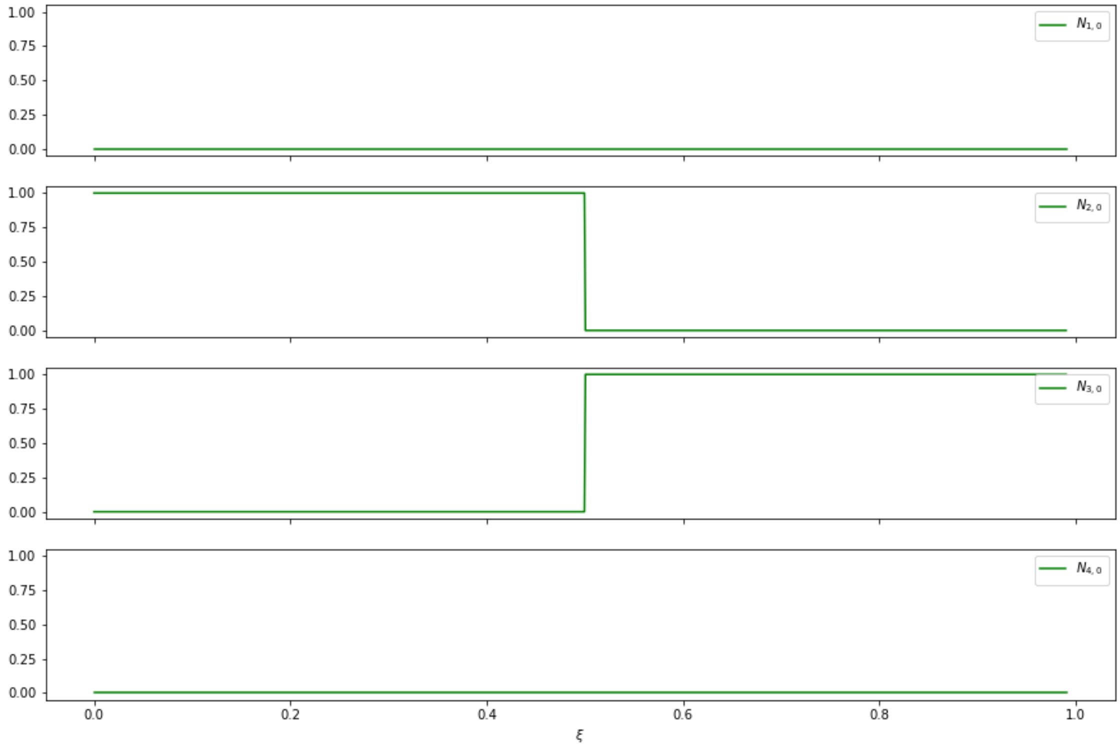


Figure 2.10: Basis functions for degree 0 ($N_{q,0}$, q = 1,2,3,4).

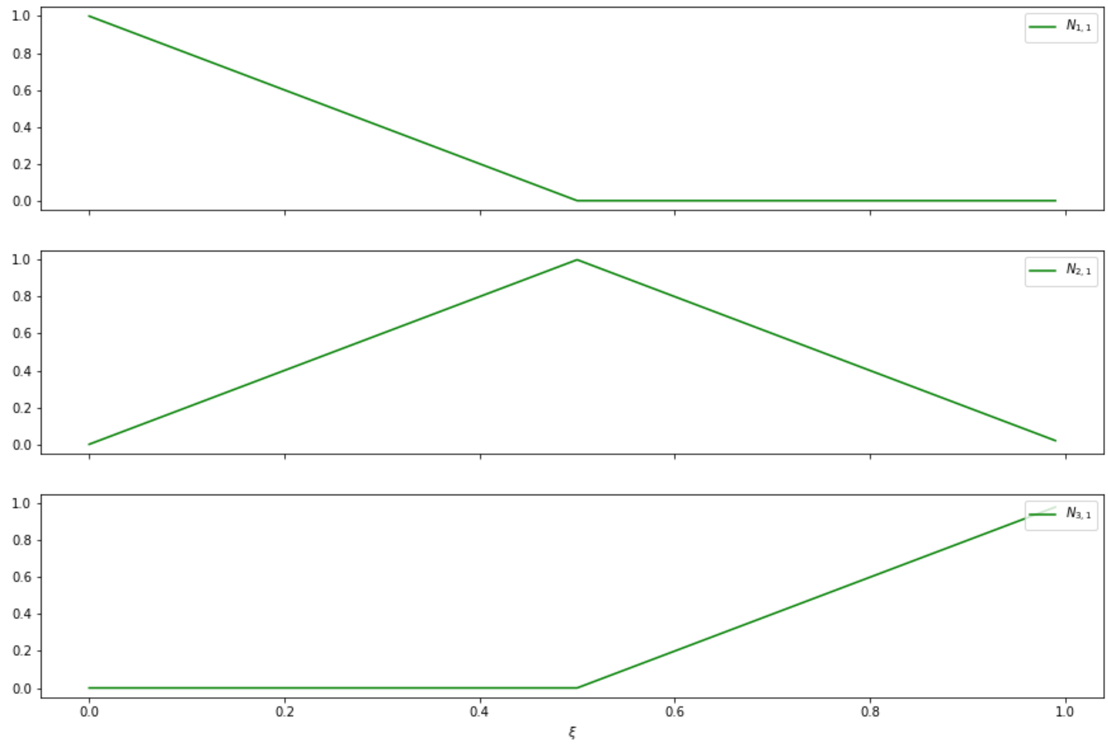


Figure 2.11: Basis functions for degree 1 ($N_{q,1}$, q = 1,2,3).

Figure 2.10 and Figure 2.11 present an example of NURBS basis functions $N_{q,p}$ of order 0 and 1 for the chosen uniform knot vector Ξ.

The second curve is of degree 3 and have four control points (Cpt):

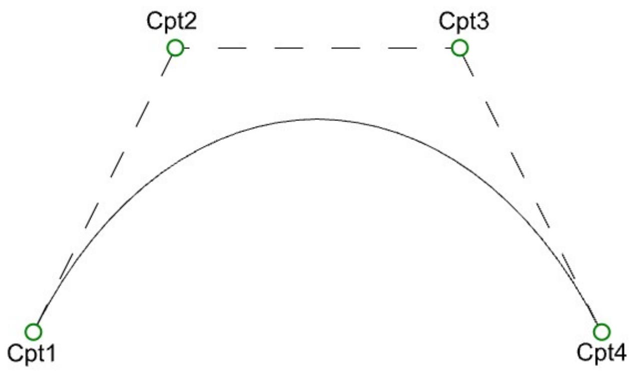


Figure 2.12: NURBS curve with four control points of degree 3.

The knot vector is as follows:

$$\Xi = [0,0,0,0,1,1,1,1]$$

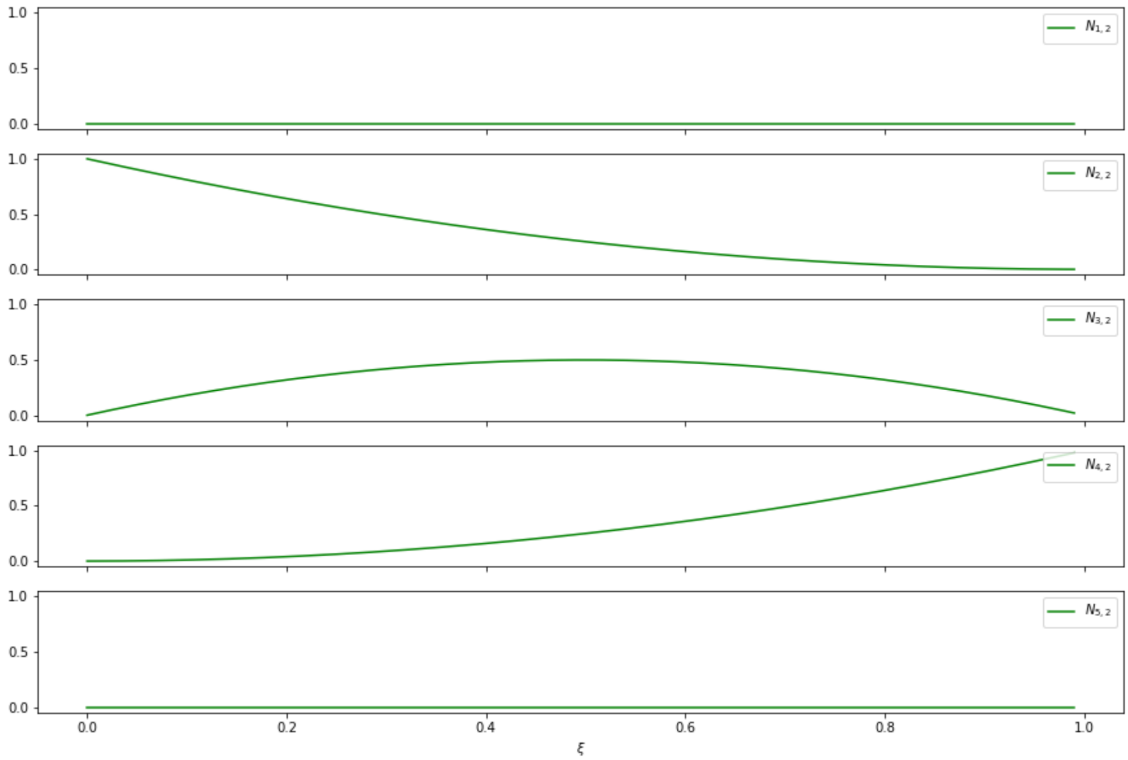


Figure 2.13: Basis functions for degree 2 ($N_{q,2}$, q = 1,2,3,4,5).

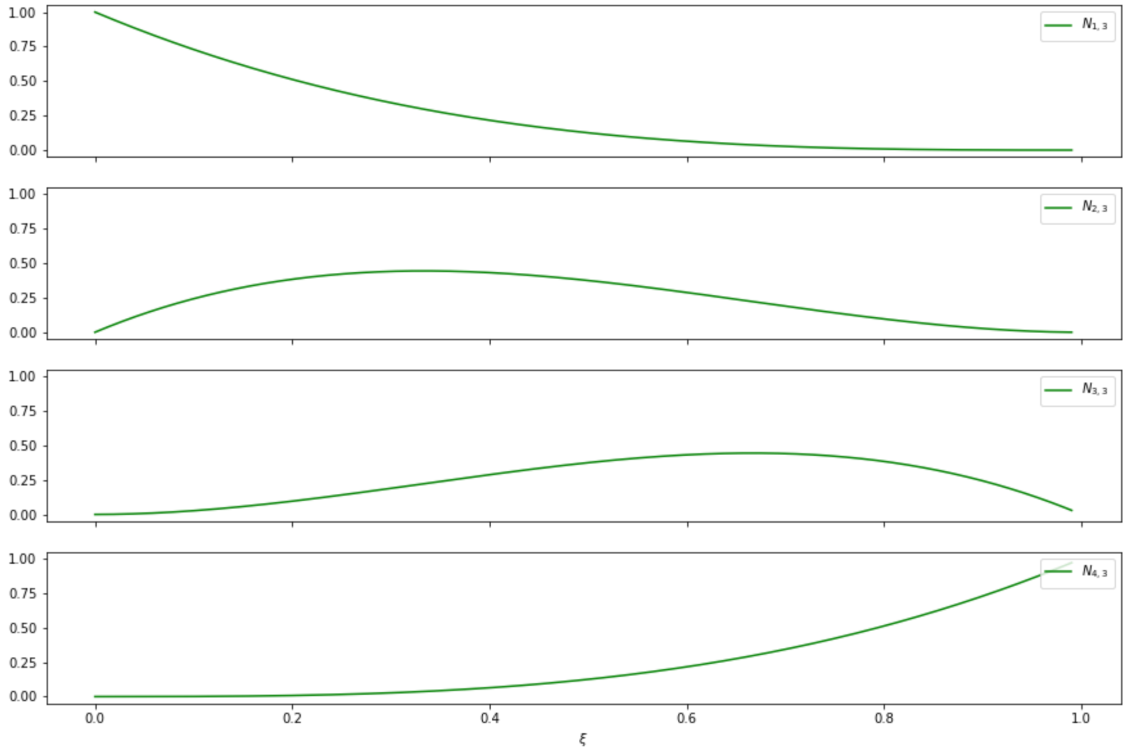


Figure 2.14: Basis functions for degree 3 ($N_{q,3}$, q = 1,2,3,4).

Figure 2.13 and figure 2.14 present an example of NURBS basis functions $N_{q,p}$ of order 2 and 3 for the chosen uniform knot vector Ξ.

The third curve is of degree 5 and have six control points (Cpt):

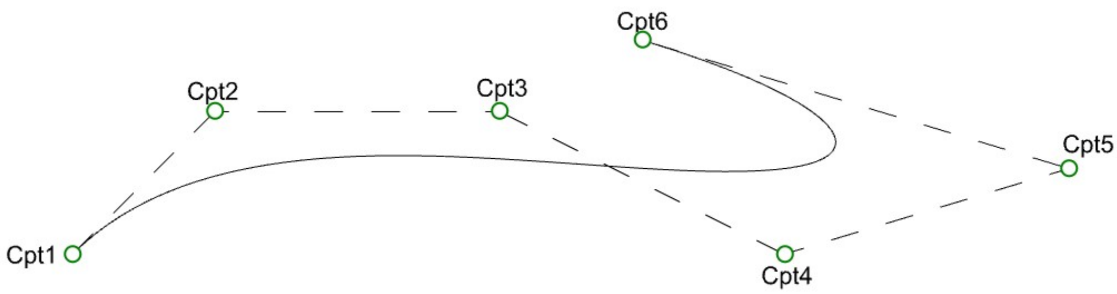


Figure 2.15: NURBS curve with six control points of degree 5.

The knot vector is as follows:

$$\Xi = [0,0,0,0,0,0,1,1,1,1,1,1]$$

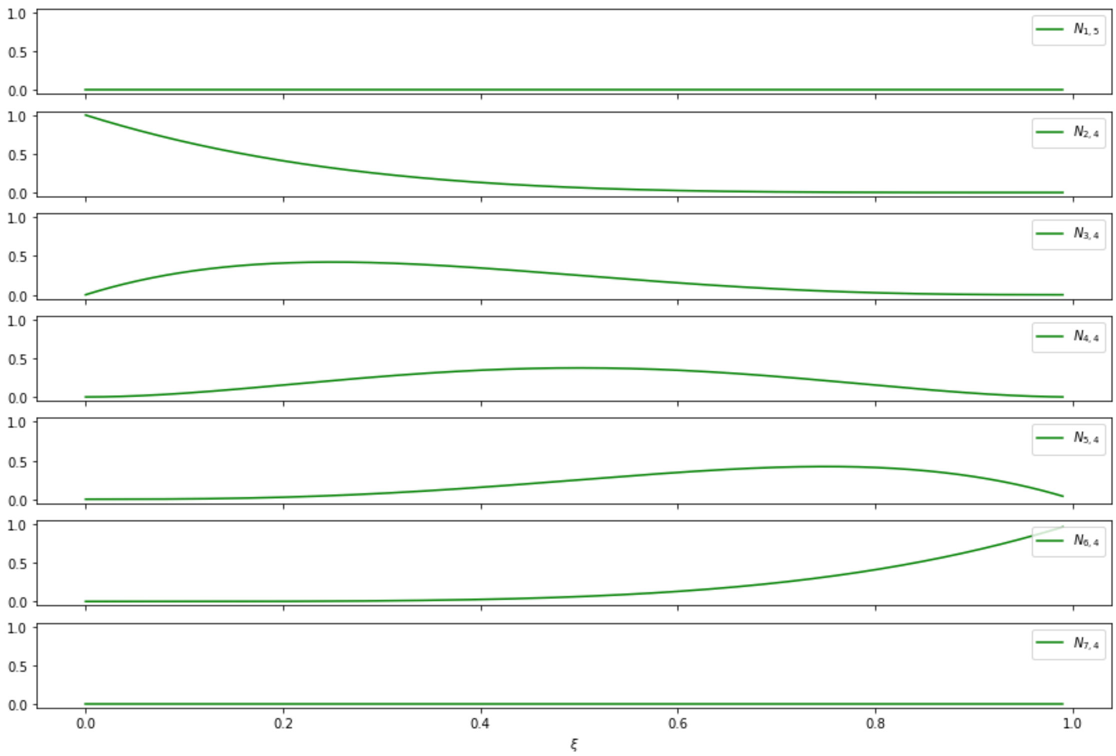


Figure 2.16: Basis functions for degree 4 ($N_{q,4}$, q = 1,2,3,4,5,6,7).

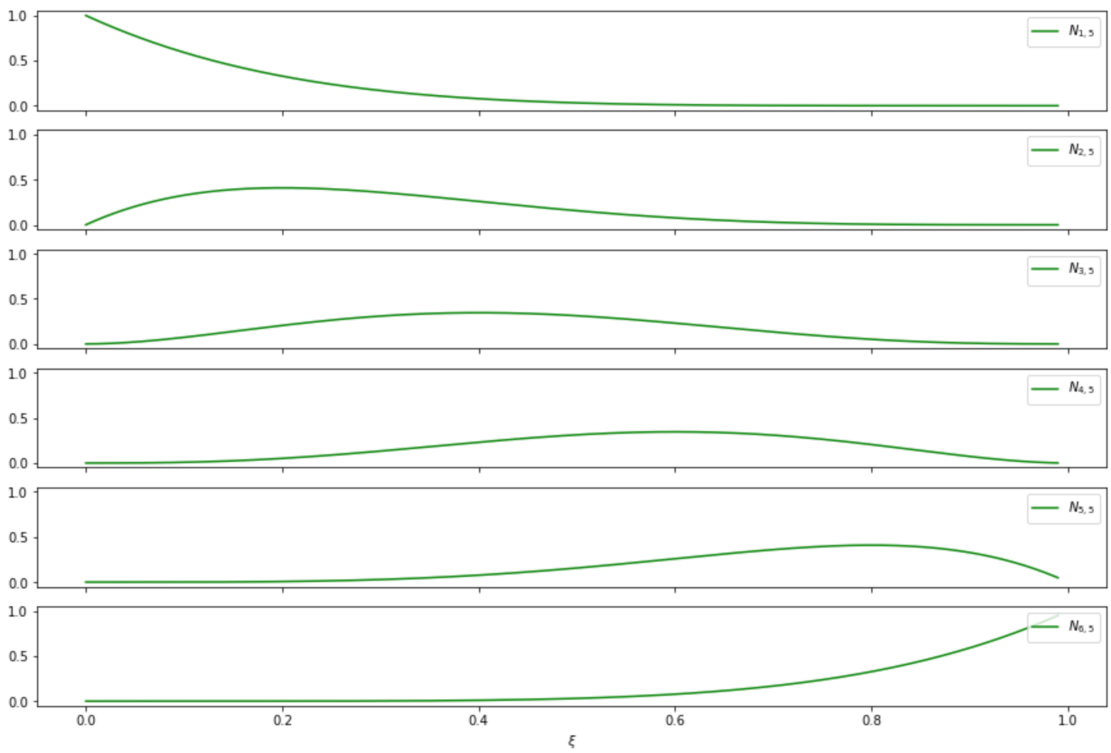


Figure 2.17: Basis functions for degree 5 ($N_{q,5}$, q = 1,2,3,4,5,6).

Figure 2.16 and figure 2.17 present an example of NURBS basis functions $N_{q,p}$ of order 4 and 5 for the chosen uniform knot vector Ξ.

2.4.3 Numerical integration

In the IGA approach to structural analysis, the NURBS basis functions are used to interpolate both geometry and the assumed solution fields (Zhang et al., 2016). NURBS are presented by high-order polynomials. Therefore, it can be difficult if not impossible to derive expressions of the integrand matrix for isogeometric elements (Bell, 2013). Numerical integration is therefore an important aspect of the Isogeometric approach to the FEM.

2.4.3.1 Gauss-Legendre Quadrature

Gauss-Legendre quadrature is a numerical method for approximating the definite integral of a function (Cotterell et al., 2009). For this numerical method the interval of the integral has to be $[-1,1]$. The formula is like this (Bell, 2013):

$$I = \int_{-1}^{1} g(\xi)\, d\xi \approx \sum_{k=1}^{n} w_k g(\xi_k) \tag{2.33}$$

where n is the number of Gauss points, w_k is the weight and ξ_k is the coordinate of the Gauss point k. w_k and ξ_k are dependent of the order of the Legendre polynomial $P_n(\xi)$ that is used. ξ_k is the root of the Legendre polynomial and w_k is the weight scaling the value of P_n evaluated in that specific coordinate. w_k is determined by this formula:

$$w_k = \frac{2}{(1-\xi_k^2)[P_n'(\xi_k)]^2} \tag{2.34}$$

The position and weight for the Gauss-Legendre quadrature rule for a normalized region are tabulated for many values of n in a number of books. Weights and coordinates for $n = 1,2,3,4,5$ are presented in Table 2.1.

Table 2.1: Legendre polynomial (P_n) points (ξ_k) and weights (w_k) for n = 1,2,3,4,5.

Number of Gauss-points, n	**Point,** ξ_k	**Weight,** w_k
1	0	2
2	$\pm\frac{1}{\sqrt{3}}$	1
3	$0\,,\pm\sqrt{\frac{3}{5}}$	$\frac{8}{9}\,,\,\frac{5}{9}$
4	$\pm\sqrt{\frac{3}{7}-\frac{2}{7}\sqrt{\frac{6}{5}}},\pm\sqrt{\frac{3}{7}+\frac{2}{7}\sqrt{\frac{6}{5}}}$	$\frac{18+\sqrt{30}}{36}\,,\,\frac{18-\sqrt{30}}{36}$
5	$0,\pm\frac{1}{3}\sqrt{5-2\sqrt{\frac{10}{7}}},\pm\frac{1}{3}\sqrt{5+2\sqrt{\frac{10}{7}}}$	$\frac{322+13\sqrt{70}}{900}\,,\,\frac{322-13\sqrt{70}}{900}$

Gauss quadrature provides an exact integration for a polynomial degree of 2n-1 with the use of n sampling points.

2.4.4 Isogeometric Analysis

Isogeometric analysis uses the exact geometry to analyse the structure. This is possible through the use of NURBS. NURBS is described with the use of basis function, which is also the foundation for the numerical analysis. The DOFs lie in the control points of the structure (Kiwi3d, 2022). Refinement of the structure can be done before and during analysis. During analysis the input for element division and polynomial degree is changeable by number.

2.4.4.1 Jacobian

To map the parent element into the physical domain a Jacobian is necessary (Dhatt et al., 2012). In order to do the integration over the basis functions, the parent element of the patch has to be in the parametric domain. This means the element is getting a length of -1 to 1 instead of the physical length. For a 1D problem the Jacobian is equal to the length of the structural member. The formula is as follows:

$$J = \frac{dC}{d\xi} = \sqrt{\left(\frac{dx}{d\xi}\right)^2 + \left(\frac{dy}{d\xi}\right)^2} \tag{2.35}$$

where C is the length of the curved member, as follows:

$$C(\xi) = \int_{\xi_0=-1}^{\xi_0=\xi} \sqrt{\left(\frac{dx}{d\xi_0}\right)^2 + \left(\frac{dy}{d\xi_0}\right)^2} d\xi_0 \tag{2.36}$$

2.4.4.2 Beam element

For the isogeometric approach, the coordinates of the geometry of the beam can be represented by using the NURBS, by the following formula (Gan, 2018):

$$x = \sum_{i=0}^{n} \mathbf{S}_{i,p}(\xi)\mathbf{P}_{xi} \qquad -1 \leq \xi \leq 1 \tag{2.37}$$

where $\mathbf{S}_{i,p}(\xi)$ are the NURBS basis functions, see section 2.4.1.2, and $\mathbf{P}_{xi}$ are the x-coordinate of the control points. The nodal displacements and rotations are treated independently by using each rational basis B-spline functions, for the isogeometric approach.

Element stiffness matrix for a straight Euler-Bernoulli beam element is given as:

$$\mathbf{K} = \begin{bmatrix} \mathbf{k}_{11} & 0 \\ 0 & \mathbf{k}_{22} \end{bmatrix} \tag{2.38}$$

where

$$\mathbf{k}_{11} = \sum_{j=1}^{ngu} \Big[\frac{EA}{J_j^2}\big(\sum_{i=0}^{nu} S_{i,pu}^1(\xi_j)\big)^T\big(\sum_{i=0}^{nu} S_{i,pu}^1(\xi_j)\big)\Big] \times Det(J_j)w_j \tag{2.39}$$

$$\mathbf{k}_{22} = \sum_{j=1}^{ngv} \Big[\frac{EI}{J_j^4}\big(\sum_{i=0}^{nv} B_{i,pv}^2(\xi_j)\big)^T\big(\sum_{i=0}^{nv} B_{k,pv}^2(\xi_j)\big)\Big] \times Det(J_j)w_j \tag{2.40}$$

The upper indexes are indicating the derivative of the function. *ngu* and *ngv* indicate number of Gauss points for DOFs of different polynomial describing the field values, while *nu* and *nv* indicate number of basis functions. $B_{i,pv}(\xi_j)$ are the *selective basis functions*. These are taken from the series of rational basis B-spline functions of the NURBS and customized to fit the classical shape functions. Selective basis functions are nessesary to apply the rational basis B-spline functions for the vertical displacement and rotations. $B_{i,pv}(\xi_j)$ are given as:

$$\begin{aligned}
B_{0,pv}(\xi_j) &= S_{0,pv}(\xi_j) + S_{1,pv}(\xi_j)\\
B_{1,pv}(\xi_j) &= S_{2,pv}(\xi_j) + S_{3,pv}(\xi_j)\\
B_{2,pv}(\xi_j) &= L/3S_{1,pv}(\xi_j)\\
B_{3,pv}(\xi_j) &= -L/3S_{2,pv}(\xi_j)
\end{aligned} \tag{2.41}$$

For a curved Euler-Bernoulli beam element, the geometry can be explained as follows:

$$x = \sum_{i=0}^{n} \mathbf{S}_{i,p}(\xi)\mathbf{P}_{xi} \qquad y = \sum_{i=0}^{n} \mathbf{S}_{i,p}(\xi)\mathbf{P}_{yi} \qquad -1 \leq \xi \leq 1 \tag{2.42}$$

The element stiffness matrix for curved Euler-Bernoulli theory beam element (Gan, 2018):

$$\mathbf{K} = [\mathbf{k}_{uv} + \mathbf{k}_{\theta}] \tag{2.43}$$

where:

$$\begin{aligned}
\mathbf{k}_{uv} = \sum_{j=1}^{ngu=ngv} \Big[&\frac{EA}{J_j^2}\big(\sum_{k=0}^{nu} B_{k,pu}^1(\xi_j)\big)^T\big(\sum_{k=0}^{nu} B_{k,pu}^1(\xi_j)\big)\\
&+ \frac{EA}{J_j R(\xi_j)}\big(\sum_{k=0}^{nv} B_{k,pv}(\xi_i)\big)^T\big(\sum_{k=0}^{nv} B_{k,pv}(\xi_j)\big)\Big] \times Det(J_j)w_j
\end{aligned} \tag{2.44}$$

$$\mathbf{k}_{\theta} = \sum_{j=1}^{ng\theta} \Big[\frac{EI}{J_j^2}\big(\sum_{k=0}^{n\theta} B_{k,p\theta}^1(\xi_j)\big)^T\big(\sum_{k=0}^{n\theta} B_{k,p\theta}^1(\xi_j)\big)\Big] \times Det(J_j)w_j \tag{2.45}$$

3 Software

To create FEA solvers for a parametric environment, five software programs have been used.

3.1 Rhinoceros3D

Rhinoceros3D, or Rhino, is a CAD software presented in a 3D environment for defining complex structures. The software is used for design, modeling, and analysing, developed by Robert McNeel & Associates (McNeel, 2022). Rhino works with NURBS as a mathematical approach to define curves.

3.1.1 Grasshopper3D

Grasshopper3D, or Grasshopper, is a graphical algorithm editor created to develop parametric structures, and is a plug-in to Rhino (Davidson, 2022). Grasshopper uses an algorithmic approach to model geometry. This is AAD, as described in section 2.2. The program is integrated with Rhino. Therefore the geometry can be obtained from Rhino modeling tools and adjusted with Grasshopper, or it can be modeled parametricly in Grasshopper. By implementing the Galapagos Evolutionary Solver within Grasshopper, structures can be optimized as a part of AAD.

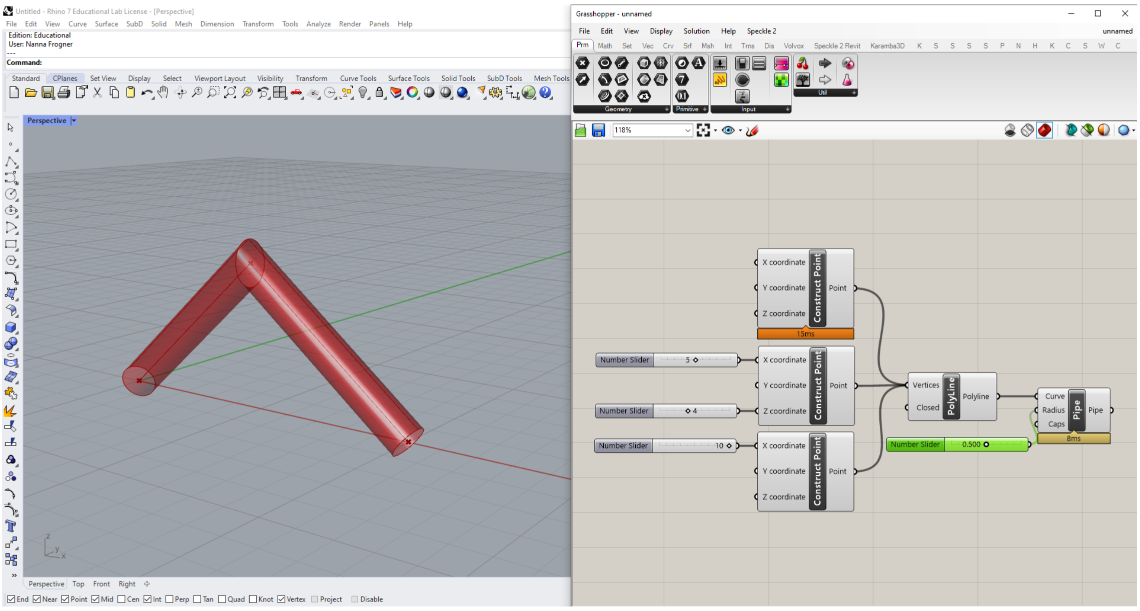


Figure 3.1: Parametric geometry created with Grasshopper in Rhino.

Grasshopper is integrated with a parametric environment that uses a visual programming language. Geometry and analysis are created by dragging boxes or "components" onto a clear canvas. The components connect through "wires" to convey information, and value inputs can easily be changed with a slider, as shown in Figure 3.1.

3.1.1.1 Karamba3D

Karamba3D, or Karamba, is a structural engineering tool for a parametric environment. The parametric finite element program provides accurate analysis of frames, spatial trusses, and shells. The beam theory is based on Timoshenko theory (Karamba3d, 2022). Karamba is a plug-in for Rhino and is fully cooperative with the parametric environment of Grasshopper. This cooperation enables the geometry from Rhino/Grasshopper to be implemented directly into Karamba, which means that the geometry does not need to be transferred between softwares. Hence, properties and restrictions can be added directly to the geometry with components from Karamba, which takes into use the same visual programming language as Grasshopper.

3.1.1.2 Kiwi!3D

Kiwi!3D, or Kiwi, is a plug-in for Rhino and cooperative with Grasshopper as a parametric structural analysis program (Kiwi3d, 2022). The first Beta release was 31.01.2020. The software is based on IGA to integrate structural analysis into CAD. The NURBS-based analysis tool uses basic functions to describe shape function for analysis and geometry.

3.2 Microsoft Visual Studio

Microsoft Visual Studio (VS) is used to develop different type of Software (Studios, 2022). It offers C# as a programming language, that works good for developing software. Furthermore, it is fully packed with a array of tools and features to elevate and enhance every stage of software development. *MathNet.Numerics* is a numerical library that provides methods and algorithms for numerical computations in science, which VS offers (Math.NET, 2022).

4 Methodology

4.1 General

The goal for this master's thesis was to examine the differences between two approaches of analysing geometry; FEA and IGA. The former is an established structural analysis, defended by numerous academics. Whereas IGA is newly incorporated as a structural analysis. To make a fair comparative study of the two softwares, a FEM solver was implemented as a plug-in to Grasshopper. In addition, a NURBS component for generating geometry as input for IGA was created. The algorithms were constructed in VS using the programming language C#. The implemented FEA code was examined against Kiwi, which is an already implemented IGA in Grasshopper. Kiwi is newly established, compared to Karamba that is a grounded FEA software in Grasshopper. Karamba is frequently applied to validate the FEM solver.

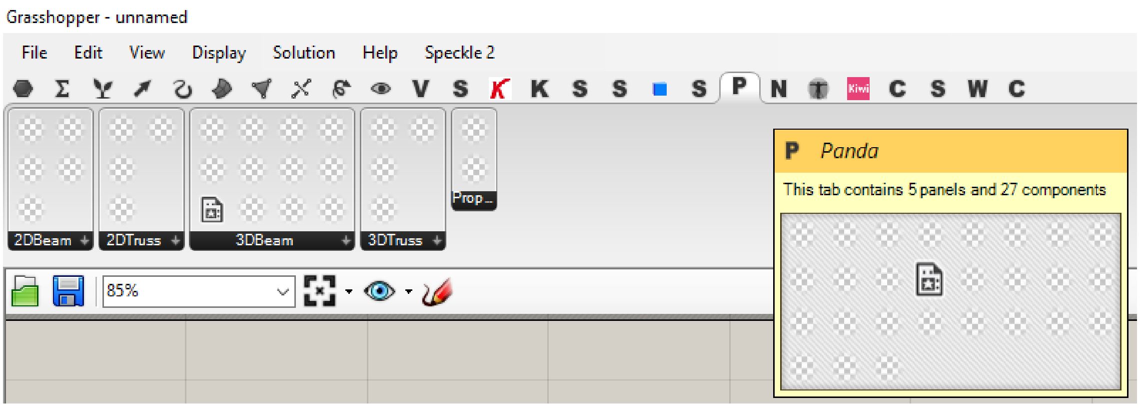


Figure 4.1: Tabs in Grasshopper, and components in Panda.

Figure 4.1 presents the structure of the plug-in in Grasshopper, which is named *Panda*. Panda is a set of 27 components that works together.

The knowledge of programming in C# and how to compose a FEA software were narrow. Starting out simple became a natural choice, to develop a profound code and to achieve a solid understanding of the basis of the FEM. Therefore, a FEA for truss elements in a 2D environment were the starting point. When this analysis returned results within accepted tolerance, the component was developed further.

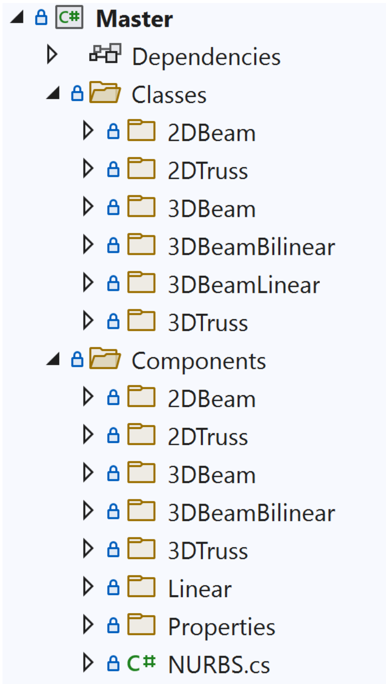


Figure 4.2: Overview of components and classes in VS.

An overview of the components created in VS is presented in Figure 4.2. Classes are used in order to assign several properties for objects. This was handy concerning information transfer between components and the organization of the geometry (elements and nodes) with indexes for the FEM. From Figure 4.2, it appears that after making components for 2D truss, the same procedure was made for 3D Truss and 2D Beam until we ended up with a 3D Beam FEA. The analysis for a 3D beam environment is the most advanced, and will therefore be described more in Section 4.2.

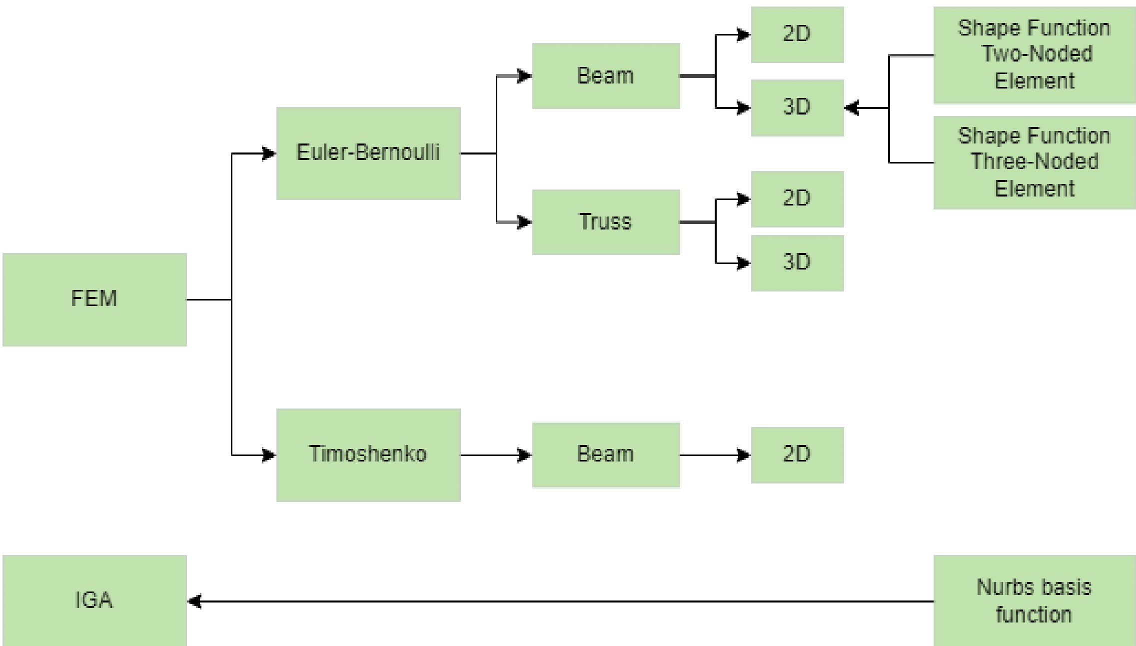


Figure 4.3: Overview of versions of solvers.

Figure 4.3 gives an overview of the different analysis components that are established. Euler-Bernoulli, as described in section 2.3.2, has been the main beam theory for FEM used in this master's thesis. Beam theory in Karamba is based on Timoshenko theory and for Kiwi it is built on Euler-Bernoulli theory. As a consequence, we created a Timoshenko assembly from theory in section 2.3.3 for comparison. This solver is only capable for 2D geometry, as it should only be compared with the simplest benchmarks of a cantilever and an arc, see section 5.2 and 5.3. At last, a component for NURBS, (see section 2.4.1), were developed to generate NURBS as input for Kiwi.

To confirm the behaviour of the FEM solver, different structures were analysed and compared with Karamba. Nodal values, displacement pattern, moment diagram and support values were validated. These assessments indicate that the stiffness matrix and shape functions are implemented correctly. Furthermore, the controls establish that the analysis is consistent for element division and organization.

Units of the components are Newton [N] and millimeters [mm]. The only exception is the length of the geometry, which should be given in meters [m].

4.2 Organization of solver

4.2.1 Euler-Bernoulli FEM solver

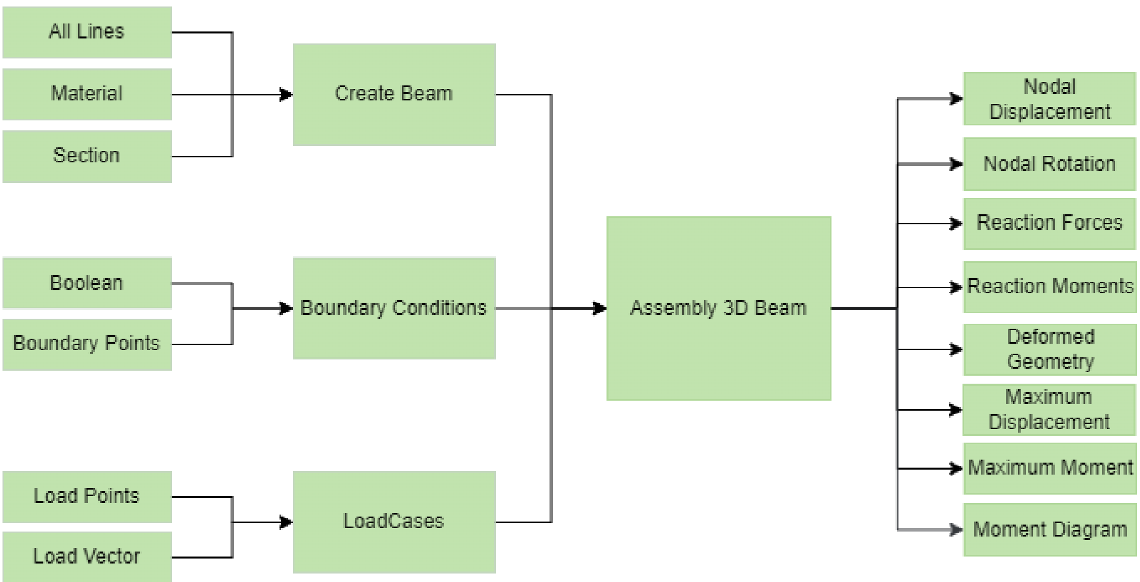


Figure 4.4: Flowchart presenting the structure of components in Grasshopper.

A simplified organization of the components in Grasshopper is showed in Figure 4.4. The *Assembly 3D Beam* is the main component where the analysis is to a desirable format for the analysis. *Create Beam* assigns a section class and a material class to the geometry. The beam class is used to specify indexes for each element, and the element nodes in the global system. The output is a Beam Class Element, containing all the given info about the beam.

Classes were used for *Boundary Conditions* and *Load Case* as well, to transfer both the position and the value of the loads and the boundary conditions. The output is nodal values (displacement and rotation), reaction values, deformed geometry (and maximum value) and moment diagram (and maximum value). See Appendix A for the complete script of all solvers. A detailed description of the components will be given in the following.

4.2.1.1 Assembly 3D Beam

Assembly 3D Beam is the component where the analysis is performed. First, a list of points is made from the nodes of the beam class objects. These points have indexing properties from the node class, in order to obtain a mathematical structure that represent the geometry. A structure like this is necessary when establishing the global stiffness matrix, in order to get the connectivity right.

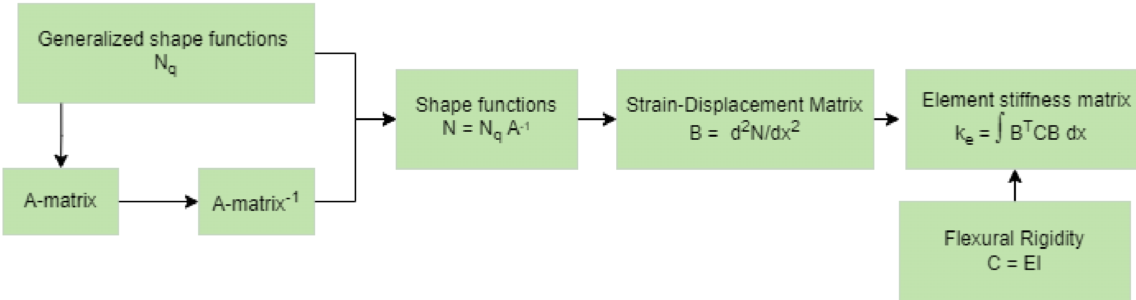


Figure 4.5: Flowchart describing how to establish the element stiffness matrix $\mathbf{k}_e$.

The local stiffness matrix $\mathbf{k}_e$ for each element is established as presented in figure 4.5. This is based on theory from section 2.3.2.1 and 2.3.2.3, and is prepared in MathCad before it is implemented in the FEM solver. The shape functions used are:

$$\mathbf{N_q} = [1 \quad x \quad x^2 \quad x^3], \qquad \text{for two-noded element}$$

$$\mathbf{N_q} = [1 \quad x \quad x^2 \quad x^3 \quad x^4 \quad x^5], \qquad \text{for three-noded element}$$

For the Euler-Bernoulli FEA solvers, structural elements with respectively two and three nodes have been developed. The former will be addressed as *FEM solver 1*, and the latter *FEM solver 2*. For FEM solver 2, the nodes are located at start, mid and end point. The location of the nodes establish the A-matrix, as described in section 2.3.2.1. The three-noded element was implemented to emphasis that FEM elements can not imitate a curved structure individually, independent of number of nodes. An angular distorted three-noded element can neither be mapped nor have DOF transformation with the existing beam theory. This is validated in the benchmark for the curve, see 5.3. Therefore, the DOFs in the nodes of the three-noded element is transformed from local to global coordinates with the same transformation as for the two-noded element.

Figure 4.6: Flowchart describing how to establish the Total Stiffness Matrix.

```
    Vector3d unitX = new Vector3d(1, 0, 0);
    Vector3d unitY = new Vector3d(0, 1, 0);
    Vector3d unitZ = new Vector3d(0, 0, 1);

    Vector3d nullvec = new Vector3d(0, 0, 0);
    Vector3d vec1x = b.axis.TangentAt(0);
    Vector3d vec1y = new Vector3d(-vec1x.Y, vec1x.X, 0);
    Vector3d vec1z = Vector3d.CrossProduct(vec1y, vec1x);

    vec1z.Unitize();
    vec1x.Unitize();
    vec1y.Unitize();

    double _1l1 = Math.Cos(Vector3d.VectorAngle(vec1x, unitX));
    double _1m1 = Math.Cos(Vector3d.VectorAngle(vec1x, unitY));
    double _1n1 = Math.Cos(Vector3d.VectorAngle(vec1x, unitZ));

    double _1l2 = Math.Cos(Vector3d.VectorAngle(vec1y, unitX));
    double _1m2 = Math.Cos(Vector3d.VectorAngle(vec1y, unitY));
    double _1n2 = Math.Cos(Vector3d.VectorAngle(vec1y, unitZ));

    double _1l3 = Math.Cos(Vector3d.VectorAngle(vec1z, unitX));
    double _1m3 = Math.Cos(Vector3d.VectorAngle(vec1z, unitY));
    double _1n3 = Math.Cos(Vector3d.VectorAngle(vec1z, unitZ));

    Matrix<double> t = DenseMatrix.OfArray(new double[,]
    {
            {_1l1,  _1m1,    _1n1},
            {_1l2,  _1m2,    _1n2},
            {_1l3,  _1m3,    _1n3},
    });
```

Listing 1: Finding angles for transformation of DOFs in 3D beam environment.

```
int node1 = b.startNode.Id;
int node2 = b.endNode.Id;

for (int i = 0; i < K_eG.RowCount / 2; i++)
{
    for (int j = 0; j < K_eG.ColumnCount / 2; j++)
    {
        K_tot[node1 * 6 + i, node1 * 6 + j] += Math.Round(K_eG[i,j],5);
        K_tot[node1 * 6 + i, node2 * 6 + j] += Math.Round(K_eG[i,j+6],5);
        K_tot[node2 * 6 + i, node1 * 6 + j] += Math.Round(K_eG[i+6,j],5);
        K_tot[node2 * 6 + i, node2 * 6 + j] += Math.Round(K_eG[i+6,j+6],5);
    }
}
```

Listing 2: Using a double *for loop* to substitute for the connectivity matrix *a*.

Figure 4.6 visualize how the total stiffness matrix for the structure is established, based on theory described in section 2.3.2.5. The local stiffness matrix is converted to global coordinates through the transformation matrix for 3D beam described in section 2.3.2.4. The angles in this matrix is identified like presented in Listing 1. The transformation Δ is stacked twice for each node on the element in the total transformation matrix T. To satisfy the connectivity of the system, a double for loop is used, see Listing 2. This should represent the a-matrix on Figure 4.6.

```
for (int i = 0; i < _BC.Count; i++)
{
    int sizeOfMatrix = K_tott.ColumnCount;
    int blockedIndex = _BC[i];

    Rvec[blockedIndex] = 0;

    for (int r = 0; r < sizeOfMatrix; r++)
    {
       K_tott[blockedIndex, r] = 0;
       K_tott[r, blockedIndex] = 0;

       if (K_tott[r, r] == 0)
        {
            K_tott[r, r] = 1;
        }
    }
}
K_red = K_tott;
```

Listing 3: Reducing the stiffness matrix and load vector to only solve the equations for active DOFs.

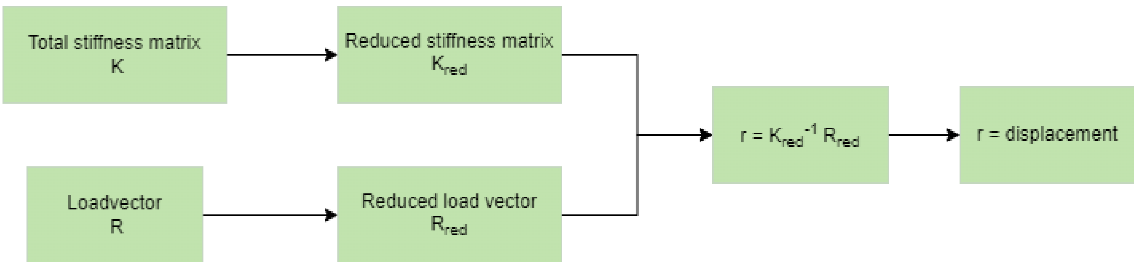


Figure 4.7: Flowchart over Code for displacement.

To only solve the structural system (Equation 2.1) where it is of interest to find the nodal values, both the stiffness matrix (**K**) and the load vector (**R**) is reduced to only include active DOFs. This is done by replacing existing values with 0 (and 1 on the diagonals) in **K** at the indexes for fixed DOFs (see Listing 3). An overview of the process is expressed in Figure 4.7.

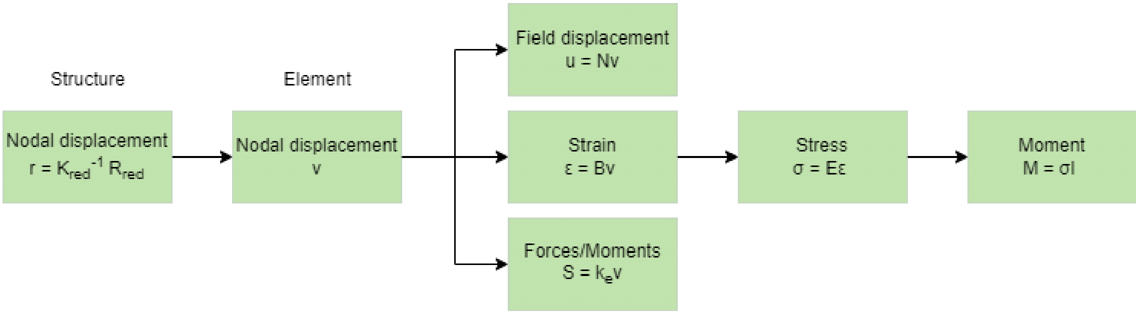


Figure 4.8: Field displacements and moment diagram established from nodal displacements and shape functions.

In Figure 4.8, the approach of calculating field displacement, nodal forces and moment diagram from nodal displacements and shape functions is presented. Reaction forces are identified by the global version of Equation 2.2. Finding the moment diagram through this process of finding field values, is the way of controlling whether the stress and strain in the FEM solver is correct. This is because Karamba is not presenting stress and strain for beam analysis.

4.2.1.2 Create Beam

There is an individual component for creating a structural object from lines in the geometry. A beam class object is the output, with the properties of a material class, section class, axis of the specific line, and nodes. First, the component assigns a material and section for the geometry. Both the material and section are classes, where the former transfer properties of density ρ, Young's modulus E, Poisson's ratio μ and shear modulus G. The latter set the section area A, moment of inertia I for both y- and z- axis and polar moment J. The *create beam* component is also giving an index to the beam class objects. A point list is made from the nodes of the axes of the beam. These points are made into node classes, in order to have the properties of both a coordinate and an index. Due to this, the nodes of the beams are structurally indexed to satisfy the connectivity of the structure.

4.2.2 Load Case

In the component for creating a load case, coordinates of a point and a vector describing the load value in x-,y- and z-directions are assembled as properties in a *load class*. The coordinate is later used in the model assembly component to compile with the list of nodes, to determine the positioning of the load values in the load vector R.

4.2.2.1 Boundary Conditions

In the component for boundary conditions, coordinates of support points and boolean are input. For each point, a boolean is used to determine whether the DOFs in that point are fixed (True) or free (False). Coordinates of the point and a boolean for each DOF are compiled in a *boundary condition class*.

4.2.3 Timoshenko FEM solver

A FEM solver based on Timoshenko theory was developed. This was done to examine whether it gave different results from Euler Bernoulli in the comparison with Karamba.

The generalized shape functions are still $N_q = [1 \quad x \quad x^2 \quad x^3]$. Displacement and rotation in the A-matrix is then established after equation 2.25 and 2.23. A is established in the same way as for Euler-Bernoulli. The generalized stiffness matrix k_q is establish by the principle of virtual displacements. From this the stiffness matrix is constructed after equation 2.26.

4.2.4 NURBS component

The aim for this component was to create NURBS, as input for corresponding IGA capable of calculating exact curved geometry with few elements (see section 2.4). The NURBS component is built after the theory in section 2.4.1. Grasshopper already has a built in algorithm for generating NURBS. This will be referred to as integrated NURBS.

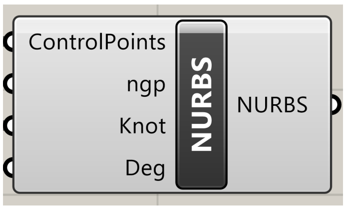


Figure 4.9: NURBS component.

Figure 4.9 shows how the component looks in Grasshopper. The script requires input of *Control Points*, *Number of Gauss points (ngp)*, *Knot vector (Knot)* and *Degree (Deg)*. These parameters have to be in accordance with each other for the script to calculate a NURBS curve. A NURBS is output, but only the *Curve Points* are visualized. These points are the coordinates of the Gauss points on the resulting NURBS curve. The equivalent algorithm in Grasshopper is capable of visualizing the resulting curve, but the NURBS component is mainly made for calculations.

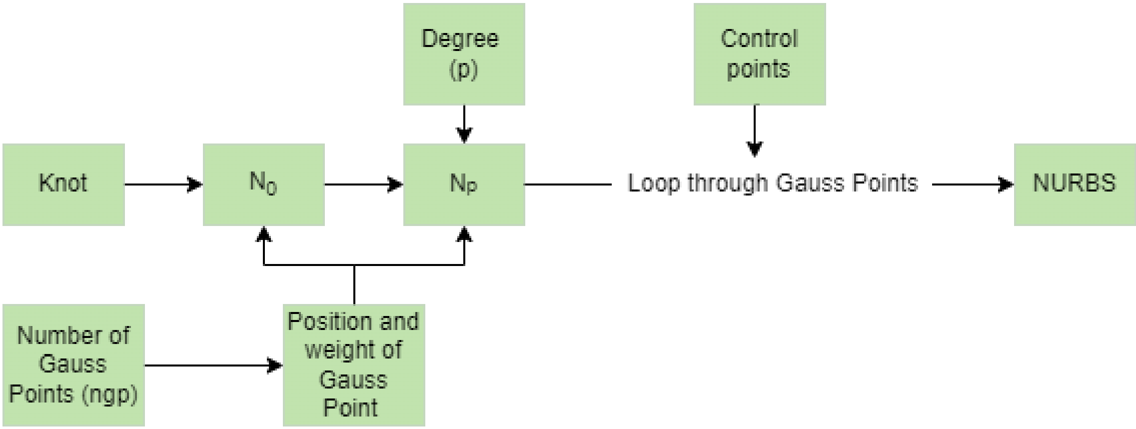


Figure 4.10: Overview of NURBS basis function script.

An overview of the structure of the NURBS script is presented in Figure 4.10. First, a function for returning properties for $n = 1,2,3,4,5$ Gauss points is made. The returning values are the weight and coordinate on a parameterized curve from -1 to 1, as presented in Table 2.1. Further, a function for creating NURBS basis function of degree 0 is made, after Equation 2.30 and 2.31. This is only dependent of the spacing in the knot vector, and the position of the Gauss point. A function for creating basis function of degree p is then made, based on Equation 2.32. This is a recursive function, meaning it is calling upon itself for $p > 1$. The NURBS is then generated by looping through the Gauss points and finding the location of each point on the resulting curve. This is done by scaling the influence of each control point with the shape function evaluated in that specific point. For the complete script, see Appendix A.

The geometry generated from the NURBS component will be input for Kiwi. Therefore, an example of how the NURBS component returns geometry will be presented in the following. This exemplification of geometry will be the basis of the benchmark in Section 5.3.

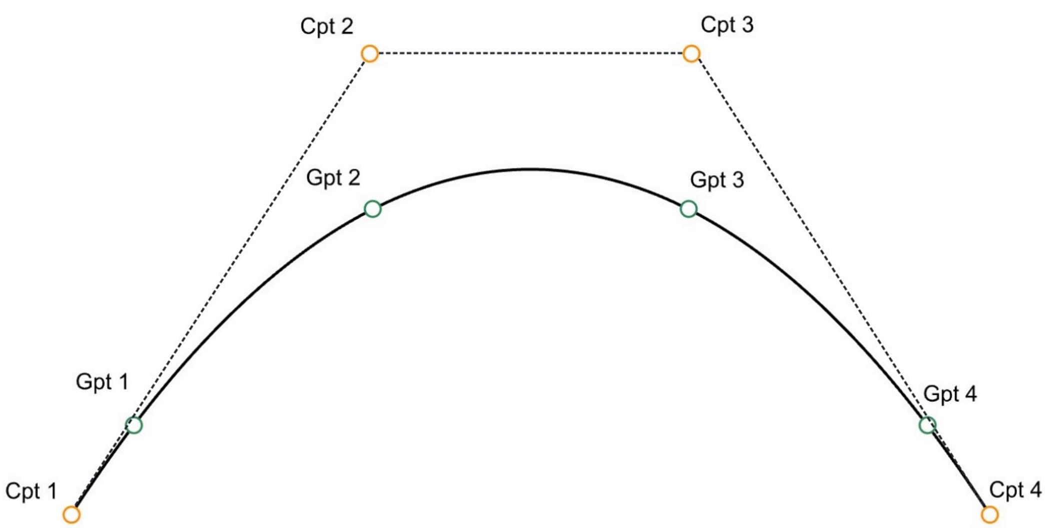


Figure 4.11: Control Points and Curve Points on NURBS curve.

Table 4.1: Coordinates for Control Points.

Control Points	Coordinates
Cpt 1	(-2, 0, 0)
Cpt 2	(-0.7, 0, 2)
Cpt 3	(0.7, 0, 2)
Cpt 4	(2, 0, 0)

Figure 4.11 presents a geometry with a non-linear shape, constructed by the NURBS component. In addition, the integrated NURBS algorithm in Grasshopper is used for visualization. The curve is of degree 3 and is constructed with four control points, with coordinates shown in Table 4.1. Knot vector and Weights are as follows:

$$\Xi = [-1, -1, -1, -1, 1, 1, 1, 1] \qquad \text{weight} = [1, 1, 1, 1]$$

Each control point is weighted 1. The curve is considered as one patch with a knot vector containing n + p + 1 (= 8) values. Only one non-zero span in the knot vector indicates a curve of one element. Integrating the curve with Gauss-Quadrature rule with four points (Gpt), result in four points distributed along the curve after Table 2.1, as shown in Figure 4.11.

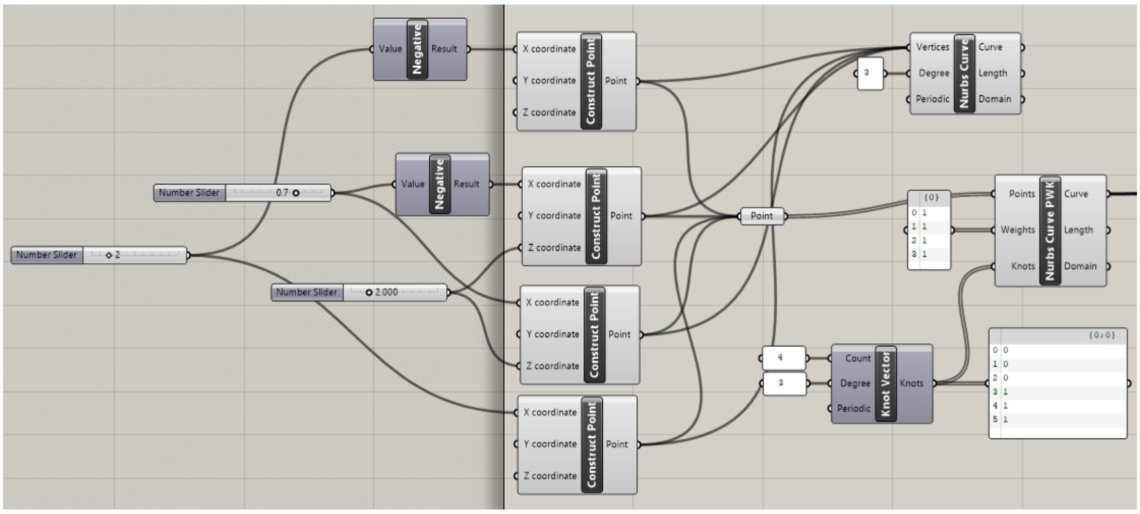


Figure 4.12: Integrated NURBS algorithm in Grasshopper for creating NURBS.

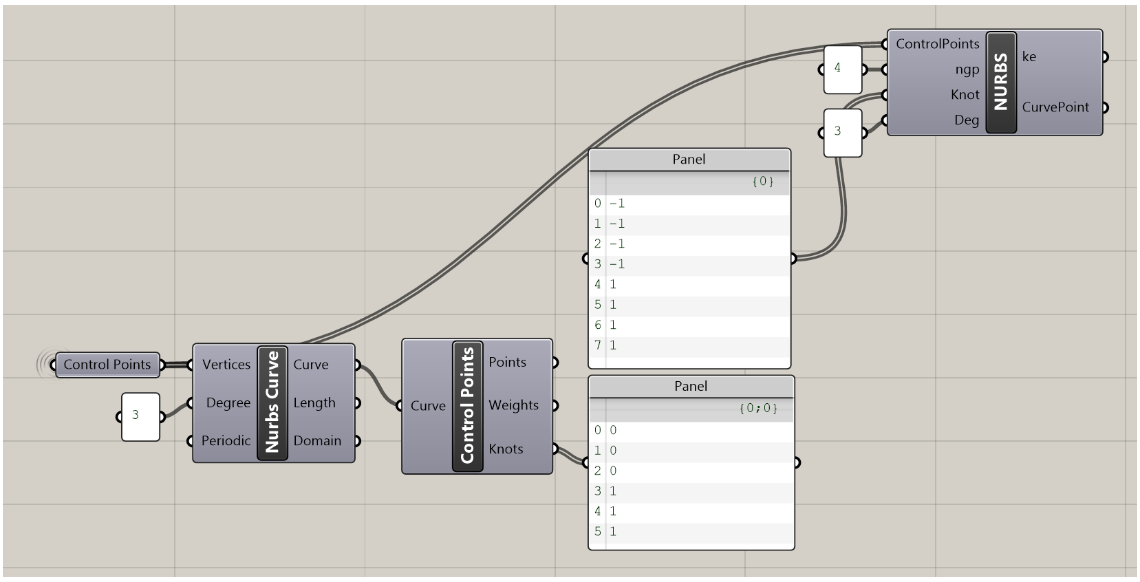


Figure 4.13: NURBS component generating NURBS.

Figure 4.12 and 4.13 present how the NURBS is constructed from integrated NURBS and NURBS component, respectively. In Figure 4.11 one can see how the NURBS component is visualizing the Gauss points on the curve, for calculation of a NURBS with four Gauss points. The NURBS component is generating a NURBS curve for calculation, but can not visualize it in Rhino. The NURBS component is using a parametric curve from -1 to 1, and is therefore using a different knot vector than integrated NURBS, where it goes from 0 to 1. The knot vectors are describing the same patch. Another difference is the length of the knot vector, as integrated NURBS is assuming that 0 and 1 should be added to the beginning and end, as the first and last knot value always should be duplicated.

4.3 Discussion

The structure of the FEM algorithms was developed rather fast. Although, after completing the script for all components, it took two months before the analysis worked for all benchmarks. Assembly components for 2D and 3D truss, and 2D beam were developed based on general FEM theory that is exemplified in literature, including visualization of matrices. Therefore, it was more trivial to implement and control essential aspects for the analysis. Developing the script for the 3D beam component became more complex. In literature, beam theory for a 3D environment is explained more briefly without examples. Therefore, establishing both the element stiffness matrix and identifying the rotations in the transformation matrix required considerably better understanding of the theory.

The development of the stiffness matrix required consistent attention to the Right Hand System (RHS). Primitively, the rotation around y- and z- axes were assumed to be similar. However, RHS constitutes opposite directions for positive rotation. The mistake was challenging to identify as the error appeared for specific cases. Furthermore, the localization of values in the element stiffness matrix were burdensome.

The stiffness matrix were established from sub matrices in MathCad for: axial, torsion, rotation around y-axis and displacement in z-direction, and rotation around z-axis and displacement in y-direction. The sub matrices were placed regarding the DOFs in the complete element stiffness matrix. In VS, values were placed at wrong matrix indexes and the rewriting included typos. In addition, to get the fractional calculation right, values consistently had to be implemented as *doubles*. Before this was perceived, many trivial mistakes resulted in fatal frustration.

Considering the transformation matrix, several construction methods from different sources were tested, but none returned accurate results for all benchmarks. Eventually, the transformation matrix was established from the principle of how a beam element rotates, with an assumption of no rotation around its own axis. Due to this assumption, the algorithm works exquisitely for circular section with moment of inertia independent of axis rotation. As a result, the x-axis is always along the beam's longitudinal axis. The y-axis will not have a component in the z-direction, based on our assumption, and is only depended on the x- and y-value in the x-axis vector. At last, the z-axis will be the cross-product of the x- and y-axis.

The NURBS component was established to achieve geometry for IGA. NURBS basis functions are the fundamental aspects for the IGA solver, since the structural analysis of IGA is performed directly on the geometry description and NURBS are describing the field values. Kiwi was used as IGA solver. By creating the NURBS component, accumulated knowledge about this approach to structural analysis were gained. This became handy in order to understand both the input and output values of Kiwi.

5 Benchmarks

The performance of the FEM solvers was controlled for both execution time and accuracy of the analysis. To validate the latter, two benchmarks were introduced: a cantilever and an arc. These benchmarks helped substantiate whether Kiwi was understood and used correctly.

5.1 Execution time

For a parametric environment, a short analysis time to keep up with the modifications in the model is desirable. The performance of execution time for the assembly components, where the analysis is performed, is therefore evaluated.

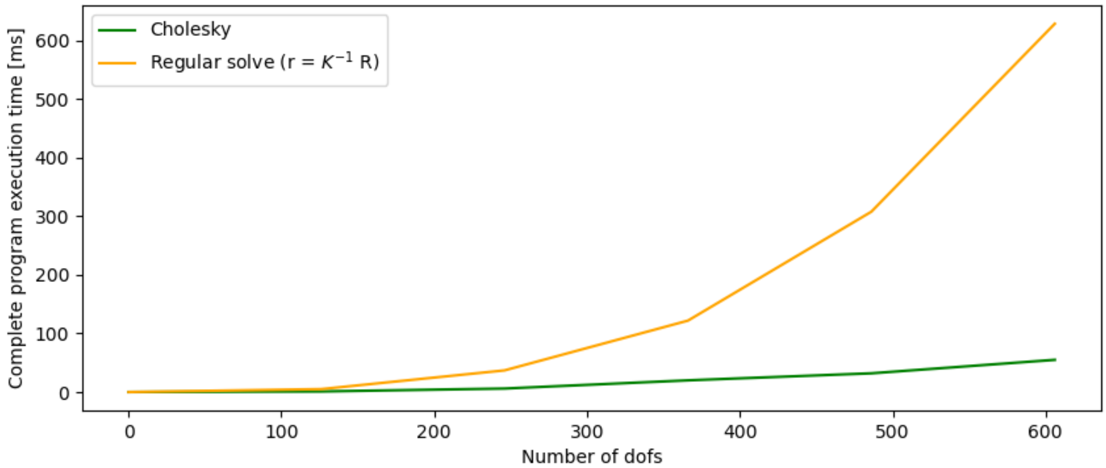


Figure 5.1: Execution time for Cholesky vs. Regular solver.

Equation 2.1, for calculation of nodal values of the structural system, requires the inverted stiffness matrix, see Figure 4.7. This operation is time consuming. *Cholesky factorization* is an built-in method in *MathNet.Numerics.LinearAlgebra*, see section 3.2, for solving systems of linear equations. Compared with the traditional method, Cholesky is rather efficient for numerical solutions. Figure 5.1 shows the execution time for the FEM solver 1 when using Equation 2.1 and Cholesky. It is clear that Cholesky is preferred.

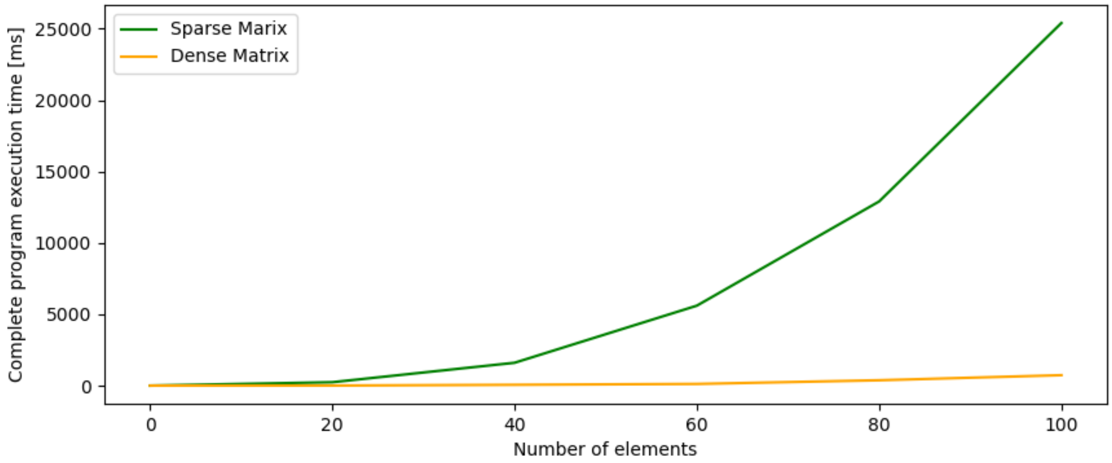


Figure 5.2: Execution time for Sparse vs. Dense Matrix.

The FEM is based on solving matrix equations. Therefore, how the matrices are stored is of influence for the calculation time. With *MathNet.Numerics.LinearAlgebra* matrices can be built as either *Dense* or *Sparse*. The former is storing values for all indexes, while the latter only considers the values different from zero and their index. Matrices in the FEM have mostly values on the diagonals, and are rather sparse. As a consequence, saving the matrices as *Sparse* would condense the script and reduce the required storage. Regardless, direct matrix decomposition methods, such as Cholesky, are optimized for *Dense* matrices only. Therefore, in the context of Cholesky, *Dense* matrices are to be preferred. This is validated by Figure 5.2, which presents the execution time for the FEM solver 1 for both *Sparse* and *Dense* matrices.

Accordingly, all solvers are based on Cholesky for solving linear equations and *Dense* storage of matrices.

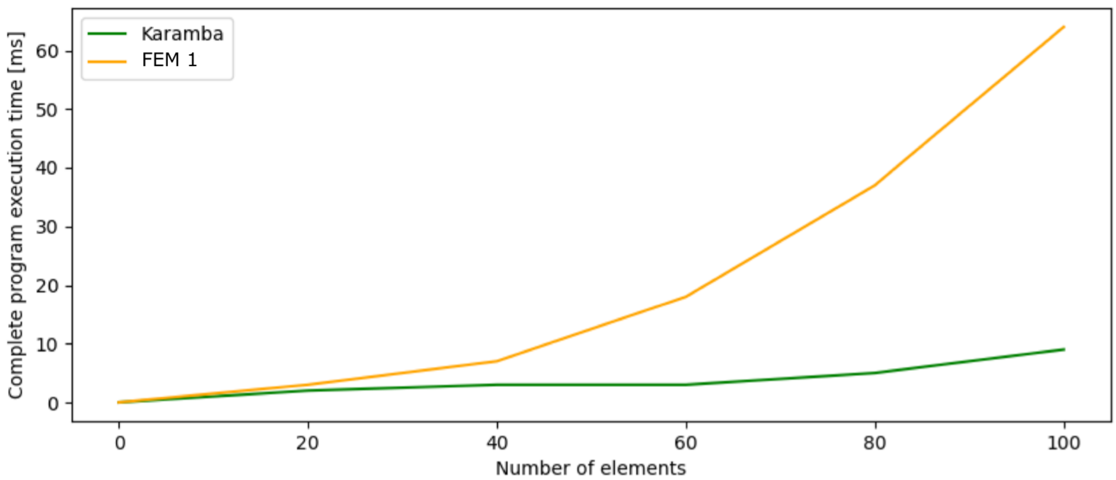


Figure 5.3: Execution time for Karamba and FEM solver 1.

Figure 5.3 presents the calculation time for the analysis component in Karamba and FEM solver 1. Karamba is faster, but Figure 5.3 indicates that the difference is not significant before passing 40 elements.

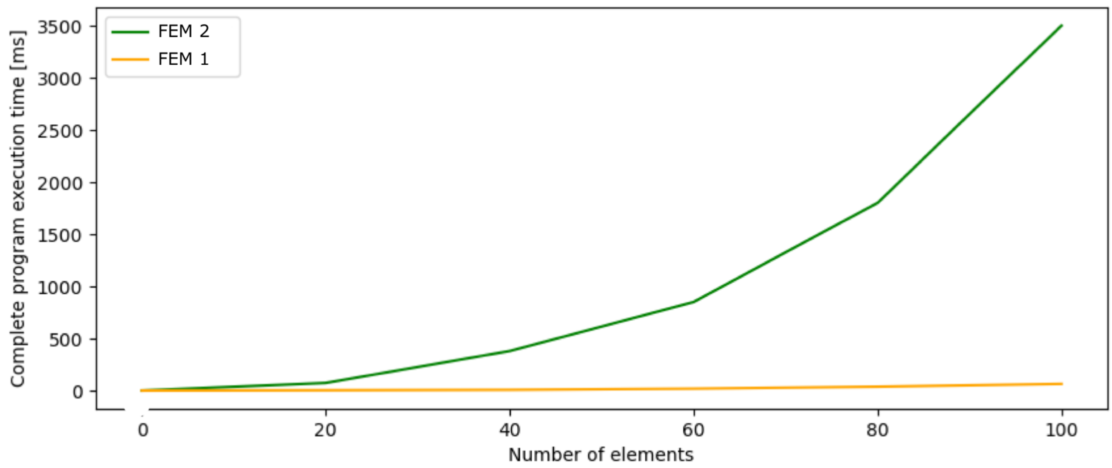


Figure 5.4: Execution time for FEM solver 2 and FEM solver 1.

Figure 5.4 presents the different execution time for FEM solver 2 and FEM solver 1, respectively. The solvers have the same structure in the script, but the former is using elements with an extra node in the midpoint. A significant difference in execution time can be observed, with FEM solver 1 being the most time efficient. Therefore, mainly results from FEM solver 1 is presented for the benchmarks and study cases.

Further performance of the FEM solvers will be evaluated through the benchmarks. Here the quality of the returning values of the solver will be presented and later evaluated.

5.2 Cantilever

A cantilever is the first benchmark. The purpose was to start with a simple task, to ascertain the consistency of the algorithm.

5.2.1 Geometry and structural model

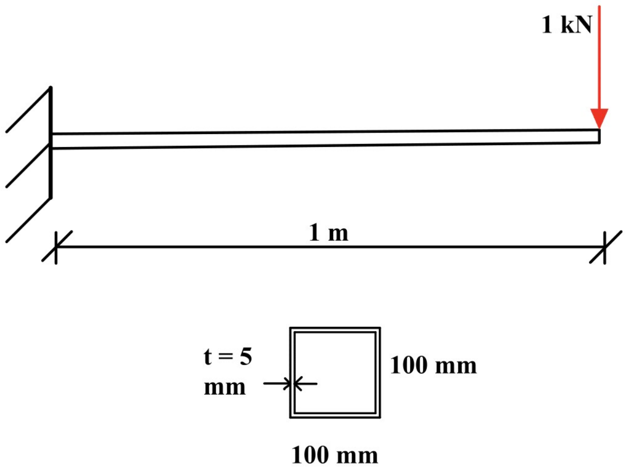


Figure 5.5: Geometry and structural model of the cantilever.

Figure 5.5 presents the geometry, the structural model, and the cross section of the cantilever. The beam is one meter long, loaded with 1 kN at one end, and is fixed at the other. The dimension of the cross section is 100x100x5 mm and is made of steel S355.

5.2.2 Results

The deflection (w) at the end of the cantilever is given by the strong form of the differential equation: $w = \frac{PL^3}{3EI}$, where I is the moment of inertia, E is the Young's modulus, P is the load and L is the length. This is the mathematical exact solution.

Table 5.1: Deflection values at the end of the cantilever.

Analyse Type	Deflection [mm]
Exact Solution	0.5539
FEM Solver 1	0.5539
Karamba	0.5670
Kiwi	0.5816
Timoshenko	0.6227

The displacement at the end of the cantilever was exact from FEM solver 1, see table 5.1. Small margins differ for results from Karamba, Kiwi and Timoshenko solver and the exact solution. Table 5.1 indicates that FEM solver 1 is more accurate than the Timoshenko solver. This is in accordance with the effort behind developing the Timoshenko solver compared with the FEM solver 1. Accordingly, the Timoshenko solver will not be presented in the further evaluation.

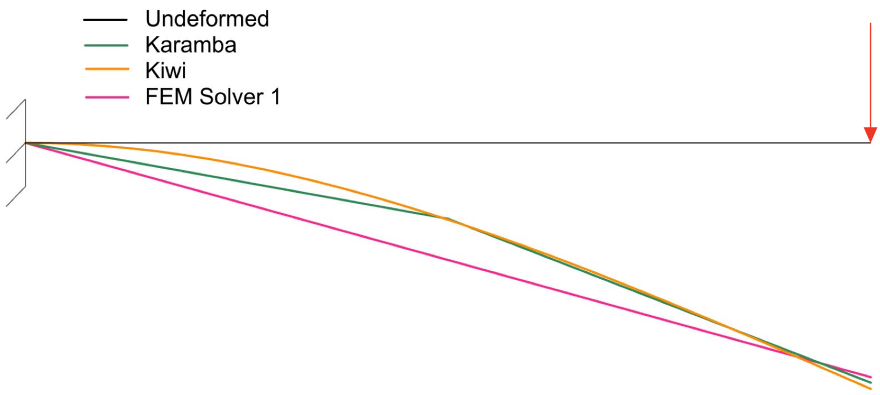


Figure 5.6: Deflection pattern for cantilever from Karamba, Kiwi and FEM solver 1. Deflection in scale 1:500.

The displacement pattern from Karamba, Kiwi and FEM solver 1 is presented in Figure 5.6. For Karamba and FEM solver 1, displacement patterns are linearly interpolated. FEM solver 1 interpolates between the two nodal values at start and end node, whereas Karamba takes an extra field value to display the displacement field. Kiwi displays the displacement pattern by a NURBS of a polynomial degree that is determined in the curve refinement in the analysis. Similar displacement patterns validated the shape functions for FEM solver 1.

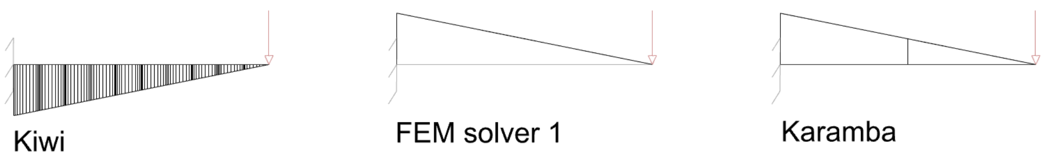


Figure 5.7: Moment distribution (M_y) from Kiwi, FEM solver 1 and Karamba.

Table 5.2: Maximum moment $(M_{y.max})$ for cantilever.

Analyse Type	**Maximum moment $(M_{y.max})$ [kNm]**
Exact Solution (M = PL)	-1
Kiwi	-0.999
FEM solver 1	-1
Karamba	-1

Figure 5.7 presents moment diagrams (M_y) from Kiwi, FEM solver 1 and Karamba. The maximum values ($M_{y.max}$) are presented in Table 5.2. For Kiwi, positive rotations are opposite from Karamba and FEM solver 1. Kiwi presents results from the Gauss points. Gauss points will not have coordinates at start and end point of the curve, and the results are therefore differing from Karamba. With a curve refinement of polynomial degree 10 and ten knot spans $\neq 0$, the values are converging as a Gauss point is approaching the node of interest. Similar moment diagrams validated the stress and strain in the structure.

Table 5.3: Support forces and moments from FEM solver 1 and Karamba.

Analyse Type	**Support Force (x,y,z) [kN]**	**Support Moment (x,y,z) [kNm]**
Exact Solution	(0,0,-P = 1)	(0,0, PL = -1)
FEM solver 1	(0, 0, 1)	(0, -1, 0)
Karamba	(0, 0, 1)	(0, -1, 0)

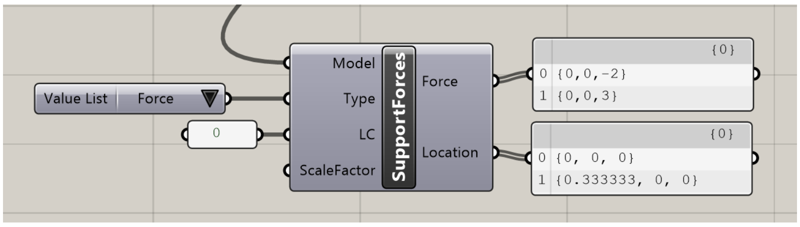


Figure 5.8: Support forces from Kiwi.

The support forces and moments from FEM solver 1 and Karamba had equal values, presented in Table 5.3. Similar results validated the algorithm, more specifically that the element stiffness matrix and total stiffness matrix are correct. Support reactions from Kiwi was difficult to interpret. Presented values were located in the Gauss points with indecisive values, presented in Figure 5.8. Therefore, the support reactions from Kiwi was insignificant for comparison and will not be presented for the other benchmark or study cases.

This particular comparison confirmed that the FEM solver 1 is working in the xz-plane. In addition, different locations in space of the cantilever was controlled and validated, but will not be presented.

5.3 Arc

Benchmark number two is an arc. The purpose of this benchmark was to control whether the FEM solver 1 works for curved structures, has correct transformations, and lastly, is continuous and has correct node division.

5.3.1 Geometry and structural model

The geometry of the arc was constructed with the NURBS component, as described in Section 4.2.4.

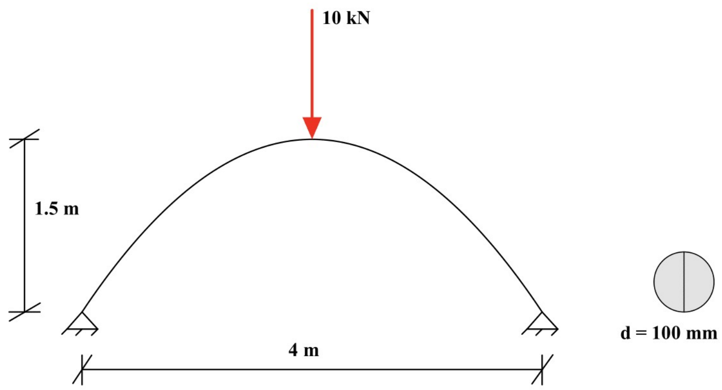


Figure 5.9: Structural model and cross section for arc.

Figure 5.9 presents the structural model and the cross section of the arc. The geometry is pinned in both ends, and loaded with 10 kN in the midpoint. Material for the arc is steel S355, and the cross section is circular with a diameter of 100 mm.

For Karamba and FEM solver 1, the analysis required a discretization of the arc into a mesh. Kiwi took the exact NURBS geometry, and refined the curve inside the analysis.

5.3.2 Results

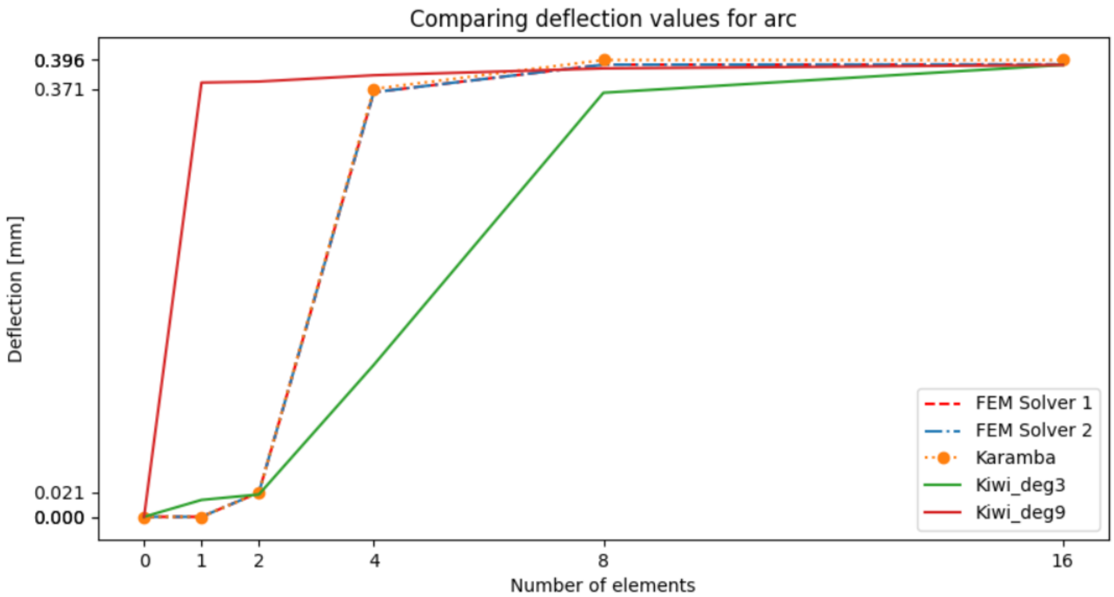


Figure 5.10: Deflection convergence for the arc. Curve refinement in Kiwi with polynomial degree 3 and 9.

Figure 5.10 presents the deflection for the arc from FEM solver 1, FEM solver 2, Karamba and Kiwi with a curve refinement of polynomial degree 3 and 9. For the FEM solvers, the geometry was discretized by dividing the arc into an increased amount of straight lines. Curve refinement with increased number of knot spans $\neq 0$ and increased polynomial degree generated a finer curve refinement for Kiwi. Examples of knot vectors for degree 3 polynomials are presented below.

For one element:

$$\Xi = [0 \quad 0 \quad 0 \quad 1 \quad 1 \quad 1]$$

For two elements:

$$\Xi = [0 \quad 0 \quad 0 \quad 0.5 \quad 1 \quad 1 \quad 1]$$

For three elements:

$$\Xi = [0 \quad 0 \quad 0 \quad 0.33333 \quad 0.66667 \quad 1 \quad 1 \quad 1]$$

Knot vectors for a finer curve refinement is described in section 2.4.1.

Figure 5.10 shows how the deflection converges when refining the basis functions in Kiwi with polynomials of degree 3 and 9. Kiwi converged with only one element with polynomials of degree 9 and after 16 elements with polynomials of degree 3.

Table 5.4: Program execution time for the arc with curve refinement of degree 3, 6 and 9 and one non-zero knot span from Kiwi.

Degree of curve refinement	**Execution time [s]**
3	0.514
6	0.517
9	0.536

The convergence was faster for a higher polynomial degree of the curve refinement, but as Table 5.4 presents, the execution time was increasing with a higher polynomial degree calculation, even for one element.

Table 5.5: Deflection values at the midpoint of the arc.

Analyse type	**Deflection [mm]**
FEM solver 1	0.392
FEM solver 2	0.392
Karamba	0.396
Kiwi	0.386

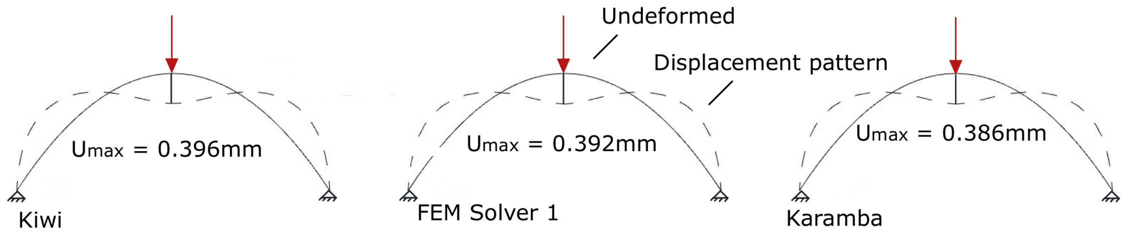


Figure 5.11: Displacement pattern for the arc from Kiwi, Fem solver 1 and Karamba.

The maximum displacement was compared in the mid-span of the arc, below the load. Necessary discretization for convergence, and convergence value is presented in Figure 5.10 and Table 5.5, respectively. FEM solver 1 and FEM solver 2 gave the same results. This similarity was common for all results controlled, therefore only results from FEM solver 1 will be presented in the following study cases. FEM solver 1 is chosen based on the execution time presented in Section 5.1. The deflection patterns from Kiwi, FEM solver 1, Karamba are presented in Figure 5.11.

Table 5.6: Maximum moment values ($M_{y.max}$) at midpoint of the arc.

Analyse type	**Maximum moment ($M_{y.max}$) [kNm]**
Kiwi	2.339
FEM solver 1	2.366
Karamba	2.370

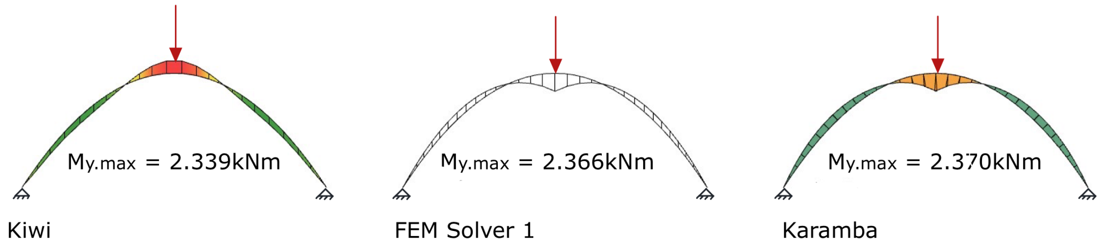


Figure 5.12: Moment diagram for the arc from Kiwi, FEM solver 1 and Karamba.

Figure 5.12 presents moment diagrams from FEM solver 1, Karamba and Kiwi. Maximum values ($M_{y.max}$) are presented in Table 5.6. Kiwi had opposite sign convertion, as commented in Section 5.2.

Table 5.7: Reaction values for the support points of the arc.

Analyse type	Loaction	Reaction Forces [kN]	Reaction Moments[kNm]
FEM solver 1	Cpt 1	(5.089, 0, 0.5)	(0, 0, 0)
Karamba	Cpt 1	(5.090, 0, 0.5)	(0, 0, 0)
FEM solver 1	Cpt 2	(-5.089, 0, 0.5)	(0, 0, 0)
Karamba	Cpt 2	(-5.090, 0, 0.5)	(0, 0, 0)

Reaction values from Karamba and FEM solver 1 are presented in Table 5.7.

5.4 Discussion

In a parametric framework, a low execution time is desirable. From Section 5.1 it is shown that using *Dense* matrices and *Cholesky* for solving linear equations, reduce the execution time considerably. By implementing these properties in the script, FEM solver 1 was competitive with analysis in Karamba when analyzing less than 40 elements. The script structure of Karamba has been developed and modified throughout the years. FEM solver 1 was created in four months. Therefore, the behavior of the solver is rather impressive.

FEM solver 1 gives output of great value; in other words, all controls are adequate compared with Karamba. Correct nodal values and support forces indicate that both the element stiffness and transformation matrix were established correctly and that the total stiffness matrix was assembled with proper connectivity. Further, the moment diagrams validate the stress and strain within the element field, concluding that the shape functions were accurately developed and implemented. Accordingly, all characteristics of the FEM were controlled, and there is reason to say that any essential aspect of the algorithm is working.

There was no mathematical advantage of using FEM solver 2 compared to FEM solver 1. The nodal values in the mid node in FEM solver 2 can be identified by finding field values or having a finer discretization of FEM solver 1. This was proven in Section 5.3 where the solvers returned the same results.

This was because a cubic displacement pattern can be described with two nodes, which is how the beam deflects for a point load. Even though FEM solver 2 has a displacement pattern of the fifth degree, it did not make the results more accurate. Nevertheless, this depends on the loading situation, but results equivalent with FEM solver 1 can always be achieved with a finer mesh. For the three-noded element to have a useful function, it needed an angular distortion in the mid node - describing the curve. This element had to be mapped to be used in FEA, as this approach to structural analysis is based on straight elements. Regardless, mapping the FEM elements with more internal nodes was not the solution. To achieve a mathematical advantage of elements with several nodes and different field descriptions, IGA was the answer. IGA uses the shape functions from the exact geometry description to describe the assumed field values.

Results from Kiwi was in accordance with Karamba, indicating that the software was applied correctly. The exact geometry of the benchmark was the input for the analysis, and discretization was performed with curve refinement. Deflection convergence was dependent on the polynomial degree of the curve refinement for IGA, as presented in Section 5.3. Even though the curve was described as a NURBS with polynomial degree 3, a higher polynomial degree described the displacement pattern. A higher polynomial degree in the curve refinement included more control points in describing the field values. Such an element converged faster, as the number of nodes increased without having more elements. In other words a higher-order polynomial degree in the curve refinement resulted in higher-order continuity in the interpolation field and increased smoothness of the solution space. Also, several non-zero knot spans improved accuracy, as this equals a finer mesh discretization in FEA. In addition, results from Kiwi were presented in the Gauss points. With this in mind, one should always ensure to create a curve refinement with coordinates of Gauss points approaching points of interest. See Table 2.1 for the location of the Gauss points. With this knowledge, correct usage of Kiwi should be achievable.

6 Study Case

This master's thesis includes two study cases to examine and evaluate the value of two approaches to structural analysis, FEA and IGA; a simple gridshell and the gridshell roof over the Great Court in the British Museum. A comparison of FEM solver 1 and Kiwi was performed through the study cases. In addition, results from Karamba was included to maintain validation of the analysis.

6.1 Gridshell

The first study case is a gridshell. Gridshells are often the load bearing system in complex spatial structures that toady's architecture is approaching. Therefore, it is appropriate to examine a gridshell when looking into the value of different approaches to structural analysis.

6.1.1 Geometry

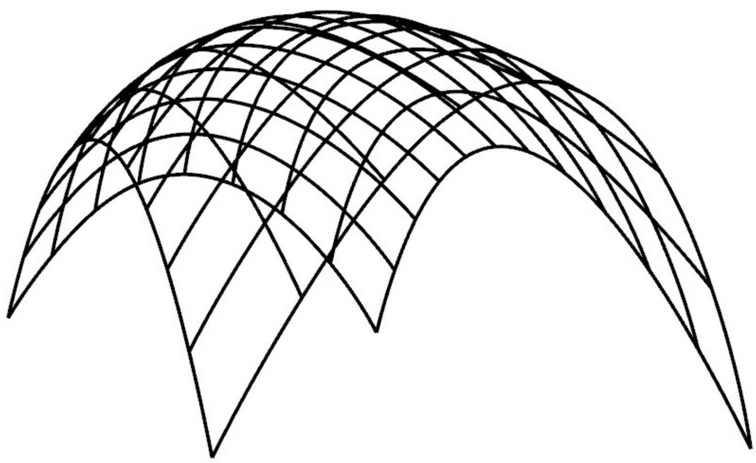


Figure 6.1: Geometry of the gridshell.

The parametric gridshell was created with components from Grasshopper, including the NURBS component. First, three NURBS with three control points each were constructed. This resulted in three quadratic NURBS. Sliders controlled the control points to modify the curve quickly. Then, a surface was made by lofting the three NURBS. This surface was the base for the gridshell. Each orthogonal side was divided into n number of divisions from the surface. A curve was then interpolated through these points. Hence, the dimension and number of divisions in both directions were parametric.

6.1.2 Structural model

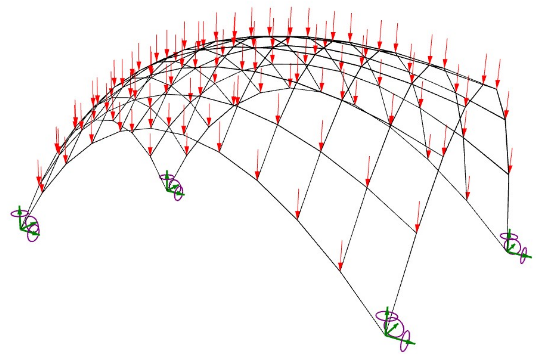


Figure 6.2: Model of the gridshell for the FEM solvers.

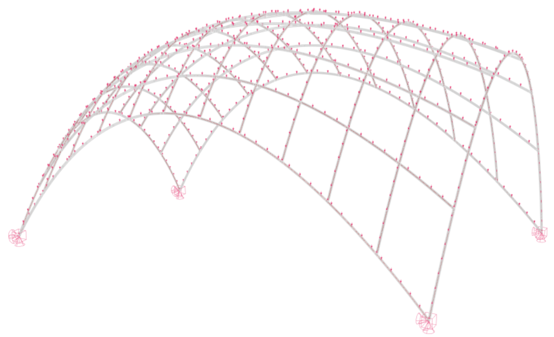


Figure 6.3: Model of the gridshell for the IGA solver.

To implement the geometry in FEM solver 1, straight lines were created between the intersection points, see Figure 6.2. This created a diszretization of eight and eleven element divisions for the two directions, respectively. The structural analysis in Kiwi was performed directly on the exact geometry from Grasshopper, shown in Figure 6.3.

The load case for the structural model was based on the opportunities developed for FEM solver 1. FEM solver 1 is restricted to having nodal point loads. Accordingly, the chosen load pattern was to have 1 kN in each node. Discretization for IGA differs from the FEM, and the DOFs are located in the control points. Control points can have coordinates outside the curve and would not perform desirable as load points. Therefore, a distributed curve load was used for Kiwi, with a load value to imitate the load pattern for FEM solver 1.

There are four fixed support points for the structure, see Figure 6.2 and 6.3. The structural model was built with steel S355, and have same circular cross section as the arc, see Figure 5.9.

6.1.3 Results

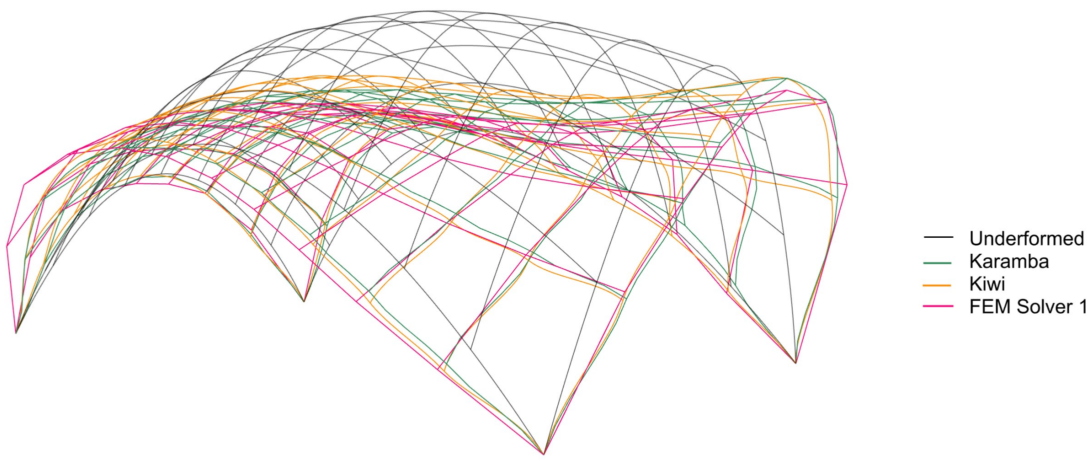


Figure 6.4: Deflection pattern for the gridshell from Karamba, Kiwi, FEM solver 1. Deflection in scale 1:100.

Table 6.1: Maximum deflection of the gridshell.

Analyse type	Deflection [mm]
Karamba	40.910
Kiwi	40.648
FEM solver 1	40.848

Figure 6.4 presents the deflection pattern for the gridshell from Karamba, Kiwi, and FEM solver 1. The shape of the deflection is similar in all cases. In Table 6.1, the minor variations in the maximum displacement between the analysis are presented. It is observed that the IGA and the FEM converged to the same solution.

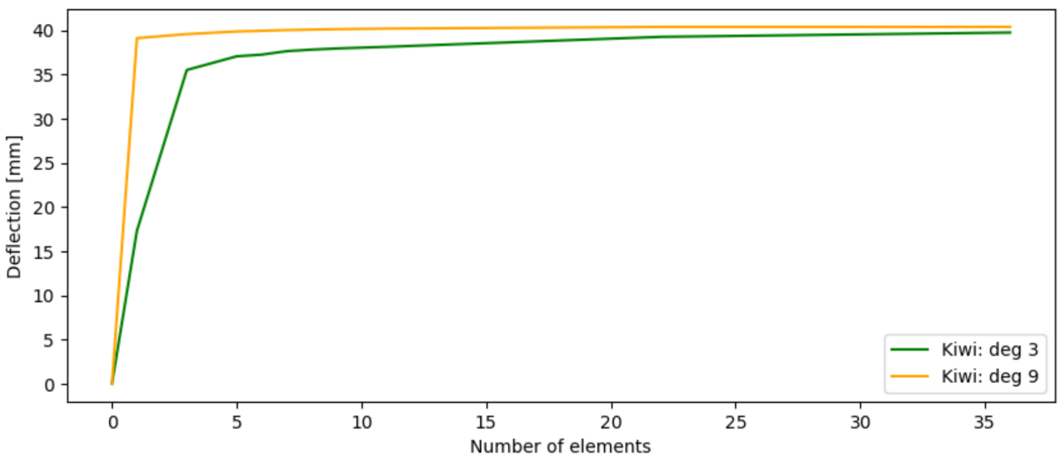
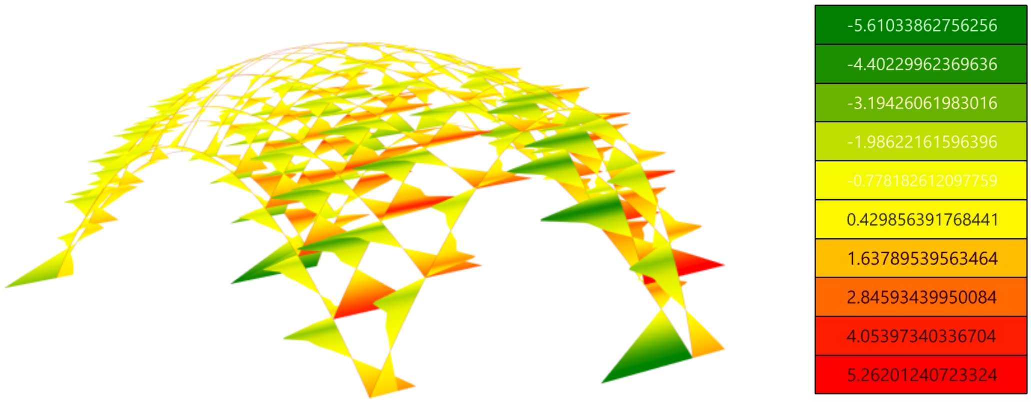


Figure 6.5: Convergence rate for deflection of the gridshell from Kiwi with curve refinement of degrees 3 and 9.

The convergence rate from Kiwi with curve refinement of polynomial degrees 3 and 9 is presented in Figure 6.5. For a curve refinement of degree 3, convergence for the deflection was not achieved for 36 elements. The geometry is curved, and FEM solver 1 (and Karamba) should have a finer discretization. Anyhow, this was the theme for the arc benchmark in Section 5.3. Therefore, it was not prioritized here, as the results from FEM solver 1 were quite similar to what Kiwi was approaching.



(b) Legend [kNm]

(a) Moment diagram

Figure 6.6: Moment diagram M_y for the gridshell from Kiwi.

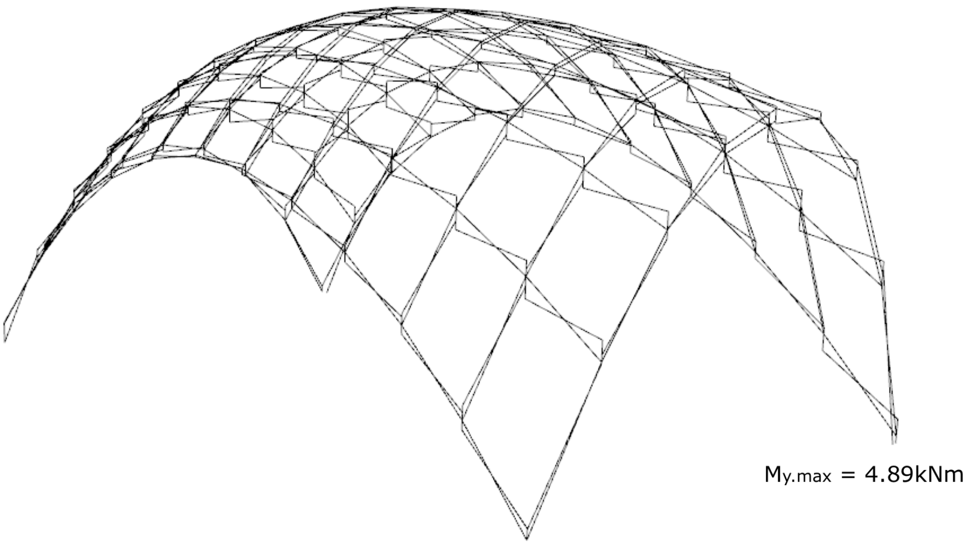


Figure 6.7: Momentdiagram M_y for the gridshell from FEM solver 1.

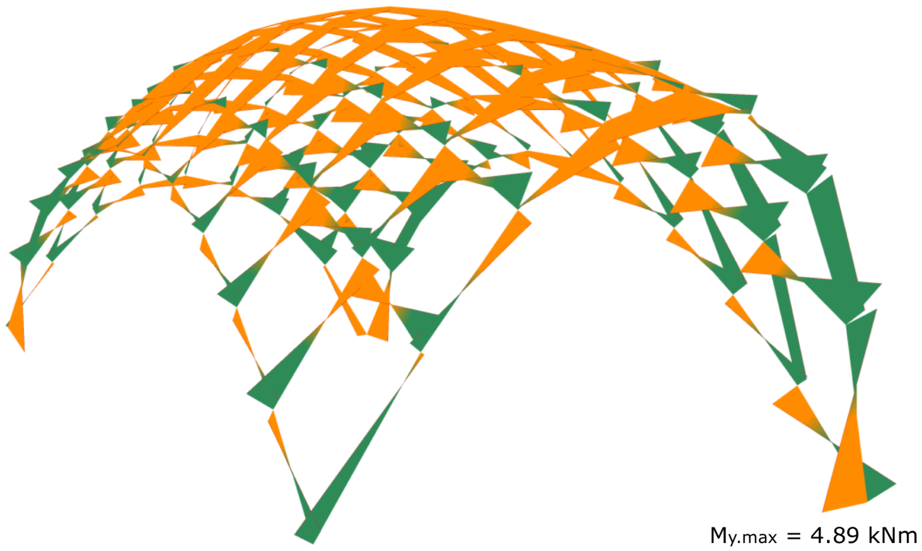


Figure 6.8: Moment diagram M_y for the gridshell from Karamba.

Table 6.2: Maximum moment $M_{y,max}$ of the gridshell.

Analyse type	**Maximum moment $M_{y,max}$ [kNm]**
Kiwi	see legend on Figure 6.6b
FEM solver 1	4.89
Karamba	4.89

The moment diagrams (M_y) from Kiwi, FEM solver 1 and Karamba are shown in Figure 6.6, Figure 6.7 and Figure 6.8, respectively. Table 6.2 presents the maximum moment ($M_{y,max}$) for the gridshell from Kiwi, FEM solver 1, and Karamba. Kiwi is calculating the moments (M_y) from each NURBS down to the support. This results in two values for the support position, which must be subtracted to get the resulting value. Accordingly, the maximum moment ($M_{y,max}$) is presented by the legend in Figure 6.6b.

Table 6.3: Reaction values in the gridshell.

Analyse type	**Support Point**	**Reaction Force[kN]**	**Reaction Moment [kNm]**
FEM Solver 1	1	(14.081, 10.201, 26.189)	(1.382, -5.380, 0.160)
Karamba	1	(14.086, 10.204, 26.192)	(-1.387, 5.384, -0.161)
FEM Solver 1	2	(13.211, -10.461, 30.289)	(-5.0491, -9.241, 0.263)
Karamba	2	(13.211, -10.462, 30.286)	(5.050, 9.242, -0.263)
FEM Solver 1	3	(-16.668, 8.436, 24.422)	(-0.368, -0.499, 1.717)
Karamba	3	(-16.669, 8.434, 24.420)	(0.369, 0.499, -1.718)
FEM Solver 1	4	(-10.624, -8.175, 23.099)	(-9.127, -1.594, -3.072)
Karamba	4	(-10.628, -8.177, 23102)	(9.131, 1.592, 3.075)

Table 6.3 presents the reaction values from FEM solver 1 and Karamba in the four support points.

FEM solver 1 and Kiwi present equivalent results compared to Karamba for all controls within an acceptable tolerance.

Table 6.4: Execution time for the gridshell from FEM solver 1, Kiwi and Karamba.

Analyse type	**Execution time [ms]**
FEM solver 1	89
Kiwi	183
Karamba	13

Table 6.4 represents the execution time for the analysis from Kiwi, FEM solver 1 and Karamba. The curve refinement for Kiwi is polynomial degree 3 and eleven/eight non-zero knot span, which should describe the same element division as for the FEM solvers, see description of the structural model.

6.2 The gridshell roof over the Great Court at the British Museum

The final study case is the complex roof structure over the Great Court at the British Museum. This structure was included as one of the study cases to compare advanced geometry with different approaches to structural analysis. The cross sections of the structure were optimized with Galapagos in the evaluation of FEM and IGA.

Figure 6.9: The gridshell roof over the Great Court in the British Museum (D'Amico, 2015).

Figure 6.9 is presenting the roof over the Great Court in the British Museum.

6.2.1 Geometry

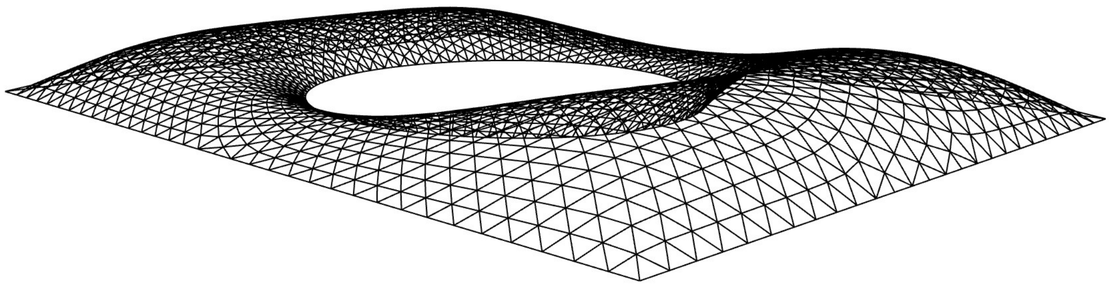


Figure 6.10: The gridshell roof over the Great Court at the British Museum drawn in Rhino.

The roof over the Great Court in the British Museum is a spectacular gridshell. The geometry is constructed as a lattice of linear elements creating a long-span curved structure (Malek et al., 2014), se geometry in Figure 6.10. The mesh of the structure is implemented in Rhino from an AutoCad file. In Grasshopper, the mesh is exploded into segments and points to control lines and nodes as inputs for boundary points, loading points, and creating structural elements.

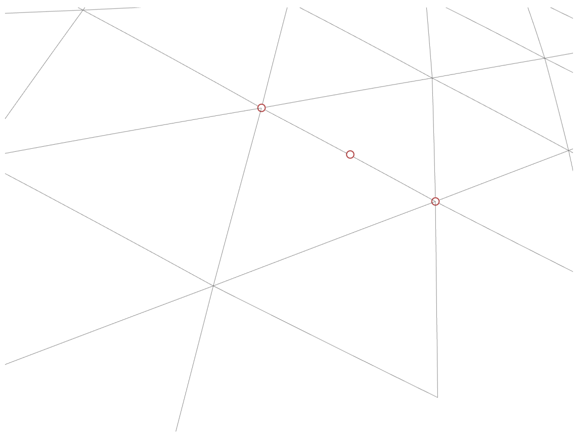


Figure 6.11: Control points for one patch of the gridshell roof geometry.

The geometry of the gridshell roof was redesigned from straight lines into NURBS as input for Kiwi. Each member in the grid is defined as one patch, see Figure 6.11. Curve refinement is expressed by degree 3 and one non-zero knot span to make the analysis as fast as possible. This creates patches with three control points as shown in Figure 6.11.

6.2.2 Structural Model

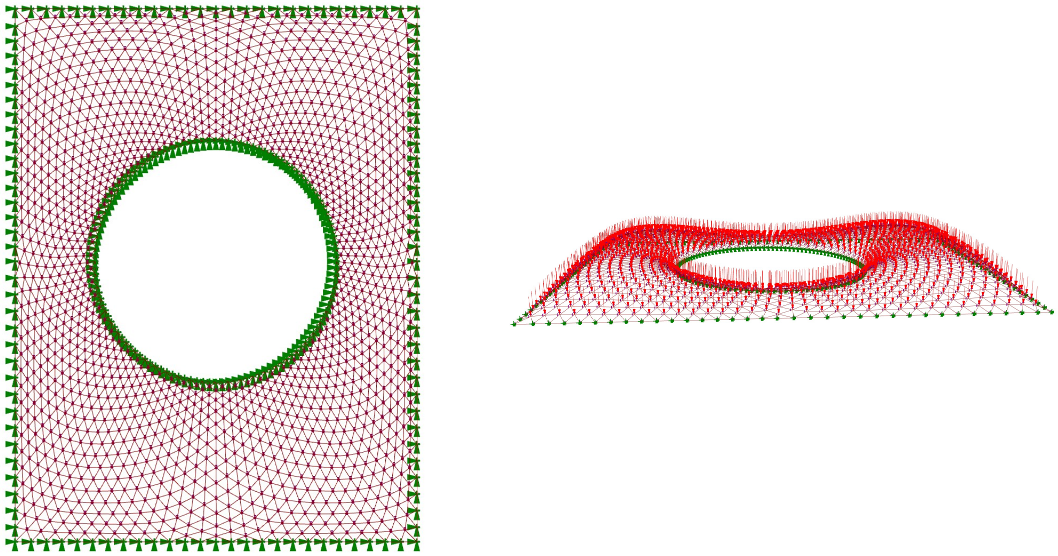


Figure 6.12: Structural model of the gridshell roof.

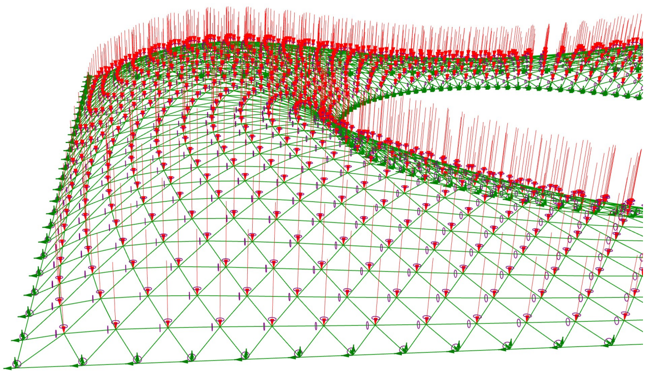


Figure 6.13: Detail of boundary points.

The structure was loaded with 1 kN at every node, except the boundary points. Point loads were also applied for IGA. This was possible because each member in the grid was described as one patch, causing a control point in each intersecting point. Boundary points were all the points along the boundary of the geometry, see Figure 6.12 and Figure 6.13. Boundary points were restricted in every translation direction and torsion, and the loading points were restricted against rotation. The material used in the analysis was S355 and the cross section was the same as for the arc, see figure 5.9.

6.2.3 Results

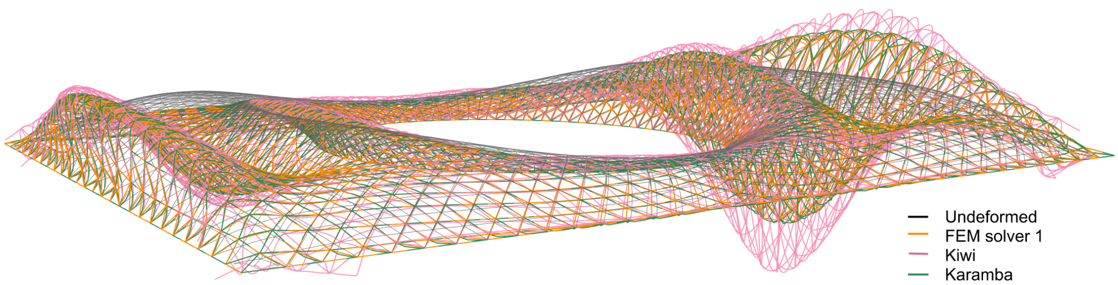


Figure 6.14: Deflection pattern for the gridshell roof. The deflection is in scale 1:5000.

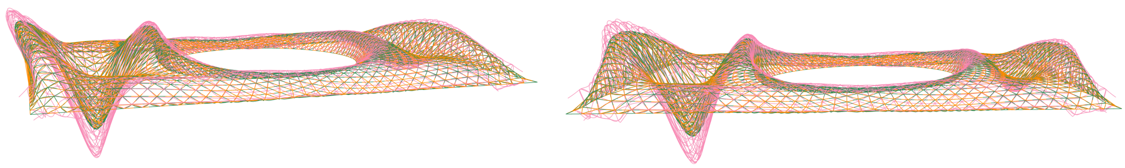


Figure 6.15: Deflection pattern from FEM solver 1, Kiwi and Karamba. The deflection is in scale 1:5000.

Table 6.5: Maximum deflection of the gridshell roof.

Analyse type	**Deflection [mm]**
FEM solver 1	3.817
Kiwi	5.164
Karamba	3.821

Figure 6.14 and Figure 6.15 present the deflection pattern of the gridshell roof from FEM solver 1, Kiwi and Karamba. Maximum displacement is shown in Table 6.5. FEM solver 1 has adequate results compared with Karamba. Both the deflection pattern and maximum displacement value from Kiwi differs from Karamba. The displacement pattern from Kiwi shows an error along the boundary of the structure. A couple of support points are not assembled in the model, as the NURBS do not fully intersect those specific points. This causes a structure of decreased stiffness, which can explain the higher value of the maximum displacement.

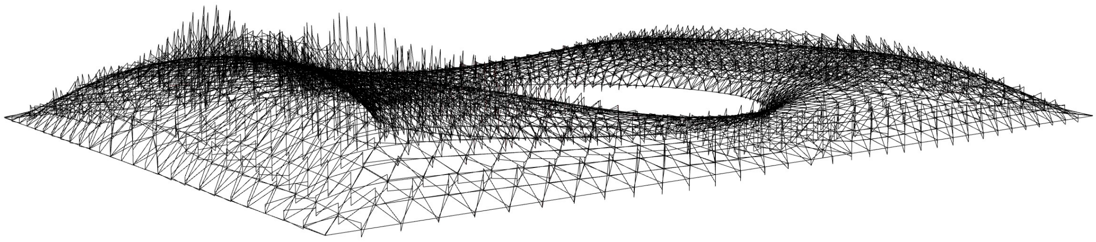


Figure 6.16: Moment distribution (M_y) in the gridshell roof from FEM solver 1.

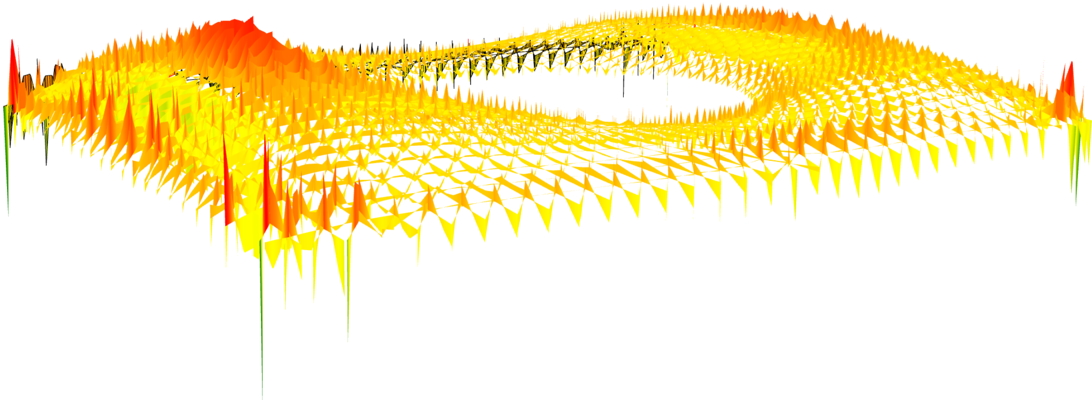


Figure 6.17: Moment distribution (M_y) in the gridshell roof from Kiwi.

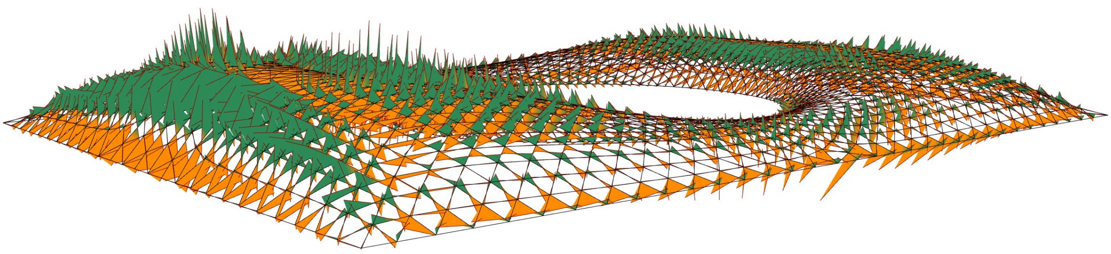


Figure 6.18: Moment distribution (M_y) in the gridshell roof from Karamba.

Table 6.6: Maximum moment ($M_{y.max}$) for the gridshell roof.

Analyse type	**Maximum moment ($M_{y.max}$) [kNm]**
FEM solver 1	0.573
Karamba	0.556

The moment distribution (M_y) from FEM solver 1, Kiwi and Karamba has peaks at approximately the same locations, shown in Figure 6.16, Figure 6.17 and Figure 6.18. Kiwi has larger moment values along the boundary, for the same reasons as for the larger displacement. The maximum moment values ($M_{y.max}$) for FEM solver 1 and Karamba are displayed in Table 6.6. Maximum moment value ($M_{y.max}$) from Kiwi is not presented, due to unfavorable presentations, like described in Section 6.1.

Table 6.7: Comparison of vector length of the reaction forces from FEM solver 1 and Karamba.

Analyse type	**Maximum Reaction Forces (vector length)[kN]**
FEM solver 1	22.6
Karamba	22.6

The support forces were difficult to compare, as there are 240 defined boundary points. To show the equivalent support forces, the maximum length of support force vector (x,y,z) was compared, see Table 6.7.

Table 6.8: Execution time for the gridshell roof from FEM solver 1, Kiwi and Karamba.

Analyse Type	**Execution Time**
FEM solver 1	5.6 min
Kiwi	5.6 min
Karamba	273 ms

Table 6.8 presents the execution time for the analysis of FEM solver 1, Kiwi and Karamba. The large execution time distinguish Karamba from FEM solver 1 and Kiwi. The gridshell roof is a complex structures that is perceptible in the execution time of the FEM solver 1 and Kiwi.

6.2.4 Optimization of the cross section

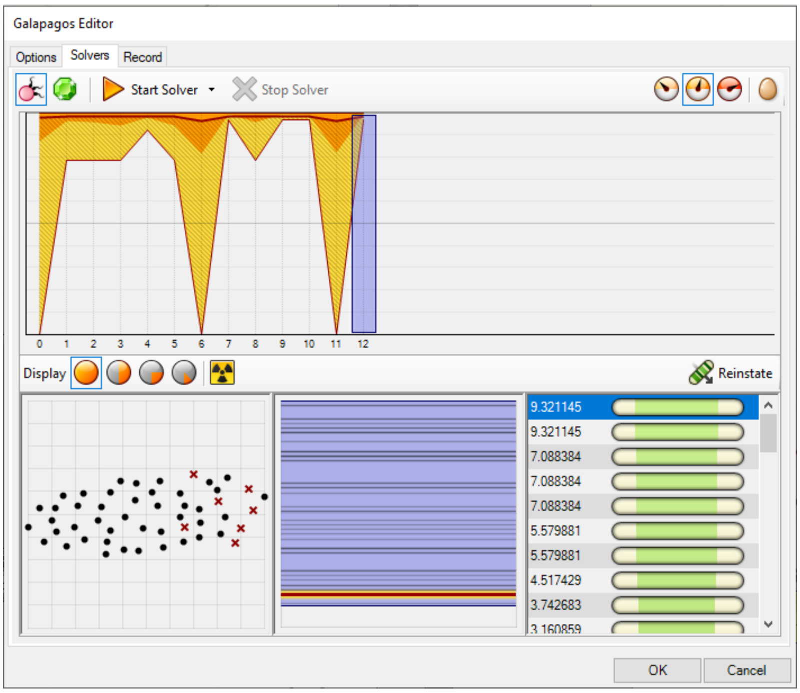


Figure 6.19: Optimization of the cross section for the gridshell roof with Galapagos.

Galapagos was used for optimization of the cross section with Karamba, as shown in Figure 6.19. Karamba was used instead of FEM solver 1 and Kiwi, due to the long execution time. The load case was including self-weight, service load, and load from the overlaying glass roof. The self-weight was changing within the iterative interval [1mm,100mm] of the radius of the cross section. The estimated load was set to 10 kN for each node, excluding self-weight. The allowable deflection was set to L/360 (= 95 mm), on the basis of the acceptable deformation for this steel roof.

Table 6.9: Optimized cross section based on minimum deflection for the gridshell roof.

Analyse type	Optimized diameter of cross section [mm]	Deflection value [mm]
Karamba	80	93.2

Table 6.9 presents the optimized cross section for the gridshell roof with Karamaba.

6.3 Discussion

The study case of the gridshell substantiated both FEA and IGA as an approach to structural analysis. The results for the gridshell from FEM solver 1 and Kiwi were adequate after validation by Karamba. In addition, both analyses ran within an acceptable execution time.

The geometry of the gridshell was constructed with NURBS. The meshing for the FEM was complicated because a finer discretization required a redesign of the geometry. Discretization of the arc was easily modified with curve division, which did not work for the gridshell.

The gridshell was constructed of NURBS that intersected and created a grid. A finer curve division of each NURBS was redesigning the grid and not creating a finer mesh. Therefore, the input geometry for the FEA was time consuming to refine. Discretization was unfortunate but this is also the well-known disadvantage of this approach to structural analysis. On the contrary, discretization for IGA was more effortless. IGA allowed for a curve refinement within the analysis, where the variables were the degree of polynomials and the number of knot spans $\neq 0$. However, it was essential to be aware of the location of the control points to ensure the correct load distribution. Additionally, the number of Gauss points in the analysis indicated where the output values would be presented.

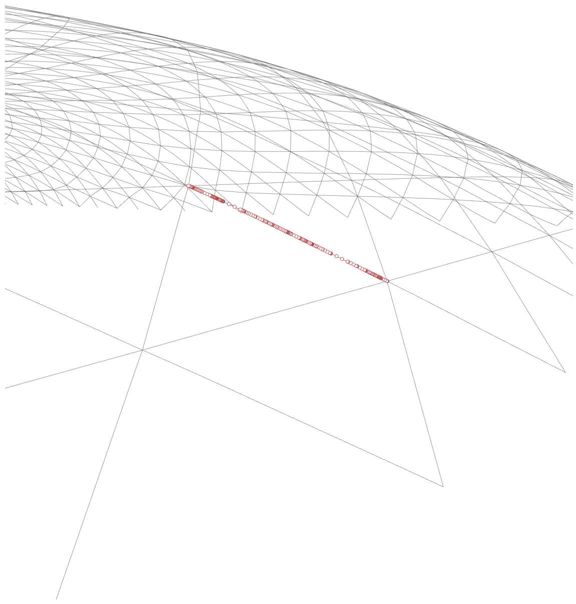


Figure 6.20: Control points for a NURBS patch after reconstructing the geometry from straight lines.

The geometry of the gridshell roof is based on straight members. Therefore, the geometry was all set to analyse with FEA. For Kiwi, the geometry had to be modified into NURBS. Within the analysis, the curve refinement were set to polynomial degree 3 and one knot span $\neq 0$ to reduce the execution time. However, when constructing the NURBS geometry each grid member was a patch of high curve refinement, see Figure 6.20. This caused a large number of control points, resulting in too many DOFs for the analysis. The NURBS had to be refined before taken as input for the analysis, like shown in Figure 6.11, for the analysis to be within a reasonable execution time.

The gridshell roof is a complex structure but, at the same time, not the most advanced structure. Gridshell structures are geometries that IGA is created for and should therefore give a more accurate result with a higher convergence rate than FEA. FEM solver 1, developed in four months, could analyse the geometry within the same execution time as Kiwi.

Therefore, Kiwi is not referable compered to Karamba, yet. IGA has great potential based on the theory, but there is no developed structural analysis algorithm to fulfill the expectations.

When it comes to execution time, both Kiwi and FEM solver 1 were slow for complex geometry. The execution time is paramount when working with conceptual design in a parametric environment. Waiting several seconds, or even minutes, for each design modification impedes creativity and collaboration. For Kiwi, the computational time was increased by increasing the number of knot spans (elements) and polynomial degree of the curve refinement, as shown in Section 5.3. The precision of results from IGA is highest when having a high degree, as this gave the highest convergence rate, see Section 6.1 and Section 5.3. This aspect was similar with a finer mesh for FEM solver 1. However, there are better options for FEA, like Karamba. For IGA, it was unfortunate that the accuracy should be at the expense of the execution time.

Galapagos was implemented in the analysis for the gridshell roof to optimize the cross sections. Optimization is a fundamental part of the design process in a parametric environment. Karamba returned accurate results within an acceptable execution time. The high execution time for Kiwi and FEM solver 1 made the solvers incompatible with Galapagos. This emphasis that Karamba is the only adequate structural analysis software for Grasshopper today, capable of completing a sufficient design process.

7 Summary

The FEM solvers were built as plug-ins for Grasshopper, in Rhino. The advantages of a parametric framework are convincing to increase the cooperation between architects and engineers. Rhino is a software well suited for a design process where geometry is adjusted and reanalyzed. This aspect was of great value when creating the algorithm for the FEM plug-ins. Geometry modifications enabled an examination of the performance of the stiffness and transformation matrix for arbitrary locations in space. When interactively changing the model and evaluating situations where errors appeared, the mistake in the algorithm could more easily be identified.

A significant difference between the software of FEA and IGA is how parameters are interpreted. The former is built on more intuitive and established concepts for engineers. In addition, the analysis indicates when the inputs are wrong, so the essence of achieving accurate results is to establish a good mesh. Kiwi converges towards accurate results based on different principles. The analysis establishes a finer curve refinement based on a higher polynomial degree and several non-zero knot spans, validated in Section 5.3. The knowledge of curve refinement is essential, as this determines the localization of DOFs and force distribution, and localisation of the Gauss points. The analysis is sensitive to input geometry without indicating error messages. The consequences can be fatal if the user is unaware of the required information concerning the theory of this structural analysis and how the software is created. On the other hand, IGA has excellent potential. When the desired design is established, only parameters for the analysis require modification to achieve a finer discretization of the geometry.

There are several differences between FEM solver 1 and Kiwi. FEM solver 1 is developed in four months. These four months also included learning the theory of the FEM and programming in C#. The algorithm gave excellent results for the study cases examined. Beyond this, there are uncertainties about the algorithm's performance, but it is reasonable to assume adequate results for corresponding structures. Kiwi is developed for a more extended period, but the complexity is not higher than for FEM solver 1. Accordingly, the theory of IGA is assumed to be more advanced and challenging to convert into a numerical and implementable language. In order to have a fair comparison of FEA and IGA, the numerical implementation of the latter should be modified and adjusted for higher complexity.

7.1 Further Work

The opportunities attached to the use of IGA in a parametric environment deserve to be investigated more. The theory has a lot of potential but needs more time to be developed further. Kiwi needs a faster execution time to compete with FEA in a parametric environment. Output values are difficult to interpret, as they are located in Gauss points that change location and amount with a finer curve refinement. At last, the program does not show error messages for wrong input; it just gives an error in the result, which is unfortunate. A proposition is to further develop the already established NURBS component to a complete IGA.

8 Conclusion

It was possible to develop a well-functioning FEA plug-in within four months. These four months included learning how to program in C#, gain knowledge about the FEM, and convert FEA into a numerical language for an algorithm in a parametric environment. Kiwi is developed within a longer period of time, but the code was neither faster nor returned more accurate results for the study cases examined. It is therefor reasonable to conclude that the theory of IGA is more complex to implement into a numerical structural analysis solver.

Today, FEA is fulfilling the duty as structural analysis. By discretizating geometry into a finer mesh, nearly all designs can be analyzed within an acceptable error. However, the design process is approaching complex spatial designs. The optimization process, especially refining geometry, is time consuming. Therefore, IGA has a significant theoretical value in structural analysis. However, this requires more research to convert IGA into a numerical and implementable language for an algorithm. The advantageous performance of the FEM is creating high expectations for a future IGA.

A considerable amount of development is required for an IGA to be a reliable tool for structural analysis in Grasshopper compared to a FEA, which has been tried and tested for decades. However, it is exciting to have a challenging method for the FEM. We are intrigued to observe whether the performance and influence of IGA will increase and equate to or substitute the FEM.

Bibliography

Bell, K. (2011). *Matrisestatikk*. Fagbokforlaget.

Bell, K. (2013). *An engineering approach to finite element analysis of linear structural mechanics problems*. Fagbokforlaget.

Caetano, I., Santos, L. & Leitão, A. (2020). Computational design in architecture: Defining parametric, generative, and algorithmic design. *9*, 287–300. https://www.sciencedirect.com/science/article/pii/S2095263520300029

Cook, R. D., Malkus, D. S., Plesha, M. E. & Witt, R. J. (2001). *Concepts and applications of finite element analysis*. John Wiley Sons Inc.

Cotterell, J., Hughes, T. J. R. & Bazilevs, Y. (2009). *Isogeometric analysis toward integration of cad and fea*. John Wiley; Sons.

D'Amico, B. (2015). Timber grid-shellstructures: Form-finding,analysis and optimisation. file:///C:/Users/ASUS/Downloads/PhD_Dissertation.pdf

Davidson, S. (2022). *Rhino 7 for windows and mac includes grasshopper*. Grasshopper3d. https://www.grasshopper3d.com/

Dhatt, G., Touzot, G. & Lefrançois, E. (2012). *Finite element method*. ISTE Ltd; John Wiley Sons, Inc.

Engle, E. (2020). What Is Parametric Design in Architecture, and How Is It Shaping the Industry? *Autodesk*.

Gan, B. S. (2018). *An isogeometric approach to beam structures*. Springer.

Gupta, K. & Meek, J. (1996). A brief history of the beginning of the finite element method. *INTERNATIONAL JOURNAL FOR NUMERICAL METHODS IN ENGINEERIN*.

Karamba3d. (2022). *Karamba, parametric engineerig*. Karamba3d. https://www.karamba3d.com/

Kiwi3d. (2022). *An introduction to iga*. Kiwi3d. https://www.kiwi3d.com/theory/

Malek, S., Wierzbicki, T. & Ochsendorf, J. (2014). *Buckling of spherical cap gridshells: A numerical and analytical study revisiting the concept of the equivalent continuum*. Massachusetts Institute of Technology, Cambridge. file:///C:/Users/ASUS/Downloads/Buckling_of_spherical_cap_gridshells_A_numerical_a.pdf

Mathisen, K. (2012). *Finite element modelling of structural mechanics problems*. NTNU.

Math.NET. (2022). *Math.net numerics*. Retrieved 26th May 2022, from https://numerics.mathdotnet.com/

McNeel, R. (2022). *Rhinoceros*. Rhino3d. https://www.rhino3d.com/

Peters, B. & Peters, T. (2013). *Inside SmartGeometry: Expanding the Architectural Possibilities of Computational Design*. John Wiley & Son.

Studios, M. V. (2022). *It's how you make software*. Retrieved 26th May 2022, from https://visualstudio.microsoft.com/

Terzidis, K. (2004). Algorithmic design: A paradigm shift in architecture ? *Harvard University*.

Zhang, G., Alberdi, R. & Khandelwal, K. (2016). Analysis of three-dimensional curved beams using isogeometric approach. *Engineering Structures*.

Appendix

A GitHub

The total script for the plug-in can be found in GitHub, in the branch named *Master*.

NTNU

Kunnskap for ei betre verd