Digital Workflow in Conceptual Structural Design

Parametric Design of Connections for Timber
Gridshell Structures

Helle Stam Faugstad
Øyvind Sunnvoll Rognes

Civil and Environmental Engineering
Submission date: June 2018
Supervisor: Anders Rönnquist, KT

Norwegian University of Science and Technology
Department of Structural Engineering

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

ACCESSIBILITY
OPEN

MASTER THESIS 2018

SUBJECT AREA: Engineering Architecture	DATE: 07.06.18	NO. OF PAGES: 12 + 120 + 57

TITLE:

Digital Workflow in Conceptual Structural Design
Parametric Design of Connections of Timber Gridshell Structures

Digital arbeidsflyt i konseptuell konstruksjonsdesign
Parametrisk design av knutepunkter for gitterskall i tre

BY:

Øyvind Sunnvoll Rognes and Helle Stam Faugstad

SUMMARY:
The question for discussion is how digital workflow, based on a parametric model, can be used as a method of design to increase the efficiency of spatial structure design. The study focus on the development of connections in timber gridshells. With a flexible digital workflow, the user is able to automatically generate structurally valid gridshell connections according to the gridshell properties. Such a workflow is supposed to enable more efficient communication between architects and engineers during the design process, by enabling the optimization of the economic, structural and conceptual sides of a project.

The digital workflow was implemented in a case study to engineer a spatial cabin structure in Norway and tested for different grid patterns and cross-sections. The case study involved cross-disciplinary work between two teams of engineers and one architect, increasing the realism of the research project.

The research lead to a digital workflow based around a particular proposed gridshell connection design. A parametric model with scripted structural verification algorithms was developed, and certain components were verified with a Finite Element Analysis. The structural verification mimics the intuitive engineering approach of testing different configurations, from simple to complex, until a valid configuration is found. The gridshell connection design proposed in the thesis also shows how the timber failure modes are critical in traditional slotted plate connection types.

The study finds clear advantageous with a digital workflow, e.g., better communication between architects and engineers and better predictions regarding structural stability. An on-hand parametric visualization of the structure facilitates for better understanding and communication, easier troubleshooting and less redundant work. It also makes it easier to experiment with changes and communicate options across disciplines, due to a work culture built on common terms.

RESPONSIBLE TEACHER: Nils Erik Anders Rønnquist

SUPERVISORS: Nils Erik Anders Rønnquist, Marcin Luczkowski

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

Abstract

Timber gridshell structures are an efficient way of covering large spaces, while it also has an extraordinary ability to capture architectural interest and exploit material properties. From an engineering point of view, the success of a project can be achieved through innovative design of form, that takes advantage of the unique structural load carrying capability of a gridshell. As gridshell design is to consider as an emergent technology, there is still a lot of potential in this research field.

The question for discussion is how digital workflow, based on a parametric model, can be used as a method of design to increase the efficiency of spatial structure design. The study focuses on developing connections in timber gridshells. With a flexible digital workflow, the user can automatically generate structurally valid gridshell connections according to the gridshell properties. Such a workflow enables more efficient communication between architects and engineers during the design process, by allowing optimization of the economic, structural and conceptual sides of a project.

The digital workflow was implemented in a case study to engineer a spatial cabin structure in Norway, and tested for different grid patterns and cross-sections. The case study involved cross-disciplinary work between two teams of engineers and one architect, increasing the realism of the research project.

The research led to a digital workflow based on a particular proposed gridshell connection design consisting of aluminum gusset plates, slotted in the glulam beams and attached to a center thin-walled aluminum cylinder. A parametric model, with parallel structural verification algorithms, was developed, and specific components were verified with further structural analysis. The structural verification mimics the intuitive engineering approach of testing different configurations, from simple to complex, until a valid configuration is found. The gridshell connection design proposed in the thesis also shows how the timber failure modes are critical in traditional connections with slotted plates.

The digital workflow succeeded in generating and structurally validating custom gridshell connections. The results can be used when discussing the structural ability of different gridshell forms and beam sizes. The study finds clear advantages with a digital workflow, e.g., better communication between architects and engineers and better predictions regarding structural stability. An on-hand parametric visualization of the structure facilitates for better understanding, easier troubleshooting and less redundant work. It also makes it easier to experiment with changes and communicate options across disciplines, due to a work culture built on common terms.

Sammendrag

Gitterskall i tre er en effektiv konstruksjonstype for store spenn, samtidig som det har en spesiell evne til å vekke interesse blant arkitekter og utnytte fordelene ved tre som materiale. Fra et ingeniørperspektiv vil et innovativt design, som effektivt utnytter den spesielle bæreevnen, kunne skape stor konstruksjonsmessig glede. Ettersom gitterskall i tre fortsatt er å anse som en fremvoksende teknologi, finnes det stort potensiale i dette som et forskningsfelt.

Forskningsspørsmålet som skal undersøkes er hvordan en digital arbeidsflyt, basert på en parametrisk modell, kan bli brukt som en designmetode for å øke effektiviteten i design av romlige gitterskallkonstruksjoner. Denne oppgaven fokuserer på utviklingen av knutepunktene i gitterskall av tre. Med en fleksibel digital arbeidsflyt vil brukeren ha muligheten til å automatisk generere godkjente knutepunkter i henhold til de gitte gitterskallegenskapene. En slik arbeidsflyt muliggjør en mer effektiv kommunikasjon mellom arkitekter og ingeniører under designprosessen. Dette muliggjøres gjennom både økonomiske, strukturelle og konseptuelle sider av et prosjekt.

Den digitale arbeidsflyten ble implementert i en casestudie som går ut på å konstruere en hytte med en romlig gitterskall struktur. Den ble så testet for ulike typer gitterskallformer og tverrsnitt. Casestudien innebar også tverrfaglig arbeid mellom to grupper med ingeniørstudenter og en arkitekt, for å gjøre prosjektet mer virkelighetsnært.

Forskningen resulterte i en digital arbeidsflyt basert på et bestemt forslag til gitterskallknutepunkt. En parametrisk modell med programmerte verifikasjons-algoritmer for konstruksjonen ble utviklet, og visse komponenter ble ytterligere verifisert ved hjelp av elementmetodeanalyse. Denne verifikasjonsmetoden etterligner den intuitive ingeniørtilnærmingen med å teste forskjellige konfigurasjoner, fra enkel til kompleks, helt til en gyldig konfigurasjon er funnet. Knutepunktsdesignet for det gjeldende gitterskallet foreslått i denne oppgaven viser også hvordan brudd i trevirket er kritisk for tradisjonelt innslissede plater.

Studien finner klare fordeler ved bruk av digital arbeidsflyt, blant annet bedre kommunikasjon mellom arkitekter og ingeniører og bedre forutsigelser knyttet til konstruksjonens stabilitet. En parametrisk visualisering av konstruksjonen muliggjør bedre tverrfaglig forståelse, enklere feilsøking og mindre overflødig arbeid. Den vil også gjøre det enkelt å eksperimentere med endringer i designet, samt å kommunisere ulike alternativer i et prosjekt på tvers av fag, ettersom prosjektet er bygget på felles premisser.

Problem Description

The topic of interest is the use of digital workflow to create connections in spatial structures. With increased computer technology, and new production methods suited for custom fabrication, more projects can take advantage of automating the detailing process. By using the concept of parametric design, the architects and engineers can alter the properties of the connections, and experiment with form. The tool also enables the users to quickly mass produce 3D-models of all the connections in a given structure, to create digital documentation and visual facilities efficiently.

This thesis will focus on using digital workflow in the design of the connections in a timber gridshell. The task involves researching different connection designs and creating a digital workflow based on a parametric model that generates the 3D-models. The digital workflow must include calculations and structural verifications of the connections, and necessary controls for the assembly of the connections. The connections are optimized according to the design forces and the calculated capacity. The process will involve exploring different existing connection designs and propose a design concept for a case study.

The digital workflow shall be tested for realistic use in a case study, involving the design of a timber gridshell roof for the cabin. Another pair of master students will collaborate on the project, proposing a global form and grid pattern. The compatibility between the digital workflow of the two teams is, therefore, a critical success factor.

Research question:

> How can a digital workflow based on the parametric modeling be used as a method of design to increase the efficiency of spatial structure detail design?

Research goals:

- Proposing a digital workflow for design of gridshell connections, from design to fabrication.
- Proposing a gridshell design suitable for an automated digital workflow.
- Additional structural verification of the connection joints to compare the reliability of the digital workflow.
- Investigate the opportunities with use of digital workflow in collaboration between architecture, engineering and construction sector.
- Propose connection design for a timber gridshell cabin.

Preface

This thesis is written on behalf of the Department of Structural Engineering at the Norwegian University of Science and Technology(NTNU). For Øyvind Sunnvoll Rognes this thesis concludes a 5 year master degree in civil engineering, and for Helle Stam Faugstad it concludes a 5 years master degree in Engineering and ICT with a specialization in structural engineering.

The thesis has been carried out in 20 weeks of work during spring 2018. The study and is a part of a larger project of planning a Gridshell cabin for NTNUI, that emerged through a collaboration between the the Conceptual Structural Design Group(CSDG) at the Department of Structural Engineering, and the Student sports club NTNUI. Cooperation with PhD candidate at the Department of architecture and technology Steinar Hillersøy Dyvik, as well as another master thesis group from the Department of Structural Engineering consisting of Åshild Huseby and Marie Eliassen, has been essential for the completion of this thesis.

We want to thank our supervisors Professor Anders Rønnquist and PhD candidate Marcin Luczkowski for guidance and support throughout the spring semester. Thank you for all the time put down to help us complete this project, we truly appreciate it. We would further like to extend our gratitude to PhD Candidate Steinar Hillersøy Dyvik, Åshild Huseby and Marie Eliassen, whom we had the pleasure of working with in this project. We have enjoyed the cooperation, and found the process very enjoyable and educational thanks to you. Last we want to thank PhD Candidate John Fredrick Berntsen, M.Sc. Kirsten Faugstad and Civil Engineer Svein Olav Sunnvoll, as well as fellow students for professional guidance and support throughout the spring 2018.

Trondheim, 2018-06-07

[illegible]

Øyvind Sunnvoll Rognes

Trondheim, 2018-06-07

Helle Stam Faugstad

Helle Stam Faugstad

Contents

Chapter 1

Introduction

1.1 Architecture, Engineering and Construction (AEC)

In the 19th century, the building industry saw the rise of engineering, bringing about a more scientific approach to constructions. The old principles of structures used by architects were slowly dismissed in exchange of engineering theory, and the collaborative relationship between the engineers and the architect became more important. The 20th century saw the rise of computer technology and the power of fast calculations. With the development of the building industry towards more complicated processes, more documentation and interaction between different fields of expertise were needed. The use of computer-aided methods equipped businesses with more efficient tools to handle complex challenges.

Lately, with the introduction of digital collaboration platforms, like *Building Information Modeling (BIM)* in the 1970's, the focus on the collaboration and precision through a *Digital Workflow* has increased in popularity. According to National Bim Standard [1], BIM can be described as:

> ... a digital representation of physical and functional characteristics of a facility. A BIM is a shared knowledge resource for information about a facility forming a reliable basis for decisions during its life-cycle; defined as existing from earliest conception to demolition.

The methodology of digital workflow uses digital software to handle and coordinate different tasks of a more extensive procedure. Because of the close intertwinement of the three AEC subjects, establishing effective and efficient communication becomes essential for the success of a project. For example, a sudden change from the architect often leads to the engineers and manufacturers having to change calculations, documentation or machine settings. It is therefore crucial that the upset has a minimal effect on the project schedule.

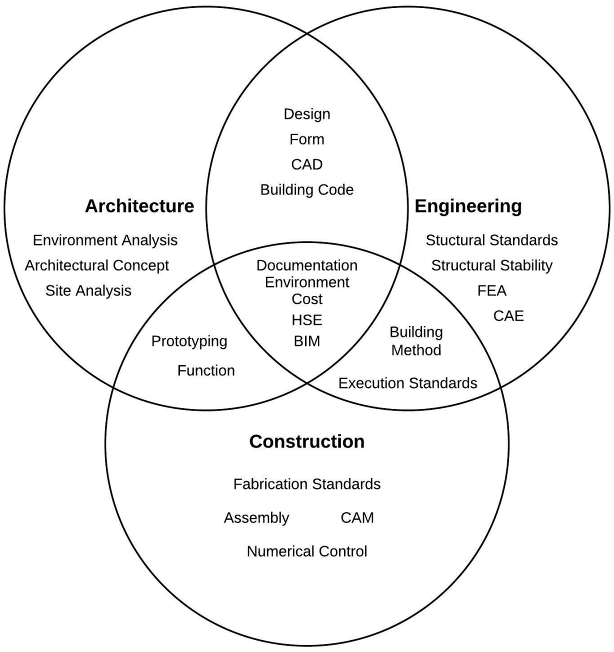


Figure 1.1: Intertwined and individual methodology in AEC.

Architecture, engineering, and construction involve very different processes and tasks. This is a very interesting topic since communication often can be a problem without cross-disciplinary understanding. Figure 1.1 shows the three subjects in a Venn-diagram, displaying how they have separate tasks and also where the three subjects meets. It is in the intersection between the different disciplines that the most interesting and innovative projects develops. As IPENZ [2] points out:

> A cultural shift is required for both engineers and architects. Engineers need to adapt positively to architects' iterative design approaches, while architects, drawing upon engineers' specialist expertise, must understand structural principles and incorporate core engineering requirements into their design imagination.

Having all the different design tools available also contributes to the developing and changing of the traditional roles for the architect and engineers. Calculations are automated, and the visualization of the whole structure is better. The process of developing a connection detail of a structure is an excellent example of when it is necessary for the engineer to make design decisions. It can be challenging to understand the details of how the force distribution of the connection works, since it demands a specific knowledge. Many of the decisions during the process of development depend on the force channeling and how the connection works statically. For situations where these kinds of decisions have to be made, the engineers becomes the only one capable of making such a design decision.

1.2 Concept of Shell Structures

The shell structures especially depend on the work of the engineer since the form of the structure often is directly derived from the flow of the forces. To make the structure work, it has to be light and robust. The challenge of constructing light, thin, and at the same time, strong roof structures led modern engineering to develop numerous different solutions. From the classical systems based around familiar elements, such as beams, columns, and slabs, to the free-formed designs of modern architecture, with, e.g. double curved shells or geometries with other irregularities.

1.2.1 Shell

To understand the structural reasoning behind the gridshell, it is important to understand the general shell structure and the type of forces that occur on a surface. Basic plate theory is used to explain this.

It is advantageous to divide the forces into the two following groups:

- Membrane forces (see Figure 1.2)
- Bending forces (see Figure 1.3)

The membrane forces are forces acting parallel to the surface. This includes both axial forces and shear forces. The bending forces consist of moments bending the plate around one of the membrane axes or shear forces acting perpendicular to the surface. Since the shell usually has a low thickness, is the corresponding moment of inertia often very low in comparison to the surafce area. Therefore, the shell might be badly suited for bending moments but well suited for membrane forces. These two groups of forces do not affect each other.

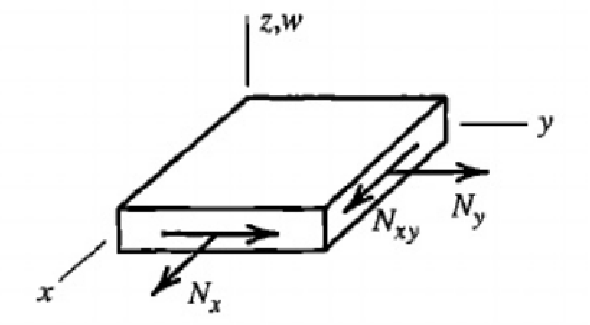


Figure 1.2: Plate membrane forces, [3].

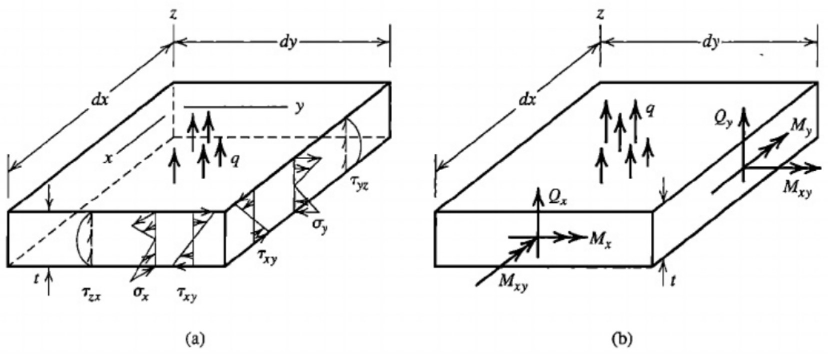


Figure 1.3: Plate bending forces. (a) Displayed as stresses. (b) Displayed as resultant forces, [3]

The motivation behind the shell structure is designing a structure that can span long without needing tall cross-sections to handle the bending moment. It is therefore in the interest of the designer to make the forces in the structure act as membrane forces as often as possible. Shaping the shell in a funicular shape turned upside down proves to be a solution to this problem.

The concept of using funicular shapes for shell structures originates from Robert Hooke (1635-1703). Hooke [4] proposed already in 1676 that a hanging chain turned upside down will stand in pure compression (see Figure 1.4). This concept is also transferable to 2D. By flipping a hanging membrane, we can get a shell which takes pure membrane forces. Ochsendorf and Block [5] point out that there is an infinite amount of possible funicular surfaces that can work in pure compression. It all depends on how much material is available and the distribution of it. Using surfaces with holes or free edges is also possible.

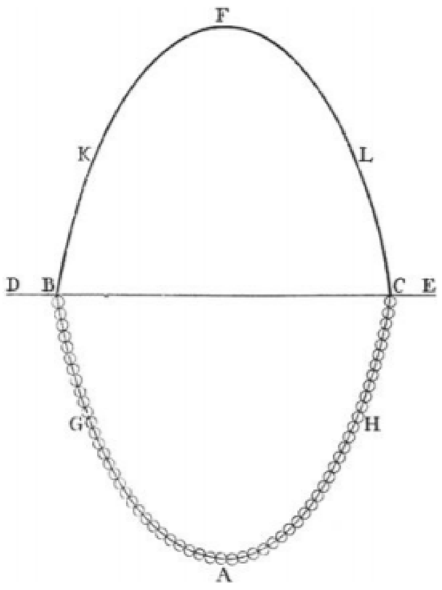


Figure 1.4: Hooke's hanging chain, [5].

Ney and Adriaenssens [6] distinguish types of structures between form-active systems and form-passive systems. Form-active structures are structures that change shape as a reaction to the forces. While the hanging cable becomes funicular due to the tensile forces acting, a gridshell must be placed in the desired shape when assembled since the deformations will amplify. Another way to describe it is thinking of the hanging cable as a system in stable equilibrium and the shell as a structure in unstable equilibrium. A consequence of this is that changes in the force distribution in the shell lead to a nonlinear increase in forces due to the deformation. This can particularly lead to challenges regarding unsymmetrical and horizontal load cases.

1.2.2 Gridshell

The gridshell distributes forces along bars instead of the shell surface. By applying the principle of "forces follow the path of least resistance (energy)," it is clear that a fine mesh with multiple possible force paths makes the gridshell share force flow properties with the ordinary shell. This shows to be a very efficient geometry avoiding large concentrated forces by distributing the forces more efficiently over the shell. The open topology of the gridshell also makes the gridshell light in comparison to compact shells.

The gridshell shares a lot of the properties with the beam grid. They are both built of bars connected to form a lattice, usually creating triangular, quadrilateral or hexagon openings in the surface. What differentiates the two types is the curved surface of the gridshell in contrast to the planar surface of the beam grid. A modern gridshell project which well represents a gridshell structure is the ceiling of the British Museum in London showed in Figure 1.5.

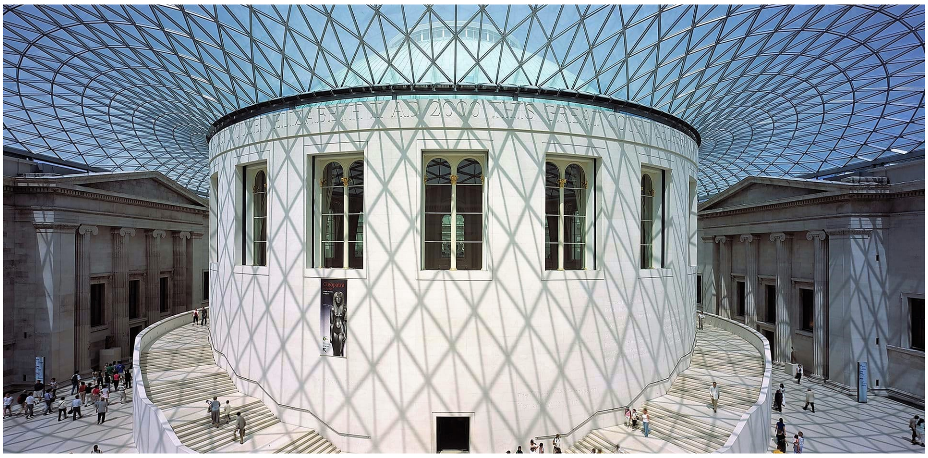


Figure 1.5: Gridshell structure in the ceiling of the British Museum in London, [7].
.

As the structure is no longer continuous, connections have to be designed carefully to distribute the forces from one bar to another correctly. The relatively small connections in a large long-span structure may seem somewhat trivial, but the connection is often a weak point in the structure and needs to be taken into consideration during the design process.

1.2.3 Kinematic Gridshell

A kinematic gridshell is a gridshell structure where the bars intertwine, and there is not a stiff connecting joint in the intersecting point of two bars. The first large-scale gridshell structure built, Mannheim Multihalle (1974) was built using this concept. An example of another kinematic gridshell structure, Trondheim Pavilion is illustrated in Figure 1.6.

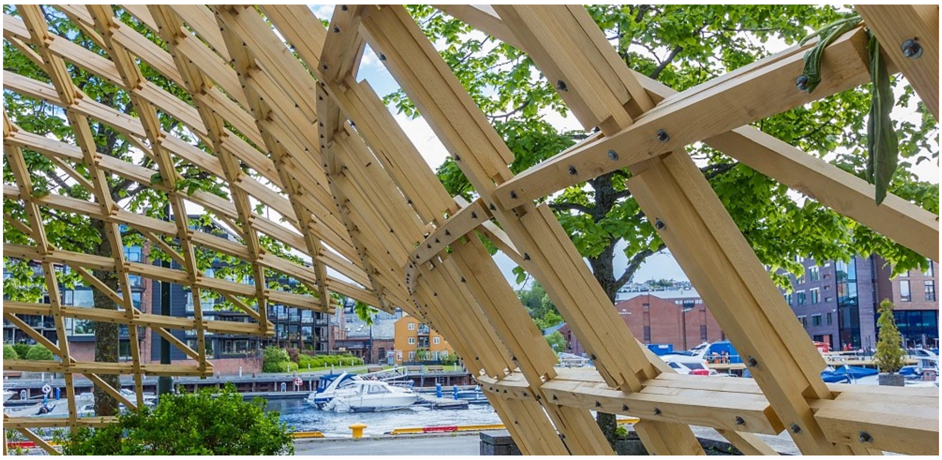


Figure 1.6: Kinematic Gridshell from Trondheim Pavilion, [8].

The process of making a kinematic gridshell is entirely different from assembling a normal gridshell. The kinematic gridshell construction process consists of building the whole grid flat. It can be built on the ground and then be pushed upwards to the desired form, or it can be built on a certain height, and lowered by gravity. The shaping process can either be done by lifting the grid from a point within the structure, or by pushing the anchor points from the edge to the correct position.

This shaping process is inducing bending moments in the structure. Unlike the normal gridshell, the kinematic gridshell is thereby a structure carried by bending moment in addition to the compression forces. It is therefore essential for a kinematic gridshell structure to be anchored with bending resistant connections.

1.2.4 Timber Gridshell

Timber gridshells is a modern and efficient way of providing unique architecture and at the same time taking advantage of material properties. Timber, if provided from certified sources, is by many considered as environmentally friendly. It provides a noticeable amount of CO_2 storage, which could lead to a negative carbon footprint [9]. The fact that it is well suited for long-span, light-weight, affordable and sustainable structures, makes it easy applicable to the modern paradigm of sustainability while it also fits the developmental architectural programs of today.

Different types of timber can be used, but often is *Engineered Wood Products* (EWP) quite suitable for structures like this. *Glued Laminated Timber* (Glulam) is suitable for this type of structure due to its lightness, ductile abilities, and strength. It shares a lot of the structural properties of traditional sawn timber. The fact that the material is orthotropic and noticeable weak in its weak direction is not necessarily a problem since the members usually are loaded axially. Hence, one is taking advantage of the benefits of timber as a material, which is assumed to be one of the main reasons why timber is so well suited for gridshell structures.

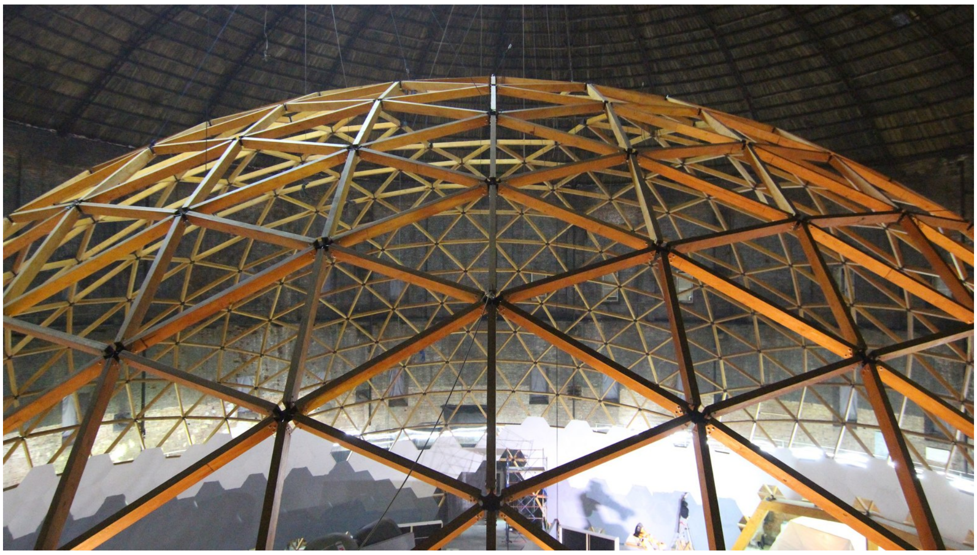


Figure 1.7: Timber gridshell from The Wooden Geodesic Dome in St. Petersburgh, [10].

1.3 Digital Workflow in Fabrication

Digital methods of fabrication is a relatively new subject of research. The development of the field has been growing rapidly since the second world war, mostly driven by the development of computer technology [11]. The topic treats the challenges of connecting the design process with the production process through digital workflow.

Caneparo [11] propounds that there exists a *digital continuum* between design and manufacturing, and claims that the digital methodology links design and craftsmanship closer together – despite the occasional view by designers of digitalization as a threat to the *physical aspect of design*. He further points out that there are primarily three technological methods that digitalize this continuum; *computer aided design, engineering or manufacturing (CAD, CAE or CAM)*, *numerical control (NC)* and *prototyping*. This thesis will focus primarily on the use of CAD, CAE, CAM, and prototyping as a methodology of design, but also investigate NC as a method of manufacturing.

According to Narayan et al. [12], CAD involves the use of computer technology to create, modify, analyze or optimize a design. This definition extends the definition of CAD beyond the conception of design only being about creating drawings and instructions, but instead of being a method for a more flexible design process. CAD offers a vast amount of different software, using different methods which can perform differently depending on the type of project.

CAD is often distinguished from CAE and CAM. By applying the proposed definition of CAD from Narayan et al. [12] to define CAE, it could involve the use of computer technology in engineering processes. A typical example of this is *Finite Element Analysis* (FEA), based on the *Finite Element Method* (FEM), which uses the large computing capacity of modern computers to make a discrete model of a system that approaches the analytic solution. In the same way, CAM might be the use of computer technology in manufacturing and fabrication. By combining CAD, CAE and CAM methodology, the people involved in a project can create a digital workflow in fabrication projects.

NC, also called *computer numerical control (CNC)*, is a method developed by American aerospace industry in 1949 [11]. The method aims to use digital instructions to control machines during production. The development was fueled by the industry's need for efficient production with low tolerance and high precision. According to Caneparo [11], the earliest example of the method is John Parsons and Frank Stulen's use of a programmable milling cutter to produce steel stringers for rotor blades in 1940. It used electric motors to move the cutter based on 200 control points programmed with punching cards.

The method of NC has developed immensely since the 1940's, and production is not limited to the number of control points, 2D or milling machines anymore. Laser cutters, milling cutters, lathes and 3D printers are examples of modern machines that make use of this technology with much more complexity than before. The 3D printer has created particular interest due to its ability to produce cheap components with complex geometry.

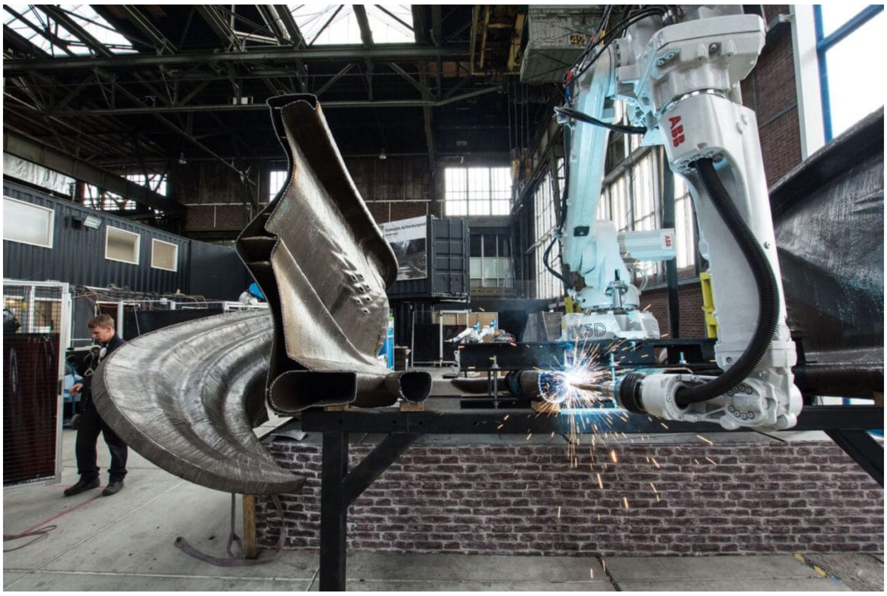


Figure 1.8: The MX3D team 3D-printing a steel bridge, [13].

A particularly interesting project exploring the use of 3D-printing in construction was the MX3D Bridge Project. According to MX3D [14], the goal of the project was to create an entirely 3D-printed steel bridge to cross the Oudezijds Achterburgwal Canal in Amsterdam. The project involved the use of CAD- and CAE-software to draw, analyze and optimize the structure, leading to a very complex geometry for fabrication. The monolithic nature and scale of the structure created the most significant challenges. It made, for instance, the structure very hard to cast. The solution was to 3D-print the entire structure using a welder attached to a robot arm. Using NC-methods, the machine could add more steel to the bridge according to the digital design.

For completeness of the description of the digital continuum in Caneparo [11], prototyping is defined by Chua et al. [15] as:

> An approximation of a product (or system) or its components in some form for a definite purpose in its implementation.

Prototyping is particularly efficient when production cost is low. The technological development, leading to a more efficient production with less use of workers, has therefore opted for the possibility of more prototyping in design.

When deciding on the methodology, the digital workflow has to reflect the specific character of the project. Some examples of this could be the need for mass production of standard parts, or production of a custom high-performance part. If a process needs mass production of standard parts, it should be possible in the CAD-software to easily copy or duplicate the model and make minor changes if needed. The fabrication method should be able to produce the same part over and over again. This could, for instance, be solved by creating one mold using NC and re-using it for molding multiple parts. For fabricating a customized part, the CAD- and CAE-software might need to be more advanced, giving the user better opportunities regarding analysis and optimization. The CAM-method of choice should also be able to tackle the irregularities that occur as a result of the design.

The most significant challenge comes to situations with a need for mass production of customized parts. Suddenly there has to be an NC-method with different machine instructions for every part, and every single part needs their CAD- and CAE-model, and all this must still be cost-efficient. The customization of gridshell connections could be such a process. Every connection needs to be able to handle incoming beams with a different orientation. This makes the design requirements different for every single connection. 3D-printing as a NC-method yield good accuracy's regarding fabrication of custom parts. The machine needs few hands to operate, has high precision and is suitable for different machine instructions. The downside is slow production speed and high cost per part. The CAD- and CAE-methods for such a project needs to balanced between the ability to optimize and the ability to duplicate. Such a method could be a *parametric approach.*

1.4 Parametric Approach to Computer Aided Design

The design process demands a significant amount of decision making, and the fundamental design choices can often lead to results of very different quality. An inefficient solution can often be expensive while a well designed, more efficient solution often takes time to accomplish. Having tools that ensure an efficient digital workflow can help the designer make more efficient choices and hence easily improve the quality of the project.

Parametric modeling uses CAD as a method of design to model the structure. The approach focuses on the possibilities of changing the design and analyze the corresponding effect. The basis of the concept is to represent different characteristics of the design by parameters, so effects in the model are displayed as the designer changes the parameters.

This approach fits with the challenges concerning gridshell connections. Parameters can represent the characteristics of the different connections, and be changed during the process. The process of verifying the structural stability can be implemented in a model such that the model can be optimized on the basis of this. For gridshell connections, this is primarily about ensuring that the beams connects properly, and that the connections transfer the forces correctly. There must be algorithms to verify the quality of every single connection. Still, the designer decides how to prioritize the changes in the model when a parameter is changed.

The parametric approach is although a comprehensive process. The designer must develop new custom tools for the current project, or seek out and combine existing tools. The tools must be robust, as they need to handle every possible problem that can occur. This is something which leads to a broader and far more complicated calculation scope than the often pragmatic approach in traditional engineering.

1.5 Manufacturing Methods for Aluminum

The different manufacturing methods presented in this sub chapter are suitable for a digital workflow, as the CAD geometry can be used in the manufacturing process. NC-instructions with the designed connections will now be sent directly to the manufacturing, automatically generated by the parametric model. Demands from the manufacturing process shall also be implemented in the digital workflow to ensure a seamless process. The different methods will be presented briefly for completeness, not for a discussion of selecting manufacturing method, as this is out of scope for this thesis.

1.5.1 Additive Manufacturing

According to [16] the definition of *Additive Manufacturing* (AM) is:

> A process of joining materials to make objects from 3D model data, usually layer upon layer, as opposed to subtractive manufacturing methodologies.

The application of this technology has in many ways changed the manufacturing sector. It is a new and innovative technique, and has caught great interest in both the AEC-industry, and received a great deal of publicity [17].

For AM the material could, for instance, be plastic, aluminum or concrete. Common for AM technologies is to use a CAD representation of the intended geometry. From the created CAD model, the AM equipment reads the data and adds the layers of liquids, powder, sheet material or other, using the layer-upon-layer technique to fabricate the 3D object. A typical production technique associated with AM is 3D-printing, but AM also includes, for instance, Rapid Prototyping (RP) and Direct Digital Manufacturing (DDM).

Aluminum 3D Printing

Aluminum is a popular material because it has a unique combination of properties, which makes it desirable for many applications. One of the attributes that make aluminum stand out from other materials is its high strength-weight ratio, which is why it is used for aerospace components, laptops, phones and sports equipment. It can be machined, welded, polished and coated. Aluminum also has good thermal conductivity [17].

For aluminum, there are different methods of 3D printing. Selective Laser Sintering (SLS) is a method where the aluminum powder is not fully melted, but instead heated up to a specific point where the powder grains can fuse together. This is allowing the porosity of the material to be controlled. However, since the powder is fused together, and not completely melted, this method will not give a homogeneous and as strong material as other methods [17].

The Direct Metal Laser Sintering (DMLS) method, also often referred to as Selective Laser Melting (SLM), is another method used for 3D printing in aluminum that goes one step further than SLS. This is an advanced manufacturing method that melts the powdered metal layer-by-layer. The printer uses a high powered laser that binds the metal particles together in the melting process. Unlike the SLS method, this method can entirely melt the metal powder into a homogeneous, solid 3D part. For this method, an automatically generated support structure is needed. This structure will later be manually removed. 3D printed models in aluminum are known to be very strong, precise and can handle details of 0.25 mm. This technique will also make it possible to make complex geometries that would be very difficult or impossible to obtain with other manufacturing methods, e.g. interlocking parts [18].

1.5.2 Aluminum Sand Casting

Additive manufacturing has proven its potential in many different fabrication areas, but despite this potential, most of the techniques still have limited production dimensions, high cost and advanced, and sometimes non-optimal material properties. The process is also quite time-consuming, especially for bigger dimensioned objects. In many cases, sand casting will be faster, have the same high precision and have lower material price [19].

Most of the metal castings are produced by sand casting. Sand casting is a quick and cost-effective production method, that is especially suited for prototyping and developing new products that don't require an accurate shape repeatably. The sand is especially well suited for this purpose because it is refractory and chemically inert, and at the same time a low-cost material. As the sand mold must be destroyed to remove the casting, sand casting typically has a low production rate. At the same time, sand casting may be the only solution for very large objects which cannot be produced with other mass production casting techniques [20].

The process of aluminum sand casting most commonly uses *Green sand* as the aluminum expandable mold. Green sand could be new, or regenerated sand which is mixed with natural or synthetic binders [20]. The Green sand will still be moist when the aluminum is poured into the sand mold. As the sand is refractory, both sand and the replica of the object is reusable, which makes this manufacturing process sustainable [21].

This mold is placed on a replica of the casting object. This first part of the process is usually done by machinery to obtain the highest precision possible, though it could be done by hand. Then the replica of the object is removed, and the cavity in the sand mold will have the shape of the object to cast [20].

The sand mold must have at least two parts, sometimes more. The upper part is called the cope, while the bottom part is called the drag. The sand mold is formed in a two-part box for protection. This box is often called a flask. The two molds are uncased in the flask halves, and before it is closed a hole in the sand, called a sprue, is formed in order to allow the molten aluminum to be filled into the cast using the so-called gravity filling method [20].

The mold is closed and clamped together, and the molten aluminum is poured into the sprue. Then it flows into the mold cavity and into the negative space left by the replica of the casting shape. Before removal of the sand, the aluminum needs to cool down. Because the green sand does not absorb heat, this cooling process is much longer than other casting methods. Because of this, the material properties are noticeably decreased. Also because of the heat from the molten aluminum, the moist in the sand is dried out, which allows the cast to crack open when the aluminum has cooled down. Common defects from aluminum casting are a residual oxide film, inclusion, core erosion, gas holes and shrinkage porosity [20]. The principle of sand casting is illustrated in Figure 1.9.

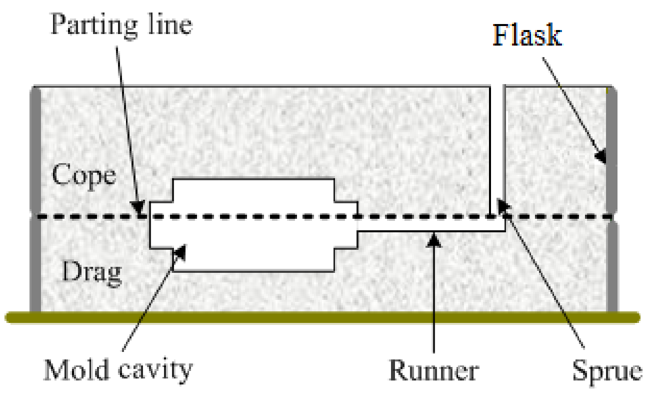


Figure 1.9: Principal of sand casting, [21].

1.6 Digitalization in AEC

As the digitalization changes the everyday of structural engineers all over the world, it is essential to keep up with the changes. Being in the lead of this development involves innovative use of digital methods. This will again lead to a more efficient workflow, making digital competence essential to cope with competition in the industry now and in the future.

The different tools presented in this thesis will, if used correctly, both simplify and clarify the digital workflow within AEC. The AEC-industry uses extensive information flow between and in the different fields. This methodology of design has a high potential to equip smarter and more efficient tools to create an interesting and cost-effective design. The workflow and tools exemplified in this thesis will display design methodology that exploits the computer's ability in data processing and algorithmic logic.

Different concepts for gridshell connections will be presented along with general design principles for connections and parametric models. A parametric model of all the connections within a given gridshell is produced in Rhino with the plug-in Grasshopper for the parametric modeling. The model is optimized using Karamba FEA Software along with implemented Eurocode checks for the different materials. Parts of a final connection is then verified using Abaqus CAE Software.

The model will be used in a case study to design valid gridshell connections for the roof of a spatial cabin. For a master thesis that is shaped based on a concrete case study, it is easy to focus on the project design rather than the research question. The goal of solving the case efficiently can steal the attention from the overall goal of the research, mainly since the process of design has to be efficient to satisfy all involved parties. It is, therefore, worth to notice that the following thesis tries to find a balanced approach to the two goals, both studying the vast possibilities of digital workflow, but also early proposing a digital workflow and a general design for the connections.

Chapter 2

Connection Design for Timber Gridshells

When optimizing any structural component, an open and general approach enabling a vast amount of different solutions will, with enough time, often result in a better and more efficient structure. In theory, if a software had an infinite amount of time to investigate grid shell connections, it could look at every way of attaching the bars, with every type of material, with every type of attachment and with every type of shape. Such a software would be able to create the perfect custom solution for every grid shell configuration possible.

A software like this would never be useful in practical engineering, where time means money and the customers demand efficiency. This points to the conclusion that there must be some common restrictions and limitations on the design, and the limitations must be based on economical, structural and practical considerations.

From a practical perspective, it is essential that the connection is based on more or less well-known techniques for attaching the timber beams to each other, e.g., nails, bolts, dowels or even carpenter joints. This might seem like a trivial thing to point out, but it is essential to understand the logic behind the approach used in this thesis.

The economic and structural limitations is often strongly connected. More material means better structural performance, but at a cost. There is a vast amount of advanced computational techniques to optimize based on these considerations, like for instance topology optimization. Using traditional engineering reasoning is although a much cheaper option where calculation time can be severely reduced by conservative design decisions, that simplify necessary calculations. Methods like topology optimization is also more expensive due to customized fabrication.

2.1 Status About Gridshell Connections

Gridshells are often more efficient ways of covering large spans than traditional shells. As this kind of structure still is considered as an emergent technology, the design of such buildings is mostly developed through a substantial amount of experimental work. It is therefore interesting and useful to further investigate existing gridshell projects.

Connections in timber structures are often the weak point, and the limiting factor for the gridshell structure. It is especially advantageous to look at reference projects due to the limited amount of large-scale gridshell structure. Especially crucial for connection in this kind of structures, is its ability to make room for the natural shrinkage and expanding of the wood. The connections designed for kinematic timber gridshells additionally needs to allow movement for the construction process, as the structure often is built on the ground and lifted up to the correct form, or built on certain height and lowered by gravity. As mentioned earlier, the focus will be on gridshells, more than kinematic gridshells, but for completeness, this section will still present the most common types of connections for kinematic gridshells [22].

2.1.1 Slotted Hole Connection

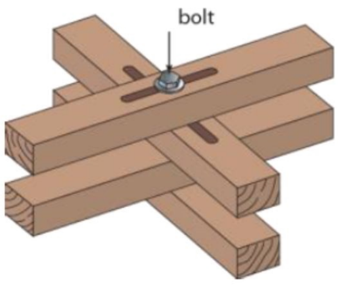


Figure 2.1: Slotted hole connection for a double layered gridshell, [23].

The slotted hole connection illustrated in Figure 2.1 can be used on both single and double layered kinematic gridshells. It is a quite simple solution for a connection where the size of the slots can be modified according to the necessary movement during construction, but also movement from shrinkage of the timber. The benefits of this solution is that one does not need an additionally manufactured connection. The only thing needed, other than the timber itself, is the bolt. A disadvantage is that the members need to be slotted, and there will be a weakness of the material in this area. It will also be essential that the slot is not made in a part of the timber where there is other weaknesses. This is the connection type that was used in the first large-scale timber gridshell, Mannheim Multihalle in 1974 (Figure 2.2 and Figure 2.3) [24].

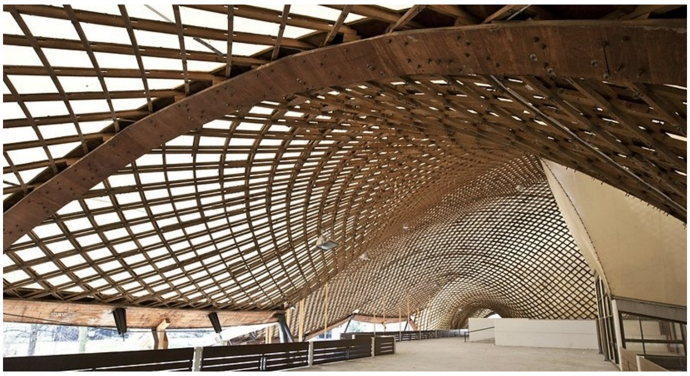


Figure 2.2: Mannheim multihalle, [25].

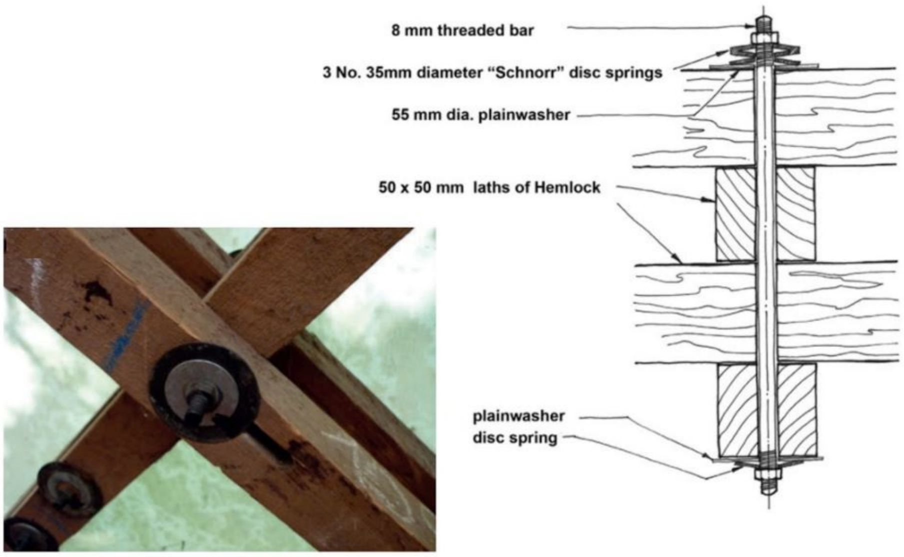

8 mm threaded bar
3 No. 35mm diameter "Schnorr" disc springs
55 mm dia. plainwasher
50 x 50 mm laths of Hemlock
plainwasher
disc spring

Figure 2.3: Connection detail from Mannheim multihalle, [26].

2.1.2 Plates and External Bolts Connection

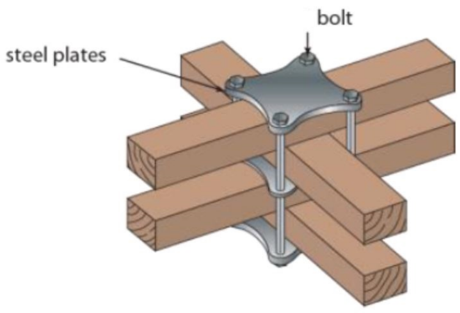


Figure 2.4: Plates and external bolts connection for a double layered gridshell, [23].

This is a patented connection first suggested by Andrew Holloway. The plates and external bolts connection is used for kinematic gridshells. For a double layered kinematic gridshell it consists of three clamping plates connected by four external bolts (see Figure 2.4). The bolt does not pass through the timber and therefore does not weaken the timber laths. This connection allows the timber laths to slide and rotate during the construction process, which is essential for kinematic gridshells. The improvement from the slotted hole connection is the way the intersections now can be fixed into place. To fix the position of the joint, and to ensure constant spacing in the whole structure, the middle plate in the connection has a point that inserts in the two central timber laths. The two outer layers are then free to rotate and slide freely to find the optimal position [24]. This intersection technique was used in the Weald & Downland Gridshell in Sussex, which was completed in 2002 (see Figure 2.5) [26].

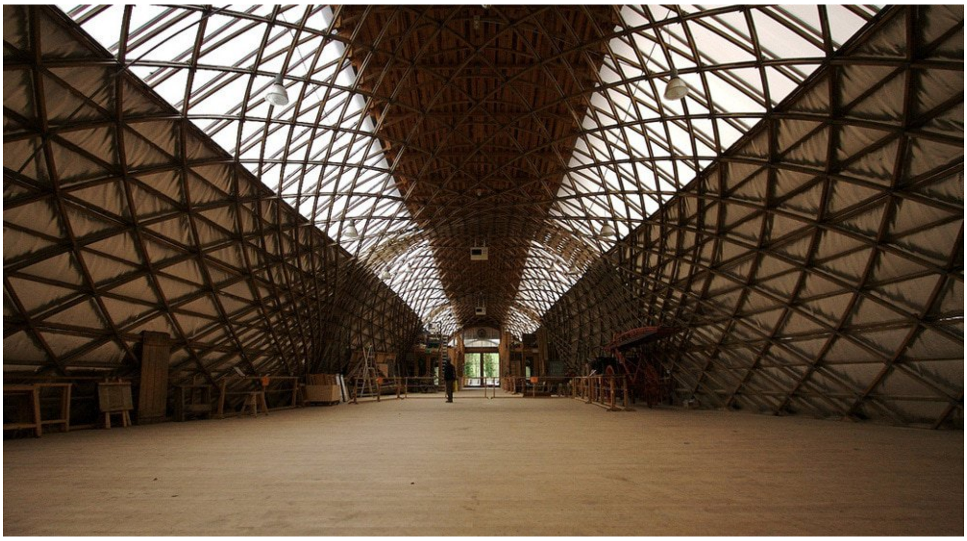


Figure 2.5: Downland Gridshell, [26].

2.1.3 Ball Joint

Figure 2.6 illustrates the principle of the ball Joint used in the Computer Morphogenesis Lab project at Politecnico di Torino. The ball joint is spherical shaped, which gives it great flexibility in the sense that the bars can be connected perpendicular to the joint at all parts of the surface. With perpendicular bars there is no eccentricity, hence no bending moment in the joint. This is especially advantageous when used with soft material like timber [27].

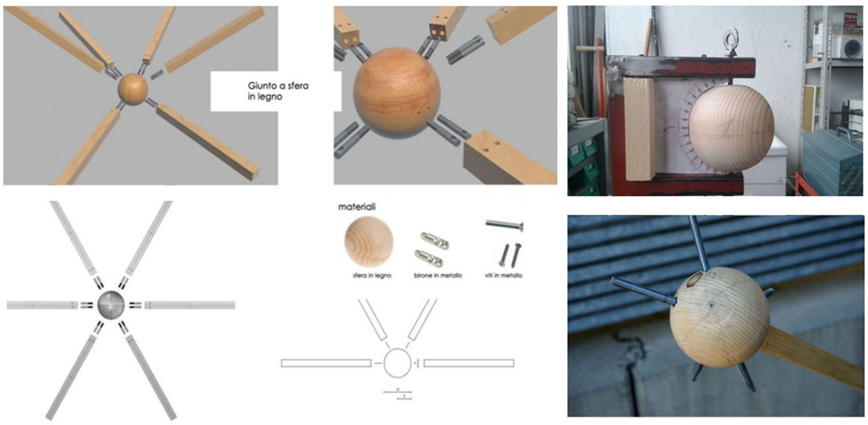


Figure 2.6: Ball joint used in the Computer Morphogenisis Lab project at Politecnico di Torino, [28].

Another reference project worth mentioned is the Roppe Bridge - a rotational parametric pedestrian bridge developed by architect Andrej Nejur and Szende Szentesi. For this project, all the geometry was developed using Grasshopper [29]. The joint consist of a plain ball joint with several connected beams. The beams are connected with crossed slotted plates that provide a certain stiffness in both directions of the cross-section [30]. Similar to the project in Turin, the ball joint is flexible in the way the beams are connected. A disadvantage of this solution is the amount of material used in the sphere. The connection is massive and similar for all the different nodes, which means that there is a lot of unexploited area in the joints.

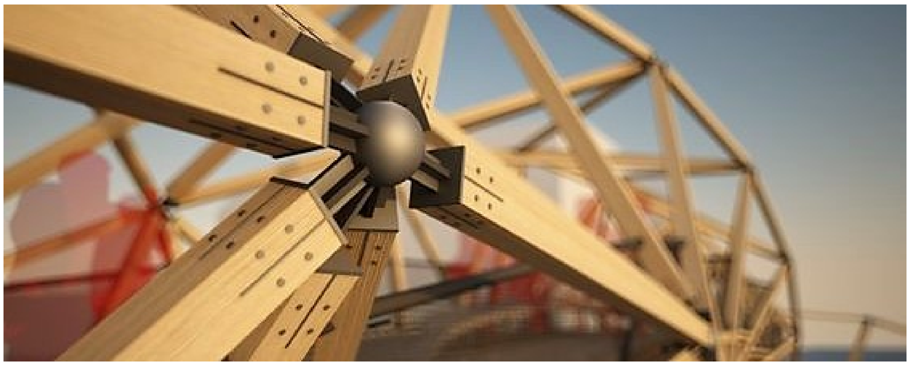


Figure 2.7: Ball joint used in Roppe Bridge project in Romania, [29].

2.1.4 Bolted Steel Plate Joint

The connection was developed during SUTD Library Pavilion project in 2013. The connection is illustrated in Figure 2.8. A steel plate is bolted onto a piece of plywood bolted between two flat profiles forming the bars in the grid. The incoming members to the node has a certain thickness, which leaves an empty void in the center of each node. The team had a structural challenge with transferring the forces continuously through the node, and came up with this solution. To achieve a direct load path, the steel plate is placed on the top and bottom of the joint. Since the edges of the plywood still are in contact with each other, this solution will establish two different load paths. One directly through the steel plate and one following the edges of the plywood [31]. This is illustrated in Figure 2.9.

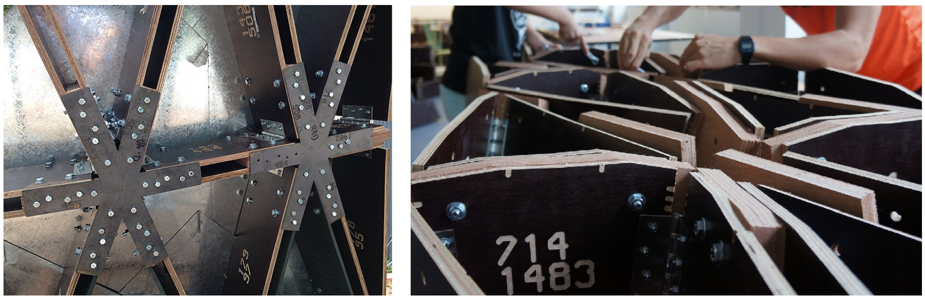


Figure 2.8: Detail of the bolted steel plate connection, [32].

A disadvantage of this connection type is the limited customizability. For the specific SUTD Library Pavilion project, the dimensions and incoming angles of the different beams is more or less equal for the whole grid shell (see Figure 2.10). When developing a more dynamic and customized structure with steep angles, this connection type would probably be difficult to use. Beams with different dimensions and angles result in an eccentrically loaded connection, and unwanted bending moments will appear. The angles of the beams in the plane of the steel plate can easily be adjusted by cutting the steel plate according to these angles. Rotation in the other two planes will be more problematic and cause eccentricities.

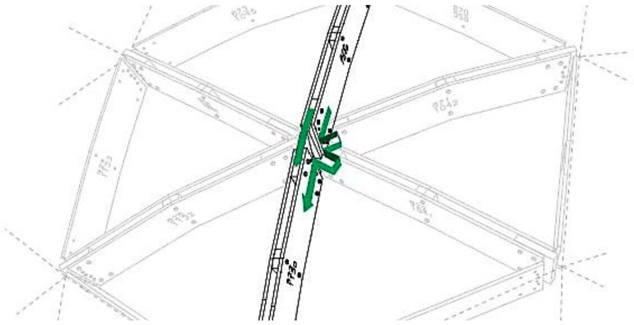


Figure 2.9: The two different load paths through this node illustrated by the two green arrows, [32].

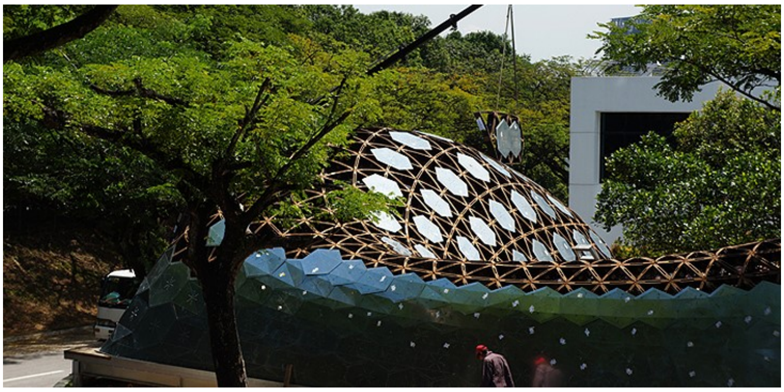


Figure 2.10: Overview of the SUTD Library Pavilion project, [32].

2.1.5 Circular Tube with Welded Plates Connection

For the Lo-Fab Pavillion project [33], the Mass Design Group used cylindrical tubes with welded plates as connections as illustrated in Figure 2.11 and 2.12. The joints were automatically assembled in a custom robotically assisted welding process and fabricated using a combination of robotic fabrication and traditional craftsmanship. The structure contained 1880 steel parts making up 376 joints.

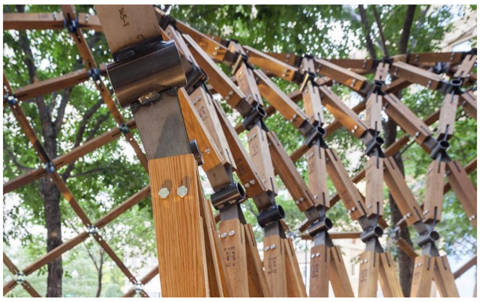


Figure 2.11: Lo-Fab Pavilion, Design Boston Biennial, [33].

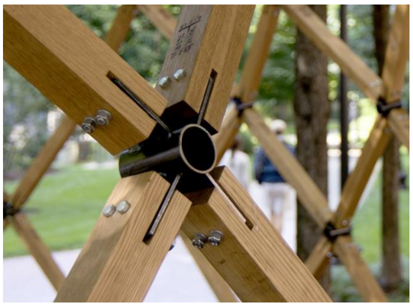


Figure 2.12: Lo-Fab Pavilion, connection detail, [33].

With this amount of parts, the construction process will have some practical issues. This aspect is the main reason why an automatically robotic workflow makes the process a lot easier [33]. Therefore this project is an excellent example of how practical issues must be taken into consideration during decision making early in the process. Properties that should be taken particularly into consideration for this connection type would be, e.g., the meeting edges of the bar. In the Lo-Fi Pavillion project is it only four connected bars, so this would not be a concern.

Considering another similar structure, The Wooden Geodesic Dome in St. Petersburg, this had to be taken into account [10]. Connection detail is illustrated in Figure 2.13. The steel plates had to be long enough to avoid the intersection of the beam corners. For this project, there is also added stiffening plates between the slotted plates to the connection, as illustrated in Figure 2.14. This steel connection is called a "Haeckel." As the plates in this kind of connection do not give much support to rotation perpendicular to the plate, it would be necessary with extra stiffening in this direction in some cases. The "Haeckel" would be one way to solve this possible problem. When avoiding the beam edges to intersect, the force path will go directly through the steel node. The steel ring, slotted plates, bolts and timber bars all together have to provide the necessary capacity.

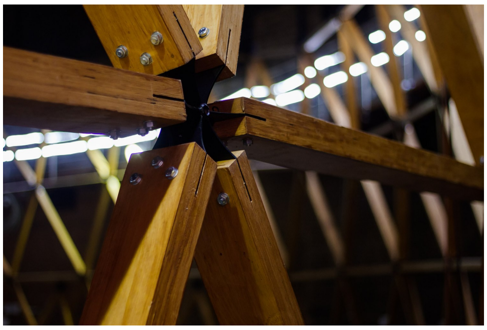


Figure 2.13: Detail from The Wooden Geodesic Dome in St. Petersburg, [10].

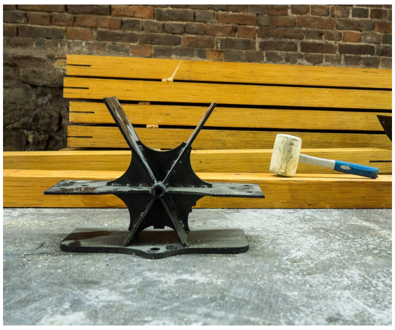


Figure 2.14: Joint used in the Wooden Geodesic Dome in St. Petersburg, [10].

2.2 Designing Timber Connections

The properties and behavior of timber makes it a demanding material in design. Timber is restrictive to a multiple of manufacturing processes that involve joining processes. While steel and aluminum can be welded and concrete can bond, timber is restricted to dowel connections. Timber also demands extra care for its very orthotropic properties. It has high strength in the fiber direction, but it is very weak perpendicular to the fiber direction. Hence, to orientate the timber such that the strong axis takes most of the forces is of importance [34].

Additionally, different conditions like load duration and relative moisture have to be taken into account. Because of the hygroscopic properties of wood, it will swell and shrink depending on the relative moisture. The connections are often the weak point in the structure, and therefore they can determine the load carrying capacity of the whole structure. As a poorly designed connection can lead to brittle fracture, it is not only the strength of the connection but also its ductility that determine its quality.

The geometry of the connection will in many cases inflict damage in the wood. By making holes and slots in the timber, its effective cross-section will be reduced. The metal parts that are used in the connection can, due to corrosion and low carrying capacity at high temperatures, become the weakest element in the connection. This is also important to take into consideration in the design process [35].

Because of the limitations of traditionally sawn sections of wood, EWPs (Engineered Wood Products) has developed a lot. EWPs comes in different forms, like cross-laminated timber (CLT), laminated veneer lumber (LVL) or glued-laminated timber (glulam). Many of these products overcome a lot of the limitations of traditionally sawn timber and are therefore becoming popular for structures like timber gridshells.

These products can be bent and shaped on a much greater scale than traditional timber. Regarding connections, products with increased strength in multiple directions will have enhanced connector strength and splitting resistance.

2.3 Design Requirements

When developing a connection design, certain attributes have to be taken into account. Succeeding in such a development process means having considered all of this. For a digital workflow, this is especially important because many of the decisions that have to be made during the development. When establishing the design requirements, one has to make sure that all the connections obtain sufficient quality. In this section, essential design requirements for connections in gridshells will be studied.

2.3.1 Customizability

In a process where the connection is going to be optimized depending on the different forces working on it, it is essential that the design allows adjustments. The forces in the connecting bars can vary between compressive and tensile forces, shear forces or moment force. Hence, the expected properties and dimensions will then vary depending on the load case. Other parameters that need to be adjustable is, e.g. number of connected beams, angles of the connected beams, the cross-section of the connected beams, and number of bolts.

It is essential that the design is equipped with components that manage to transfer the loads desirably. Since the connections in a grid shell can have a varying number of connected bars with varying angles, it has to be ensured that for all these different connections, the forces are checked correctly.

2.3.2 Material Efficiency

Like many other design problems, a critical factor is the strength-to-material ratio. It is desirable to achieve as much strength as possible from as little material as possible. With this said, many will immediately think of topology optimization, the mathematical method that optimizes the material layout by minimizing the amount of material on a given design space. An example of a closely related project using this principle in the design is the pavilion structure presented in Williams et al. [36]. The fabricated plastic nodes are topology optimized and manufactured through Fused Deposition Modelling

(FDM) as illustrated in Figure 2.15. for this thesis, topology optimization will be out of scope.

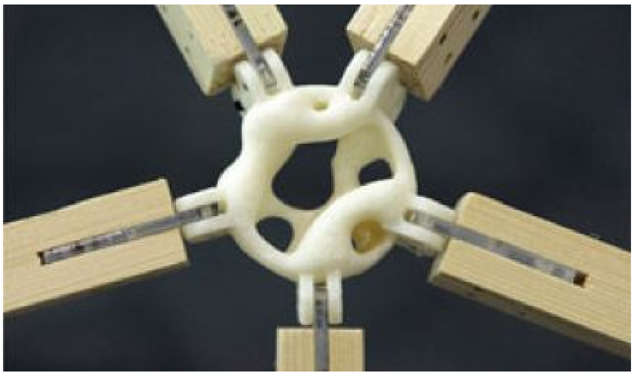


Figure 2.15: An example of a fabricated prototype node, [36].

One way of making an as efficient design as possible is to make conscious priorities of what parameters to modify in the optimization process. This is described more detailed in Chapter 4.4.3. To maximize the efficiency of the connection, it would, for instance, be better to increase the diameter of the bolt before adding a new bolt with the previous diameter. It will also be better to increase the bolt diameter before increasing the thickness of the plate.

2.3.3 Assembly

It is easy to forget the assembly process as the design process often focuses on the structural aspects. It is essential to make sure the proposed design is feasible and convenient for assembly. Included in this consideration are the number of individual parts that need to be assembled, and the total weight of the materials that need to be transported. With fewer parts, it will be less work, and with less weight, it will be faster assembly and transport. It is also important to imagine how the assembly is going to be carried out. If there is a significant number of different parts, it would be useful with a numbering system, or another system to help to keep track of the process.

Another practical issue related to the assembly process is the insertion of the bolts. The beams have to be attached to the connection in such a way that it will be possible to insert the bolts and tighten the nuts. At least one of the angles needs to be big enough for this, and the bolts for one of the connecting beams has to be assembled last.

In general, for the assembly, it will be advantageous with as many similar parts as possible. E.g., the same bolt dimensions for the whole structure, and the same connections for similar situations. This will make the assembly a lot easier, and also cost-efficient. At the same time, parametric optimizing of the structure will find individual solutions for each connection. Trying to find a combination of the individual solutions and practical limits will be an essential challenge. It is also important to point out that the optimization can be done based on a lot of different criteria's. Considering this as a quite comprehensive and time-consuming process, parts of this will be out of scope for this thesis.

2.3.4 Structural Verification

Having a connection consisting of well-known components will, in general, implicate having well-known verification methods. The Eurocode has well-known verification methods for individual parts of a connection. As a connection consists of different materials, different Eurocodes needs to be involved in the verification process. Having a design that is not too complex will also simplify structural verification in other software as well.

There are vast amounts of standardized methods in the structural codes regarding calculations of dowel type connections. By smart positioning of the dowel connections to fulfill the requirements, the calculation methods can be simplified and reliable. It can, for instance, be advantageous if the dowels are placed in rows and columns, as the calculation methods for such dowel configurations are well known.

2.3.5 Manufacturing

Different methods of manufacturing is presented in Chapter 1.5. In the design process, it is essential to take the necessary considerations to make the manufacturing process as efficient as possible. It is important not to make the geometry too complicated, as many methods of manufacturing are limited when it comes to geometry and size. Making the right design decision for the manufacturing also depends on the chosen material. Most materials have strengths and weaknesses, and the manufacturing will often influence these. It is essential to choose a manufacturing process which takes advantage of the material strengths.

2.4 Design Proposal

The design proposed in this thesis is based on well-known and simplistic components which offer the possibility to dimension the connections using standardized methods.

The center of the connection consists of a thin-walled cylinder, connected to four plates that are angled according to the incoming beams (see Figure 2.16). Then the plates are slotted in the glulam beams and fastened with bolts in a regular rectangular grid.

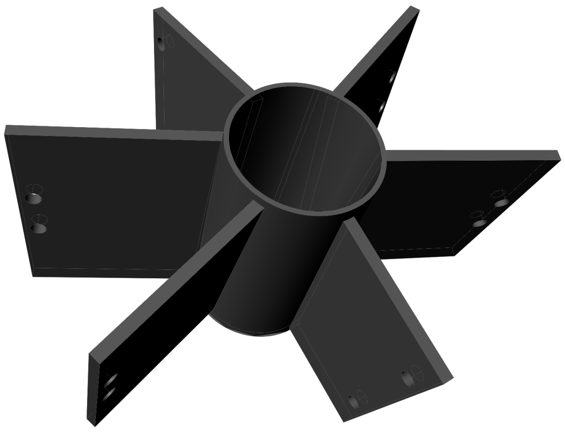


Figure 2.16: The design proposed, based on the requirements and specifications.

Using a cylinder as the core of the connection makes sense considering both production and structural characteristics. The cylinder distributes the forces between the gusset plates and makes it relatively stiff in comparison to other designs. At the same time, the cylinder is moving some of the mass out of the center of the connection, making the buckling length of the gusset plates shorter, and reducing the mass by not letting the gusset plates meet in the middle. Making the cylinder thin-walled will save material and at the same time maximize the effect of the material used. It also makes sense considering production, since the shapes both can be cast or welded from standard parts.

The slotted aluminum plates are calculated according to the Eurocode 9 [37], Eurocode 3 [38], and Eurocode 5 [39], using well-known formulas. Depending on the method of fabrication there might be particular verification methods needed.

As seen in the Lo-Fi pavilion project in Chapter 2.1.5, a similar design with cylinder and plates has been done before on a double curved roof. Hence it should be possible to implement for other similar projects parametrically. Another inspiring and close related project is Roppe Bridge mentioned in the section about Ball joints (see Chapter 2.1.3). The models were fully parametrically developed in the parametric CAD-software Grasshopper, and the design is flexible for parametric adjustments. In the project, the structure required rotational stiffness in both directions, hence the crossing plates.

The proposed design is not very complicated, and possible to customize in many ways. The dimensions of each connecting plate, bolt holes and bolts can be adjusted individually. The diameter and thickness of the cylinder can also be changed according to the needs for the individual connections. Considering a double curved shape, the angles of the connected plates will have to vary in all three rotational directions. Using this design, all of the above attributes will be feasible, and it will just be a matter of adjustments in the parameters. All the modifications can be made advantageously in a parametric model.

Chapter 3

Structural Verification of Connection Design

An essential part of the digital workflow, is the structural verification of the connections. The methods shall be general, such that every possible configuration of the connections can be verified. The simplicity of the methods is therefore important, as it both simplifies the process of programming and the necessity for documentation showing the validity of the program.

All the failure modes can be separated to three categories for the connection: Failure in timber beam, gusset plate and cylinder. The following chapter will briefly introduce the failure modes and establish a workflow for calculations.

3.1 Connections with Multiple Dowel Fasteners Loaded Eccentrically

For connections with multiple fasteners such as screws, bolts or nails, it is essential to determine the forces working on the different fasteners initially. For a concentrically loaded connection, the forces acting on the different fasteners are equal and parallel to the external load:

$$F_{\mathrm{f}} = \frac{N}{n_{\mathrm{f}}} \tag{3.1}$$

If the connection is loaded eccentrically, the load distribution changes. The eccentrical force will give momentum to the group of fasteners. The load from this force acting on fastener i was proposed by Larsen and Enjily [34] to be:

$$F_{\mathrm{m,i}} = \frac{r \cdot R}{I_{\mathrm{p}}} \cdot r_{\mathrm{i}} \tag{3.2}$$

Where I_{p} is the moment of inertia of the fastener group, r_{i} is the radius vector to bolt i, R is the force, and r is the distance from the origin. Note that I_p is defined without regard to the cross-sectional area of the bolt, meaning the formula only works for bolt groups with equal bolt size. The moment of inertia is therefore defined as in Larsen and Enjily [34]:

$$I_{\mathrm{p}} = \sum_{\mathrm{n=1}}^{n} ({x_{\mathrm{i}}}^2 + {y_{\mathrm{i}}}^2) \tag{3.3}$$

Where x_{i} is the distance from the origin in x-direction, and y_{i} the distance from the origin in y-direction. If a connection is subjected to both a concentrically and eccentrically force, the different contributions for each fastener have to be summed up. It is also advantageous to separate between the contribution in x- and y-direction, as capacity calculation for timber often separates between loads in the fiber direction and perpendicular to it.

3.2 Spacing of Bolts

For the calculations of the gusset plates in the connection, it is important to verify that the bolts are spaced sufficiently. For aluminum plates, recommended values for minimum spacing can be retrieved from Eurocode 9, Table 8.2 [37]. The values are displayed in Table 3.1.

Table 3.1: Minimum spacing for bolts, [38].

Distance		Minimum
End distance	e_1	$1.2 \cdot d_0$
Edge distance	e_2	$1.2 \cdot d_0$
Spacing	p_1	$2.2 \cdot d_0$
Spacing	p_2	$2.4 \cdot d_0$

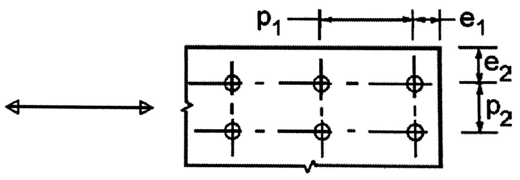


Figure 3.1: Symbols for spacing of multiple fasteners, [39].

For bolted connections in timber, minimum spacing is given in Eurocode 5 [39] and given in Table 3.2:

Table 3.2: Minimum spacing for bolts, [38].

Distance		Angle	Minimum
Spacing within one row parallel to grain	a_1	$0° \le \alpha \le 360°$	$(4+\|cos\alpha\|)\cdot d$
Spacing of rows perpendicular to grain	a_2	$0° \le \alpha \le 360°$	$4\cdot d$
Distance between bolt and loaded end	$a_{3,t}$	$-90° \le \alpha \le 90°$	$\max[7\cdot d, 80mm]$
Distance between bolt and unloaded end	$a_{3,c}$	$90° \le \alpha \le 150°$	$(1+6\cdot\|sin\alpha\|)\cdot d$
		$150° \le \alpha \le 210°$	$4\cdot d$
		$210° \le \alpha \le 270°$	$(1+6\cdot\|sin\alpha\|)\cdot d$
Distance between bolt and loaded edge	$a_{4,t}$	$0° \le \alpha \le 180°$	$\max[(2+2\cdot sin\alpha)\cdot d, 3\cdot d]$
Distance between bolt and unloaded edge	$a_{4,c}$	$180° \le \alpha \le 360°$	$3\cdot d$

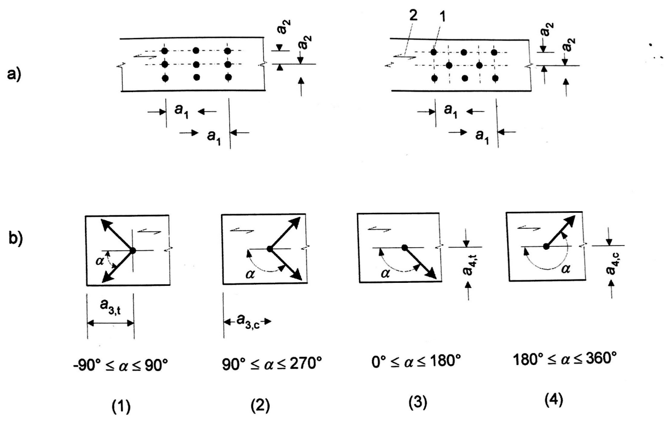


Figure 3.2: Symbols for spacing of multiple fasteners, [39].

As a_1 and a_2 always will be larger than p_1 and p_2, it is only necessary to calculate a_1 and a_2. Since a_4 is larger than e_2 and the beam and plate edges are in line, e_2 can also be neglected.

It is also advantageous to have the perpendicular edge spacing $a_{4,c}$ and $a_{4,t}$ equal to each other, avoiding further eccentric loads by making the center line of the glulam beam intersect the center of the bolt group. Therefore:

$$a_4 = \max\Big((2+2\sin\,\alpha)\cdot d,\ 3\cdot d\Big)$$

3.3 Timber Capacity

3.3.1 Johansen's Approach

Johansen [40] describes what has been later known as Johansen theory or Johansen's approach. This is a simplified theory for calculation of the plastic load-carrying capacity of dowel-type timber connections without the effect of tension force in the dowel. A dowel type connection is a connection consisting of dowel-type fasteners such as nails, bolts, dowels, and screws. K.W Johansen first proposed the theory in 1941. Today it is often known as the European Yield Model (EYM). It is described closely in Porteous and Kermani [41] and Larsen and Enjily [34].

The Johansen Theory is considering connections with both single and double shear planes consisting of timber-timber, timber-wood or steel-timber. It is only valid if the failure is ductile and not brittle such as splitting. In Eurocode 5 [39], methods have been developed to ensure that the failure is in ductile rather than brittle mode. For example, if the minimum spacings and edge or end distances prevent splitting when the connection is subjected to a lateral load.

It is assumed that the fastener acts as a laterally loaded beam, loaded by a constant contact pressure q per unit length. According to the ductile failure theory for dowel-connections, the fastener and the timber or wood-based material connected will behave as essentially rigid plastic materials. This assumption simplifies the analysis for the Johansen equation.

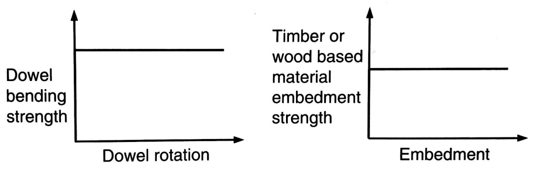


Figure 3.3: Strength/strain relationship used for dowel connections, [41].

The possible failure modes that can arise is often referred to as modes type 1,2 and 3. Mode type 1 is when failure only occurs by embedment of the connection material, and there is no yielding in the fastener. Mode type 2 is when failure is due to a combination of material embedment failure and single yield failure in the fastener. Mode type 3 is when there is a combination of material embedment failure and double yield failure in the fastener. The three different modes can arise for all the different connection types (see Figure 3.4, 3.5, 3.6 and 3.7) [41].

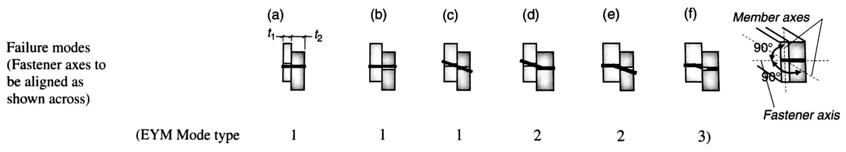


Figure 3.4: Failure modes for single shear plane for timber-timer and timber-wood based connections, [41].

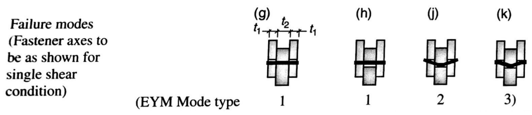

Failure modes (Fastener axes to be as shown for single shear condition)
(g)
(h)
(j)
(k)
t_1
t_2
t_1
(EYM Mode type
1
1
2
3)

Figure 3.5: Failure modes for double shear plane for timber-timer and timber-wood based connections, [41].

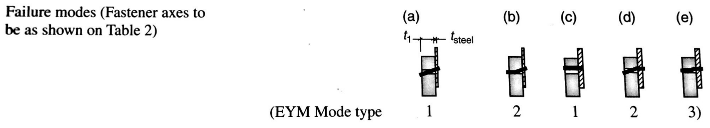


Figure 3.6: Failure modes for single shear plane for steel-timber connections, [41].

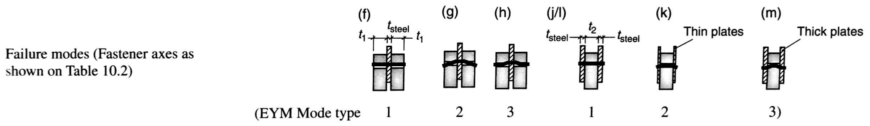


Figure 3.7: Failure modes for double shear plane for steel-timber connections, [41].

For the proposed connection, the failure modes and associated strength equations for the double shear plane steel-timber connection are used (see Figure 3.7). The equations are dependent on the geometry of the connection, the embedment strength of the timber and the bending strength of the fastener. It is also required that the fastener will not withdraw from the connection. For timber-steel connections, the capacity of the steel plate must also exceed the connection strength. If the thickness of the steel plate is less than or equal to $0.5 \cdot d$, it classifies as a thin plate. If it is thicker or equal to d, and the tolerance of the hole diameters is less than $0.1 \cdot d$ it is classified as a thick plate. For plates with a classification between thin and thick plate, the result can be found by linear interpolation between the two capacities [41].

When calculating for a steel plate as a middle member, the equations are valid for any thickness of the steel plate. The characteristic load carrying capacity per shear plane per fastener for this type of connection is the least of the three following equations [39]:

$$R_{\mathrm{v,k}} = f_{\mathrm{h},\alpha,\mathrm{k}} \cdot t_1 \cdot d \tag{3.4}$$

$$R_{\mathrm{v,k}} = f_{\mathrm{h},\alpha,\mathrm{k}} \cdot t_1 \cdot d \left(\sqrt{2 + \frac{4 \cdot M_{\mathrm{y,Rk}}}{f_{\mathrm{h,k}} \cdot d \cdot t_1^2}} - 1 \right) + T \tag{3.5}$$

$$R_{\mathrm{v,k}} = 2.3 \cdot \sqrt{M_{\mathrm{y,Rk}} \cdot f_{\mathrm{h},\alpha,\mathrm{k}} \cdot d} + T \tag{3.6}$$

Where Equation (3.4) is for failure mode f (mode type 1), Equation (3.5) is for failure mode g (mode type 2) and Equation (3.6) is for failure mode h (mode type 3) as shown in Figure 3.7. T is the rope effect (see Chapter 3.3.2) and $f_{\mathrm{h},\alpha,\mathrm{k}}$ is the characteristic embedment strength at an angle α to the grain. It is given by:

$$f_{\mathrm{h},\alpha,\mathrm{k}} = \frac{f_{\mathrm{h,0,k}}}{k_{90} \cdot \sin^2(\alpha) + \cos^2(\alpha)} \tag{3.7}$$

$f_{\mathrm{h,0,k}}$ is the characteristic embedment strength parallel to the grain. As explained in Porteous and Kermani [41] the embedment strength of timer or a wood-based product, f_{h}, is the average compressive strength of the timber or wood-based product under the action for a stiff straight dowel loaded as shown in Figure 3.8.

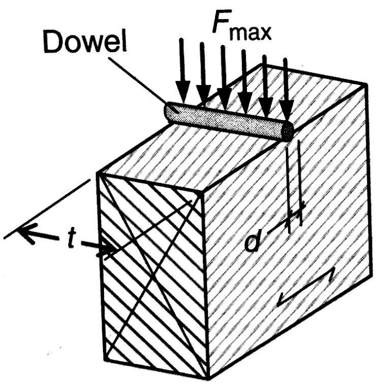


Figure 3.8: Embedment strength of a timber or wood-based material, [41].

$$f_{h,0,k} = 0.82 \cdot (1 - 0.01 \cdot d) \cdot \rho_k \tag{3.8}$$

k_{90} is a constant dependent on the type of wood.

$$k_{90} = \begin{cases} 1.35 + 0.015 \cdot d & \text{(softwood)} \\ 1.30 + 0.015 \cdot d & \text{(LVL)} \\ 0.90 + 0.015 \cdot d & \text{(hardwood)} \end{cases} \tag{3.9}$$

The values for softwood will be used, as it better matches the properties for Norwegian spruce or pine.

$M_{y,Rk}$ is the characteristic yield moment of the fastener. The original Johansen theory calculated this value as the moment at the elastic limit of the fastener, so it was derived as the product of the yield strength and the elastic modulus of the fastener. As the method developed, researchers concluded that this method gave a lower bound strength, and started to use the elasto-plastic strength. This takes into account the amount of rotation in the failure state for different kind of fasteners. It depends on the tensile strength of the fastener which includes the effect of strain hardening as well as the variation in the material strength [41]. For bolts the characteristic yield moment is:

$$M_{y,Rk} = 0.3 \cdot f_{uk} \cdot d^{2.6} \tag{3.10}$$

Where f_{uk} is the characteristic tensile strength of the bolt.

3.3.2 The Rope Effect

In Equation (3.5) and (3.6) there is a contribution T to the capacity. T is called the friction effect, or rope effect. The friction forces between the members and the withdrawal resistance have been neglected in the original Johansen theory. In EC5 the equation has been modified to include the rope effect. However, this contribution is only relevant for the failure modes that involve yielding of the fastener.

Different types of friction can occur in a connection. The friction of interest in this chapter is the type of friction that occurs when the fastener starts to yield and pulls the members together as it deforms under the lateral load. Hence this is a type of friction that will always arise in failure modes where the fastener yields. The vertical forces caused by the bending of the fastener and the friction has to be added to the Johansen equation [41].

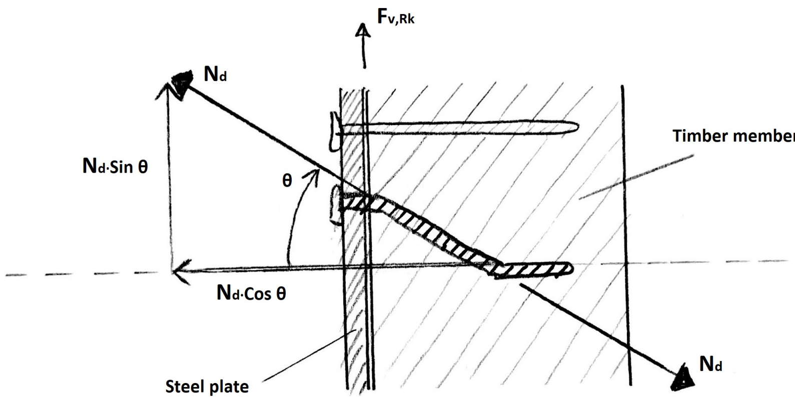


Figure 3.9: Illustration of the rope effect.

Consider a single shear connection consisting of a timber member connected to a thin steel plate by a single dowel-type fastener. This situation is shown in Figure 3.9. Under the lateral shear force, the fastener bends in the timber member and allows to rotate with an angle θ to the original position. Hence, besides being subjected to bending, the fastener will be subjected to a tension force N_d due to the withdrawal effect. N_d will have both a vertical ($N_\mathrm{d} \cdot \sin\theta$) and a horizontal ($N_\mathrm{d} \cdot \cos\theta$) component. The horizontal component will press the steel plate onto the timber member and add a vertical resistive force. If the coefficient of the friction between the steel plate and the timber member is assumed to be μ, this additional contribution will be $\mu \cdot N_\mathrm{d} \cdot \cos\theta$.

The contribution from the rope effect will then be the sum of all these vertical forces;

$N_d \cdot (\sin\theta + \mu \cdot \cos\theta)$. In EC5 the component $N_d \cdot \sin\theta$ is replaced with $F_{ax,Rk}/4$. $F_{ax,Rk}$ is the characteristic withdrawal capacity of the connection. The other component; $N_d \cdot \sin\theta$ is set to be a percentage of the Johansen contribution of the capacity, which means that the contribution from the rope effect is $F_{ax,Rk}/4$. This is limited to be maximum a given percentage of the Johansen contribution (see Table 3.3).

Table 3.3: Limiting percentages for the "Rope effect", [39].

Fastener type	Percentage
Round nails	15 %
Square nails	25 %
Other nails	50 %
Screws	100 %
Bolts	25 %
Dowels	0 %

For bolts, $F_{ax,Rk}$ is the least of the design strength of the bolt and the design capacity of the washer ([41]):

$$F_{ax,Rd,bolt} = 0.9 \cdot f_{uk} \cdot \frac{1}{\gamma_{M2}} \cdot A_{bolt} \tag{3.11}$$

$$F_{ax,Rd,washer} = 3\frac{k_{mod} \cdot f_{c,90,k}}{\gamma_{M,Connection}} \cdot \frac{\pi}{4} \cdot (d_w^2 - (d+1)^2) \cdot \tag{3.12}$$

3.3.3 Effective Characteristic Capacity for Row of Bolts

The capacity for a row of bolts placed in the direction of the fibers shall also be calculated for a reduced capacity due to the number of bolts. Eurocode 5, 8.1.2 [39] states that the capacity for the row of bolts in the direction of fibers can be calculated as:

$$F_{v,eff,Rk} = n_{eff} F_{v,Rk} \tag{3.13}$$

where $F_{v,Rk}$ is the capacity of a single bolt in the direction of fibers, while n_{eff} is the effective number of bolts. If the load acts parallel to the fibers, n_{eff} is calculated by:

$$n_{eff} = \min\begin{cases} n \\ n^{0.9}\sqrt[4]{\frac{a_1}{13d}} \end{cases} \tag{3.14}$$

$n_{eff} = n$ if the load is perpendicular to the fibers. Loads with angle can be linearly interpolated between the two results.

3.3.4 Timber Splitting Verification

Splitting is a brittle failure mode that is a result of crack formation parallel with the wood fibers due to shear forces acting on the bolt group. Eurocode 5 suggests a method of calculation to ensure that Johansen failure occurs before splitting [39].

$$F_{\text{v,Ed}} \leq F_{\text{90,Rd}} \tag{3.15}$$

$$F_{\text{v,Ed}} = \max\left(F_{\text{v,Ed,1}},\ F_{\text{v,Ed,2}}\right) \tag{3.16}$$

$$F_{\text{90,Rk}} = 14bw\sqrt{\frac{h_{\text{e}}}{\left(1-\frac{h_{\text{e}}}{h}\right)}} \tag{3.17}$$

where $w = 1$ for bolt connections. Figure 3.10 shows how splitting can be calculated for connections with a diagonal force acting upon it.

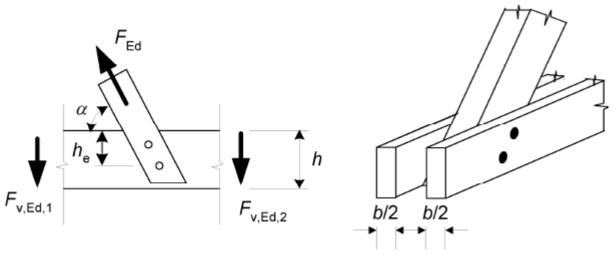


Figure 3.10: Splitting of timber, [39].

3.4 Gusset Plate Capacity

When calculating the capacity of an aluminum plate with multiple fasteners, one has to take into account several types of failures. For a plate subjected to mainly normal force, but also momentum and shear, the capacity checks that have to be done is:

- Shear Resistance for Bolt
- Bearing Resistance
- Cross-Section Capacity
- Block Tearing
- Buckling Capacity

3.4.1 Shear Resistance Bolt

As every bolt has two shear planes, the capacity becomes according to Eurocode 3, Table 3.4 [38].

$$F_{\mathrm{v,Rd}} = 2 \cdot \frac{\alpha_{\mathrm{v}} \cdot f_{\mathrm{ub}} \cdot A}{\gamma_{\mathrm{M2}}} \tag{3.18}$$

Where A is the tensile stress area of the the bolt A_{s}. α_{v} is dependent on the bolt classes. for classes 4.6, 5.6 and 8.8 $\alpha_{\mathrm{v}} = 0,6$ and for classes 4.8, 5.8, 6.8 and 10.9 $\alpha_{\mathrm{v}} = 0,5$.

It is assumed in this thesis that the bolts are steel bolts of class 8.8.

3.4.2 Bearing Resistance

The bearing resistance is calculated according to Eurocode 9, Table 8.5 [37]

$$F_{\mathrm{b,Rd}} = \frac{k_1 \cdot \alpha_{\mathrm{b}} \cdot f_{\mathrm{u}} \cdot d \cdot t}{\gamma_{\mathrm{M2}}} \tag{3.19}$$

Where α_{b} is the smallest of α_{d}, $\frac{f_{\mathrm{ub}}}{f_{\mathrm{u}}}$ or $1,0$.
α_{d} will be, in the direction of load transfer:

For end bolts

$$\alpha_{\mathrm{d}} = \frac{e_1}{3 \cdot d_0} \tag{3.20}$$

For inner bolts

$$\alpha_{\mathrm{d}} = \frac{p_1}{3 \cdot d_0} - \frac{1}{4} \tag{3.21}$$

k_1 will be, perpendicular to the direction of load transfer, the smallest of:

For edge bolts

$$2,8 \cdot \frac{e_2}{d_0} - 1,7 \quad or \quad 2,5 \tag{3.22}$$

For inner bolts

$$1,4 \cdot \frac{p_2}{d_0} - 1,7 \quad or \quad 2,5 \tag{3.23}$$

3.4.3 Cross-Section Capacity

The cross-section must have sufficient elastic or plastic cross-sectional capacity to handle the design loads. It is therefore necessary to check if the cross-section is valid for the actual forces, e.g. tension, compression, shear, bending moment and the combination. For simplicity, the plates are calculated using the cross-sectional control according to Eurocode 9, Chapter 6.2 [37] for compression, tension, bending moment and the combination of all.

3.4.4 Block Tearing

Eurocode 9, Chapter 8.5.2.2 [37] gives the design capacity for the block shear tearing resistance of a concentric loaded bolt group:

$$V_{\text{eff,1,Rd}} = \frac{f_u \cdot A_{\text{nt}}}{\gamma_{\text{M2}}} + \frac{f_y \cdot A_{\text{nv}}}{\sqrt{3} \cdot \gamma_{\text{M0}}} \tag{3.24}$$

As the yield line is perpendicular to the force direction through the entire cross-section, the formula becomes equal to Equation (3.24) which is arleady calculated.

3.4.5 Buckling Capacity

The plate are very slender. This makes it necessary to control for column buckling. The following criterion can be used as a verification, according to Eurocode 9 [37]:

$$N_{\text{b,Rd}} = \frac{\kappa \chi f_o A_{\text{eff}}}{\gamma_{\text{M1}}} \tag{3.25}$$

where χ is the reduction factor due to slenderness and defects while κ is the reduction factor due to welding of aluminum.

χ can be decided by

$$\chi = \frac{1}{\Phi + \sqrt{\Phi^2 - \overline{\lambda}^2}} \leq 1,0 \tag{3.26}$$

where

$$\Phi = 0.5\left[1 + \alpha\left(\overline{\lambda} - 0.2\right) + \overline{\lambda}^2\right] \tag{3.27}$$

α is the imperfection factor. The most conservative estimate is to set α equal to 0.32, [37].

$\overline{\lambda}$ is the relative slenderness of the plate. This can be calculates using the relationship:

$$\overline{\lambda} = \sqrt{\frac{Af_o}{N_{\text{cr}}}} = \frac{L_{\text{cr}}}{i}\frac{1}{\pi}\sqrt{\frac{f_o}{E}} = \frac{kL}{\pi}\sqrt{\frac{f_o A}{EI}} \tag{3.28}$$

It is reasonable to assume that the buckling length of the plate will be equal to the length from the outer radius to the timber edge, meaning $k = 1.0$. There could be an issue where the plate bends inside the timber slot, maybe necessitating $k = 2.0$ at worst case scenario, but for simplicity, $k = 1.0$ is assumed.

Remark: If the aluminum connection uses a production method without welds, set $\kappa = 1.0$.

3.5 Cylinder Capacity

The cylinder is the key to transfer the forces between the gusset plates. The circular shape enables the forces to travel both clockwise and counterclockwise through the cylinder wall, making it hard to find a simple way of calculating the stress in the cylinder. As a result, a collection of key failure modes were located, and corresponding simplified methods of calculation were selected.

As aluminum and steel share many properties, the method for gusset plates connected to a circular CHS-profile (described in Eurocode 3, Chapter 7.4 [38]) can be used to find the fundamental failure modes and calculation methods for the aluminum connection. Figure 3.11 shows the connection.

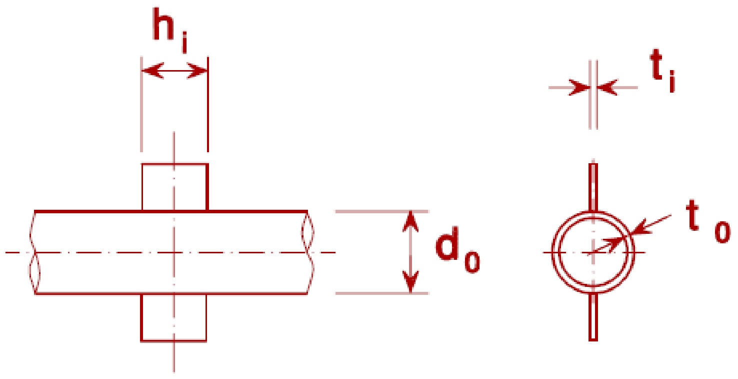


Figure 3.11: Plates connected to a CHS-profile, [38].

3.5.1 Punching Shear Failure

Thin walled cylinders, where $10 \leq d_0/t_0 \leq 50$, shall always be checked for punching shear failure. Eurocode 3, Chapter 7.4 gives formulas for plates welded to CHS-profiles

[38]:

$$\sigma_{\max} t_1 = \left(\frac{N_{\mathrm{Ed}}}{A} + \frac{M_{\mathrm{Ed}}}{W_{\mathrm{el}}}\right) \cdot t_1 \leq \frac{2 t_0 \left(f_{y0}/\sqrt{3}\right)}{\gamma_{\mathrm{M5}}} \tag{3.29}$$

Although this formula is supposed to be used for truss systems where the CHS-profiles are considerably longer than the cylinders used in this connection design, it should not make too much difference since it will not alter the shear stress path.

The choice of γ_{M5} varies considerably for different materials and production techniques. γ_{M5} is set to 1.00 for welded steel connections [38]. For cast aluminum, the yield strength is usually reduced by a safety factor of 1.10 [37]. It therefore makes sense to set $\gamma_{\mathrm{M5}} = 1.10$ for cast aluminum connections. If the connection between the gusset plate and the cylinder is created by welding wrought aluminum, the reduction in the yield strength of aluminum in the heat affected zone shall be taken into account [37].

3.5.2 Cylinder Face Failure

Eurocode 3, Table 7.3 also describes formulas for testing a thin cylinder wall for yield stress [38]:

$$N_{1,\mathrm{Rd}} = 5 k_p \mu \frac{f_{y0}}{\gamma_{\mathrm{M5}}} t_0^2 \left(1 + 0.25\eta\right) \tag{3.30}$$

$$M_{\mathrm{ip},1,\mathrm{Rd}} = h_1 \cdot N_{1,\mathrm{Rd}} \tag{3.31}$$

$$M_{\mathrm{op},1,\mathrm{Rd}} = 0 \tag{3.32}$$

so that

$$\frac{N_{\mathrm{i,Ed}}}{N_{\mathrm{i,Rd}}} + \left[\frac{M_{\mathrm{ip,i,Ed}}}{M_{\mathrm{ip,i,Rd}}}\right]^2 + \frac{|M_{\mathrm{op,i,Ed}}|}{M_{\mathrm{op,i,Rd}}} \leq 1.0 \tag{3.33}$$

η is defined as h_1/d_0. Since the cylinder does not act as a truss system chord, it is assumed that the stress axially in the cylinder $\sigma_{0,\mathrm{Ed}} = 0$, meaning that k_p can be set to 1.0. μ is a reduction factor due to the multiple gusset plates spaced around the cylinder (as described in Chapter 3.5.3).

It is not as clear whether this formula works properly for this connection, especially since the form of the equation is very simplified and unfamiliar from an analytically perspective, and cannot be traced back to a more precise equation. It is also problematic to identify the additional structural effects of the connection in comparison to a truss system due to the same reasons. It is therefore essential to compare the utilization from this method with the FEA results from Chapter 5.4.

3.5.3 Effect from Spatial XX-Connection Type

The XX-connection is considered in the Eurocode 3 [38] as a spatial connection where four bars are connected to the cylinder in the same cross-sectional plane. Figure 3.12 shows how the different bars act on the cylinder.

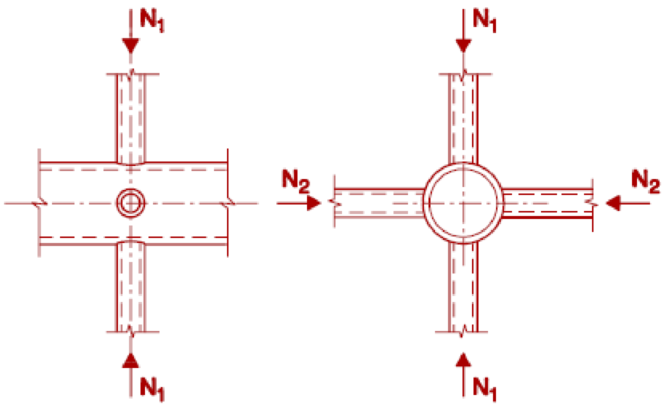


Figure 3.12: Spatial XX-connection [38].

Eurocode 3 [38] states that for XX-connection types, there should be used a capacity reduction factor equal to

$$\mu = 1 + 0.33 N_{2,\mathrm{Ed}} / N_{1,\mathrm{Ed}} \tag{3.34}$$

where $|N_{2,\mathrm{Ed}}| \leq |N_{1,\mathrm{Ed}}|$.

As many of the connections can be considered XX-connections, the following method seems reliable for calculating the combined effect if it can be considered conservative enough. This is mainly since the worst case scenario for the cylinder for any number of bars, is when $N_{2,\mathrm{Ed}}$ and $N_{1,\mathrm{Ed}}$ act correspondingly as tension and compression with the same magnitude. Therefore, it can be assumed that such a situation can be used as a conservative estimate for connections with more gusset plates than four, applying the maximum tensile and compressive force as $N_{1,\mathrm{Ed}}$ and $N_{2,\mathrm{Ed}}$.

The angular difference between the beams should not matter, as it will not exceed the conservative estimate. As long as the angular difference is larger than 30°, the calculations will still be valid, as stated in Eurocode 3, Chapter 7.1.2(3) [38].

Remark: It can also be considered conservative to calculate with the reduction factor $\mu = 0.67$ constantly for every connection.

3.5.4 About the Accuracy of Calculations

As previously stated, according to Eurocode 3, Chapter 7.4, the simplified approach is only valid for thin-walled cylinders where $10 \leq d_0/t_0 \leq 50$. This is not the case for all of the connections because of a production criterion where the thickness of the cylinder wall was set to be minimum 40 % of the thickness of the gusset plates. This can, in some cases, lead to cylinders with $d_0/t_0 < 10$. According to Eurocode 3, Chapter 7.4.1, this makes it necessary to calculate all of the failure modes in Eurocode 3, Chapter 7.2.2 [38]. Such specimens shall be re-examined through calculation of more failure modes or more advanced structural analysis, like FEA.

The choice of safety factors and calculation methods is partly out of scope of this thesis – since this thesis primarily focuses on the possibility of parameterization of connection design, and less on related calculation theory. It was therefore favorable to reduce the time used in finding accurate formulas for the static behavior, and instead spend time solving the practical and structural challenges with parametric models. Therefore the material strength and cylinder calculations used should not be considered anything else than a rough estimate. The FEA in Chapter 5.4 does, however, look more into the accuracy of these calculation methods and can potentially validate this approach.

Chapter 4

Parametric Design of Grid Shell Connections

4.1 Software for Digital Workflow

4.1.1 Rhino 3D

Rhino 3D is a CAD-software licensed by Robert McNeel Associates for 3D-modelling and rendering. Particularly important is the ability of Rhino to export 3D-models in sat-file format, which will later be used for structural analysis in this theses.

4.1.2 Grasshopper

Grasshopper is a visual programming tool developed to create parametric models. It comes as a plugin for Rhino, enabling Grasshopper to use the geometry library of Rhino when operating on 3D-objects.

Programming in Grasshopper consists of connecting small objects, called *components*, together in more complicated structures. The components takes an input, processes the data, and generates output. E.g., *Circle CNR* is a component that takes a center point, a normal vector, and a radius as input, and creates a circle-object as output. Figure 4.1 shows this component. Three parameters (A Point, Vector and a value of 1.00) are connected to the component at the corresponding connection on the left side, and a circle parameter extracts the circle object from the component on the right side. By connecting multiple components, the user can construct more complex processes with multiple different components. Grasshopper offers a vast library of built-in components but also enables the user to download custom toolkits from other developers, or even to program custom components in C# or Visual Basic.

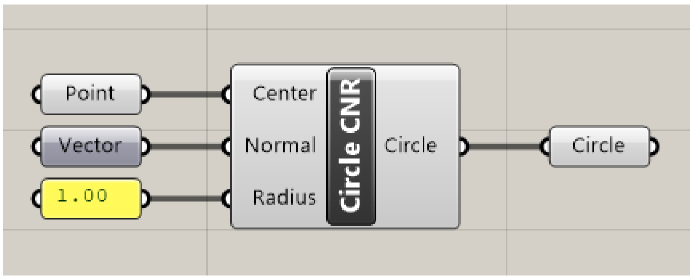


Figure 4.1: The *Circle CNR*-component working in the Grasshopper interface.

Data Trees

A key concept in Grasshopper is the *Data Tree*-object. The data tree is a hierarchical data structure where data is stored in *branches*. Figure 4.2 visualizes a data tree with four levels of branches. The orange points with red lines symbolize the data stored in the different branches.

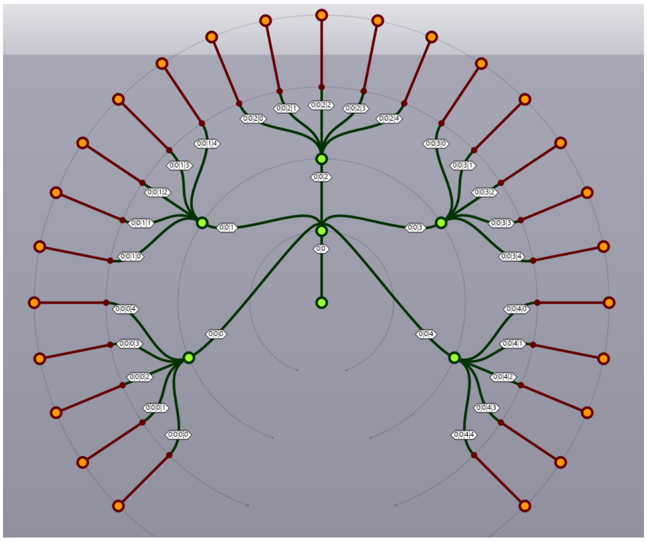


Figure 4.2: Visualizing the hierarchical structure of a data tree.

4.1.3 Karamba

Karamba is a plugin developed for use in Grasshopper. It implements FEA for beam and shell models with different components for different types of analysis, e.g. linear static analysis, nonlinear static analysis, linear buckling analysis or dynamic modal analysis.

It also comes with components to read beam forces and displacements. This makes it easy to find the forces acting on the connections and find the nodal displacement, later used in Chapter 5.4 to apply the loads to the bolts in the FEA.

4.2 Scope of the Parametric Model

The development of a parametric model for grid shell connections is motivated by the need for structural analysis and customization of the grid shell connections. The main tasks are to propose a custom connection design for every grid shell connection, verify the capability of handling the design load, and generate the corresponding 3D-models for analysis and manufacturing.

The strategy for creating a fully automated connection generator is heavily dependent on the choices made in Chapter 2, where some fundamental choices for the design were made to simplify the parametric model. The choice of designing the connection of a cylinder connected with flat gusset plates makes it simple to generate the necessary geometries and assemble the 3D-model, as shown in this chapter. Figure 4.3 shows the proposed parametric model. The model uses a combination built-in components and custom C# components.

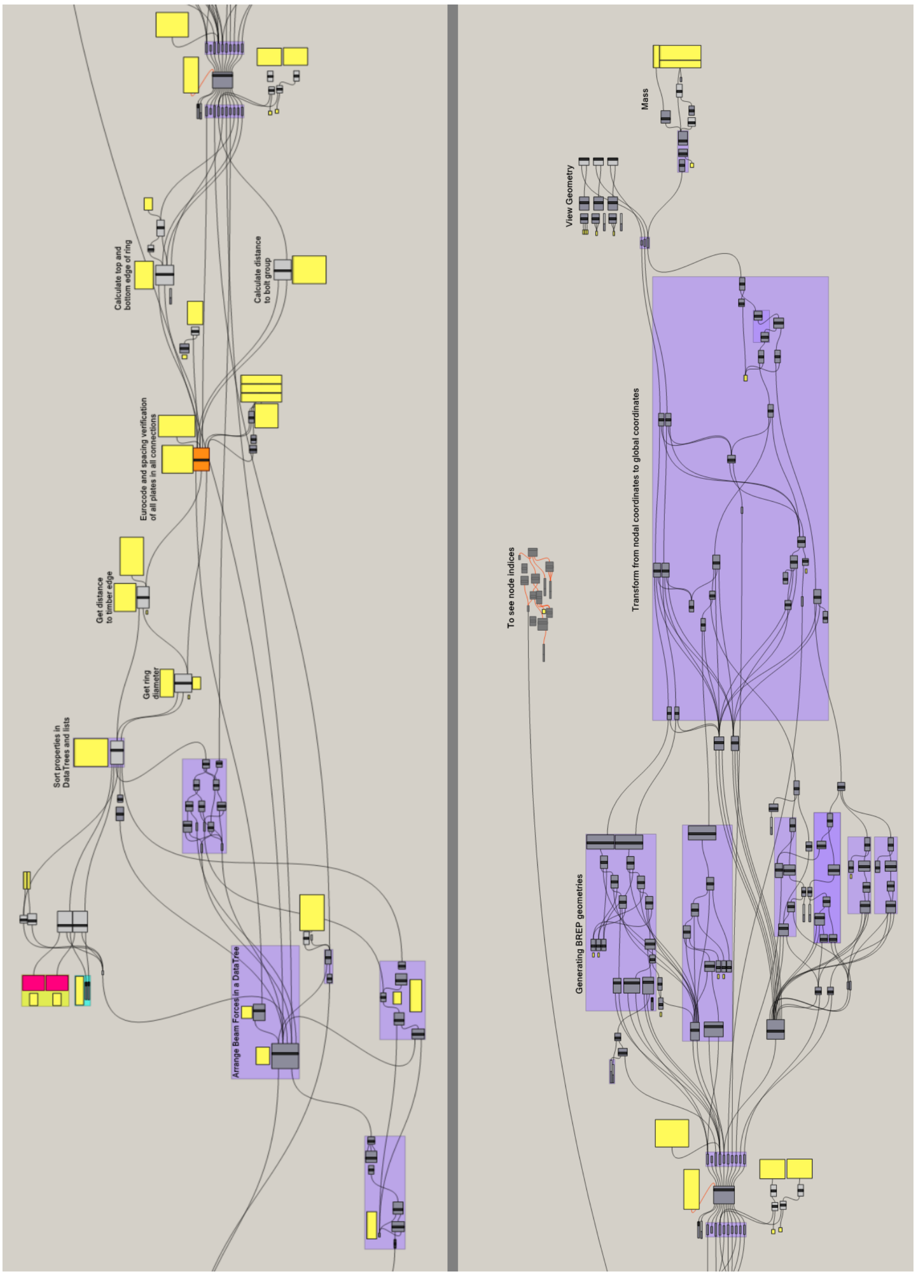

Calculate top and bottom edge of ring
Calculate distance to bolt group
Eurocode and spacing verification of all plates in all connections
Get distance to timber edge
Get ring diameter
Sort properties in DataTrees and lists
Arrange Beam Forces in a DataTree
Mass
View Geometry
Transform from nodal coordinates to global coordinates
To see node indices
Generating BREP geometries

Figure 4.3: Screenshot of the entire parametric model. The left side continues on the right side.

4.3 Coordinate Systems

The connection needs multiple different coordinate systems when handling the geometry in different situations. The following coordinate systems were used with corresponding transformation rules.

4.3.1 Global Cartesian Coordinates

The global coordinate system corresponds to the defined coordinate system of the structure. All the connections will finally be generated in the global coordinates x, y and z. Since this coordinate system is standard for all connections, all the connections and beams can be plotted in the same space – simplifying the visualization of the entire structure to ensure that none of the connections or beams intersect. Figure 4.4 shows a gridshell where all connections and beams are generated in the global coordinate system.

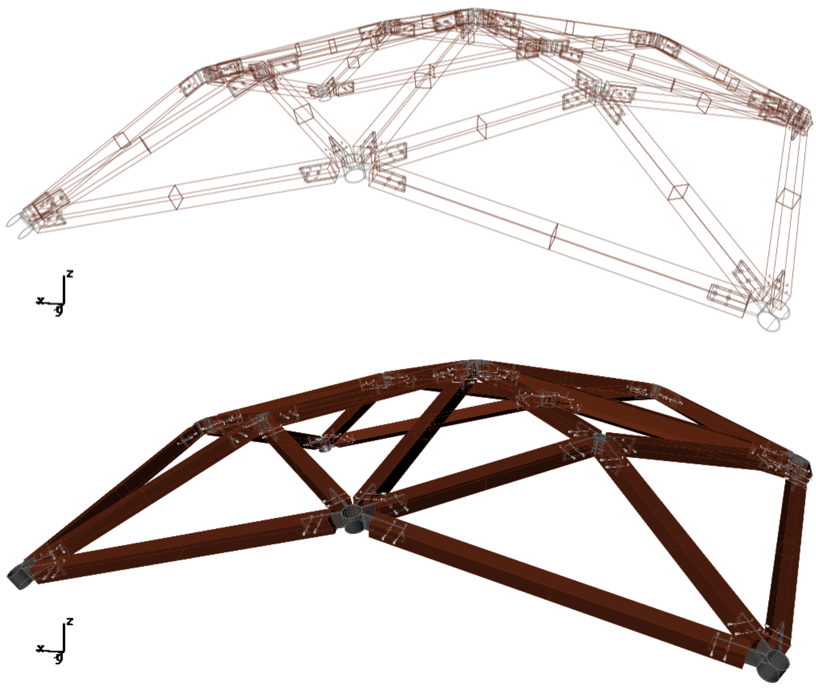


Figure 4.4: Example of gridshell where all connections and beams generated together in the global coordinate system.

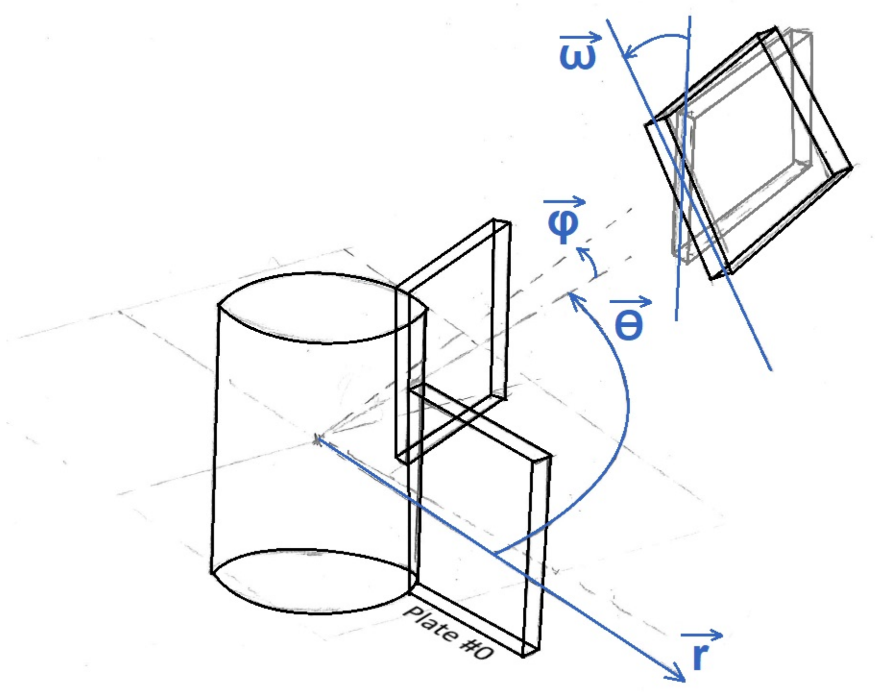


Figure 4.5: Local spherical coordinates for the connection.

4.3.2 Local Spherical Coordinates for the Node

The nodal coordinates are modified spherical coordinates, describing how the different plates are connected to the cylinder. Using spherical coordinates enables us to easier find the relevant angles for the gusset plates and simplifies the spacing verification considerably. This is further described in Chapter 4.4.2. The set of coordinates used are θ, ϕ, r and ω. Figure 4.5 shows how the different coordinates are defined, including ω which describes the rotation of the plates.

The transformation between nodal coordinates and global coordinates is made by first defining a plane with a normal perpendicular to the shell surface, and projecting the axial vector of the beam with index 0 onto that plane. Then, let the projection define the r-axis, and let the preceding plane be the horizontal plane in the spherical coordinate system.

In certain situations, it might be necessary to transform the local spherical coordinates to local cartesian coordinates, mainly since most geometry is generated in cartesian coordinates. Then, the normal vector to the shell surface is the local z-axis, the *r*-vector is the local x-axis, and the origin is unchanged.

4.3.3 Local Cartesian Coordinates for the Plate

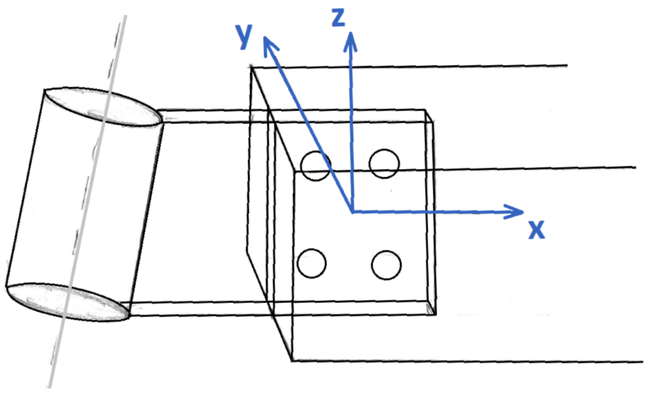


Figure 4.6: Local cartesian coordinates for the plate.

Figure 4.6 shows the definition of the coordinates. The coordinate system is defined with the origin placed both in the center of the bolt group and the center of the cross-section of the beam.

The plate coordinates are primarily used in verification of the bolt connection between the aluminum and timber. The directions of the axes are the same as the local coordinates of the beams, which avoids transformations of the beam forces. Using the method described in Chapter 3.1, it is easier to calculate the moment of inertia for the bolts when the origin is positioned in the mass center of the bolt group.

When rotating the plates according to the cylinder, a reliable method of transformation is needed. Aircraft principal axes (pitch, roll, and yaw) was chosen for transformation from plate coordinates to local node coordinates. Figure 4.7 shows the different rotational motions. All rotations use the local origin as the center of rotation, making all the forces from the beams act in one point. Rotation is always relative to the original plate coordinate system, implying that the coordinate system does not change orientation during transformation. The procedure of rotating a plate correctly relative to the cylinder is shown in Figure 4.8, according to the angles in Figure 4.5. The *r*-vector can be used to translate the object according to the new origin.

The reasoning behind the use of aircraft principal axes rests primarily on the simplicity of the system and its ability to describe any 3D-configuration in an understandable language, making communication and debugging of the program more convenient.

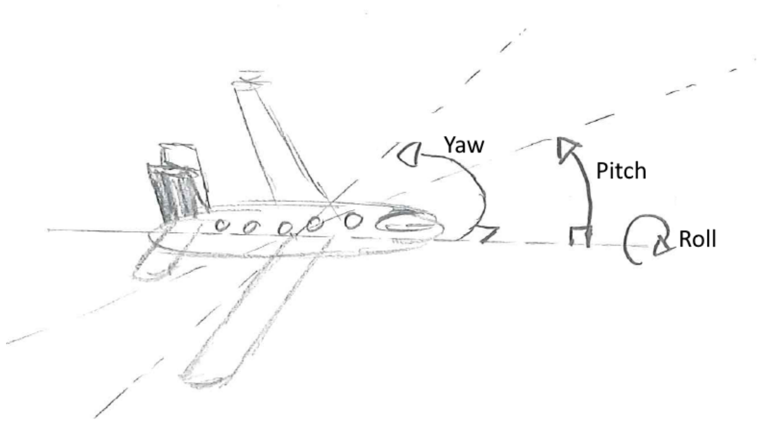


Figure 4.7: The transformation between plate coordinates and node coordinates uses the same notation as aircraft principal axes.

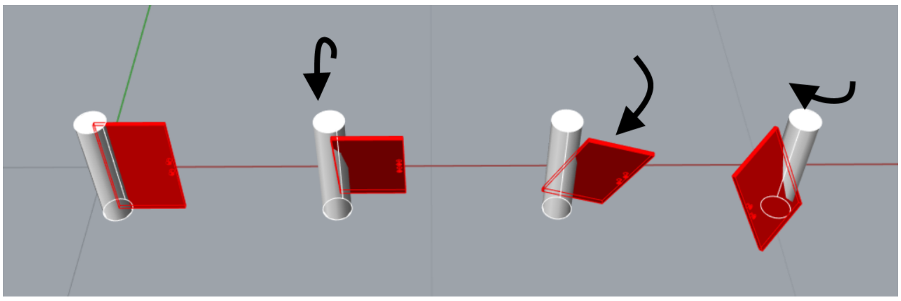
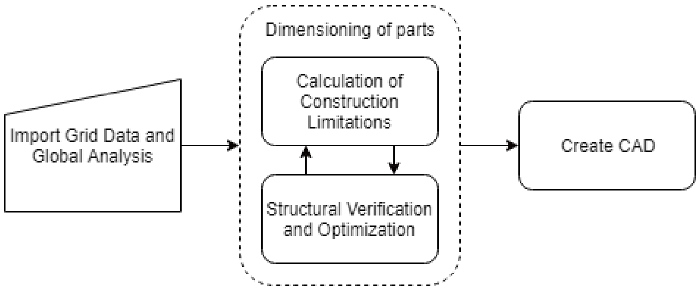


Figure 4.8: The three rotational motions orienting the plate correctly according to the node. From left to right; unrotated, roll, pitch and yaw.

4.4 Processes and Sub-Processes

To simplify and organize the concept of the parametric model, the model is divided into smaller parts. Each part is responsible for solving a smaller portion of the problem and deliver a valid solution with all necessary data.

Figure 4.9 visualizes the partition of the process and creates a very simple chart of how the program works. The program is partitioned into four parts; *input processing, spatial limitations, structural verification and optimization* and *generating 3D-models.*

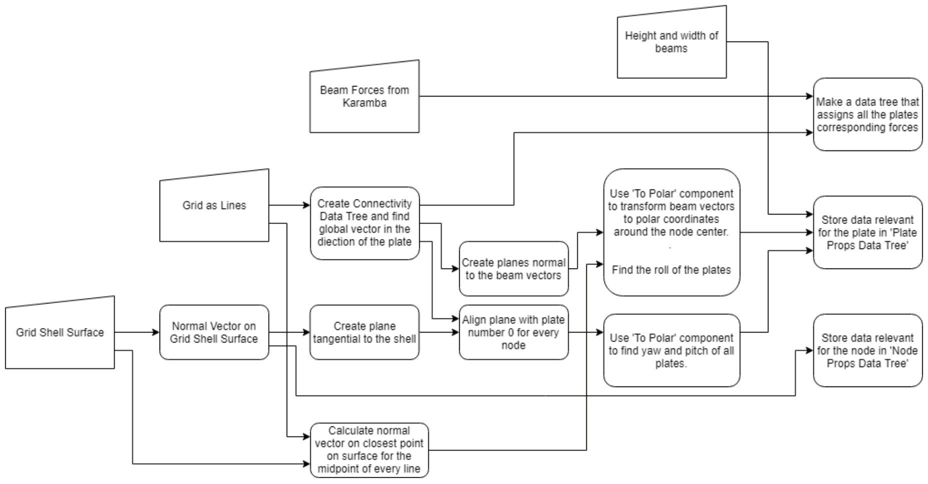


Dimensioning of parts
Import Grid Data and Global Analysis
Calculation of Construction Limitations
Structural Verification and Optimization
Create CAD

Figure 4.9: The general flow of information in the parametric model.

4.4.1 Input Process

The main tasks of the input processing are to collect the information about the gridshell, and establish a data tree structure to store the information more conveniently. The data tree also has to collect information about the general orientation of the plates, number of attached plates and the dimensions of the connected beams. The input processing takes four data inputs; the surface of the structural grid, a list of lines representing the position of the beams in the global coordinate system, the height and width of the beams and the forces corresponding to the beams. The flow chart in Figure 4.10 displays the general tasks and information flow.



Height and width of beams
Beam Forces from Karamba
Make a data tree that assigns all the plates corresponding forces
Grid as Lines
Create Connectivity Data Tree and find global vector in the diection of the plate
Use 'To Polar' component to transform beam vectors to polar coordinates around the node center.
Find the roll of the plates
Store data relevant for the plate in 'Plate Props Data Tree'
Create planes normal to the beam vectors
Grid Shell Surface
Normal Vector on Grid Shell Surface
Create plane tangential to the shell
Align plane with plate number 0 for every node
Use 'To Polar' component to find yaw and pitch of all plates.
Store data relevant for the node in 'Node Props Data Tree'
Calculate normal vector on closest point on surface for the midpoint of every line

Figure 4.10: The general outlines of the input processing.

By constructing data trees where the first layer of branches corresponds to the nodes and the second layer corresponds to the plates, it becomes easier to fetch information about a plate when needed. Every node is assigned an index value and the plates connected to the node also get separate indexing. E.g. a system of nodes might have the following plate indices: {0,0}, {0,1}, {1,0}, {1,1}, {1,2} and {2,0}. This system has three nodes. The first node is connected to two plates, the second node is connected to three plates while the last one only got one plate.

The Input Processing creates *Plate Properties Data Tree* and *Node Properties Data Tree* to store lists of properties for every plate and node. The information in Table 4.1 and Table 4.2 is attached to the list in the data tree during the input processing. The different components in the model take the data trees as input, append more information to the lists and outputs the updated data tree.

Table 4.1: Information in Plate Properties Data Tree from input processing.

Index	Information
0	Width of timber beams
1	Height of timber beams
2	Roll of plates
3	Yaw of plates
4	Pitch of plates

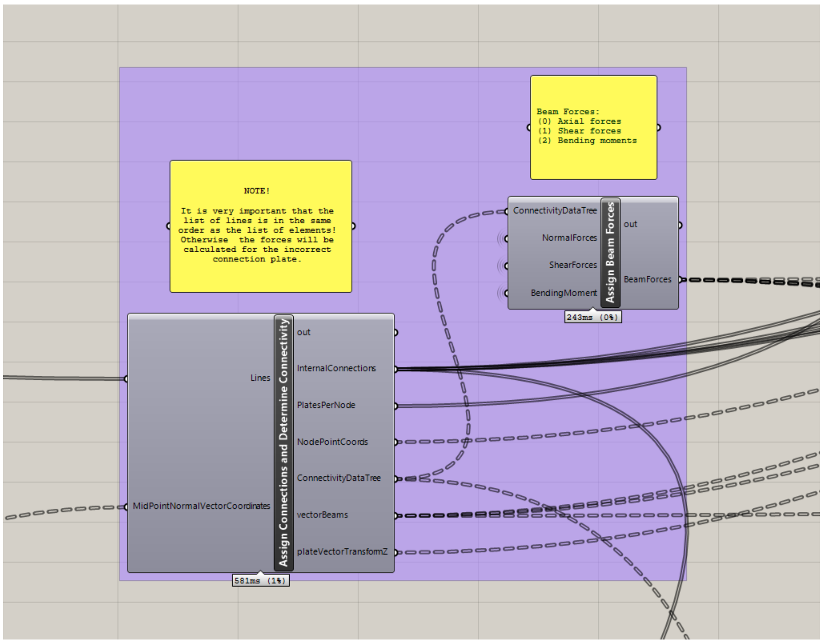
Table 4.2: Information in Node Properties Data Tree from input processing.

Index	Information
0	Node center coordinate in global x-direction
1	Node center coordinate in global y-direction
2	Node center coordinate in global z-direction
3	Normal vector on surface in node, x-component
4	Normal vector on surface in node, y-component
5	Normal vector on surface in node, z-component

By looping over all the lines and locating their start points and end points, the model creates a connectivity data tree where every branch corresponds to a gusset plate. Every branch stores two values; the line index and whether the node is the start point or end point of the line. Figure 4.11 shows the module where the connectivity data tree is generated from the lines. The connectivity data tree is primarily used for two things; assigning beam forces to the correct plate, and calculating the orientation of the gusset plates in local spherical node coordinates.

The assignment of beam forces is shown in Figure 4.11. A programmed component called *Assign Beam Forces* is used to create a data tree with three levels of branches;

node, plate, and load case. Then it assigns beam forces according to the connectivity data tree.



```
Beam Forces:
(0) Axial forces
(1) Shear forces
(2) Bending moments
```

```
NOTE!

It is very important that the
list of lines is in the same
order as the list of elements!
Otherwise  the forces will be
calculated for the incorrect
connection plate.
```

Figure 4.11: Components creating the connectivity data tree, beam forces and some other input processing.

The parametric model only fetches the axial forces N_x, vertical shear forces V_z and in-plane bending moments M_y. This is due to the assumption that the nodes are hinged for in-plane rotation. The model is although flexible enough to allow appending other beam forces, e.g. V_z, M_z or M_x, at a later point. This might be necessary if the connection design changes to a stiffer type. By adding more input nodes to the component, and appending this information in the C# component, more force data will automatically be fed to the structural verification code where it can easily be fetched.

Calculating the orientation of the plates is primarily done using built-in components from the standard library of Grasshopper. The assembly of the components is shown in Figure 4.14. The component group defines a plane tangential to the shell surface (see Figure 4.12) and planes normal to the beams (see Figure 4.13). The planes are used together with the normal and axial vectors of the beams to find the rotation angles ω, ϕ and θ for all plates. The normal to the beams are found using the component group in Figure 4.15.

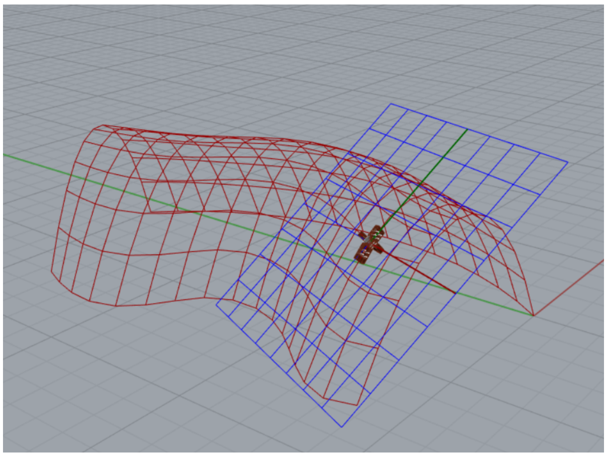


Figure 4.12: A plane tangential to the gridshell surface at the node.

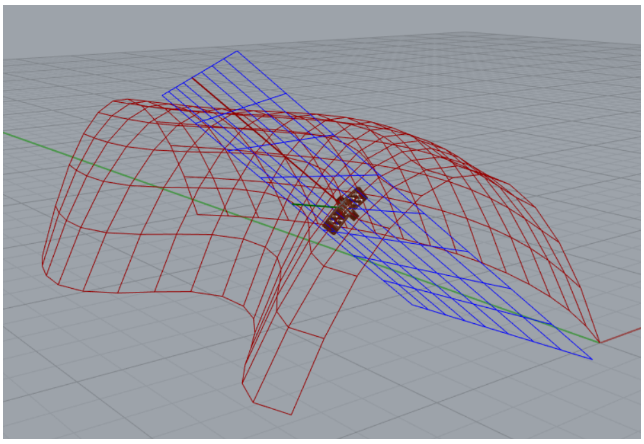
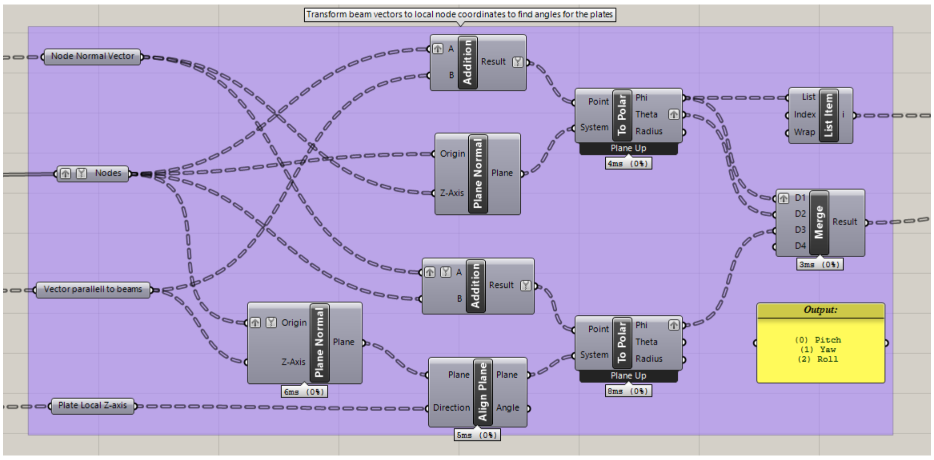


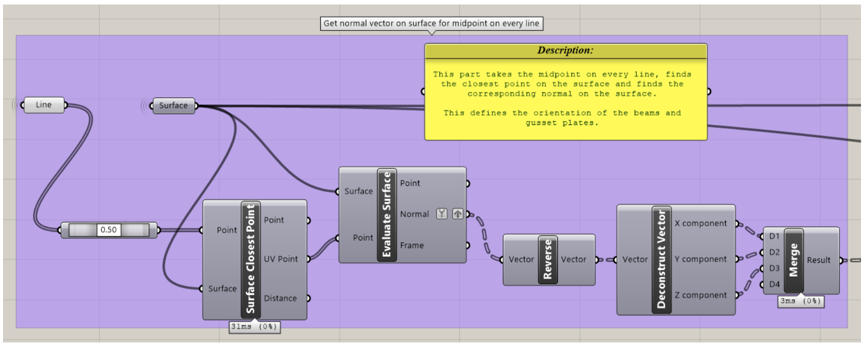
Figure 4.13: A plane normal to the beam vector and aligned with the surface normal vector at the middle point of the beam.



Output:

(0) Pitch
(1) Yaw
(2) Roll

Figure 4.14: Components finding the necessary rotations to transform the plates from plate coordinates to node coordinates.



Description:

This part takes the midpoint on every line, finds the closest point on the surface and finds the corresponding normal on the surface.

This defines the orientation of the beams and gusset plates.

Figure 4.15: A code block finding the normal vector on the surface for the middle point of every line.

4.4.2 Spatial Limitations

After the input has been stored and transformed into more suitable data, the necessary dimensions and configuration of the connection have to be calculated. For convenience, the processes concerning the dimensions of the connections have been split into two sub-processes; *spatial limitations* and *structural verification and optimization*. The first process covers mainly spacing and dimensioning to ensure the practicality and possibility of assembly, e.g., avoiding overlapping parts or incomplete connection between components. The latter investigates the structural behavior of the connections and choose, e.g., necessary gusset plate thickness, bolt size and number of bolts to ensure structural stability during all design loads.

It is important to note that the two sub-processes depend heavily on each other. The necessary cylinder diameter is, for instance, dependant on the thickness of the gusset plates, while the thickness of the gusset plates is calculated from elastic buckling, which depends on the cylinder diameter. This interaction might indicate that two sub-processes should be collectively calculated in the optimization process. Splitting them does, however, lead to particular advantages organizationally, primarily about the complexity of the optimization process. Merging them would lead to a messy and unorganized code block.

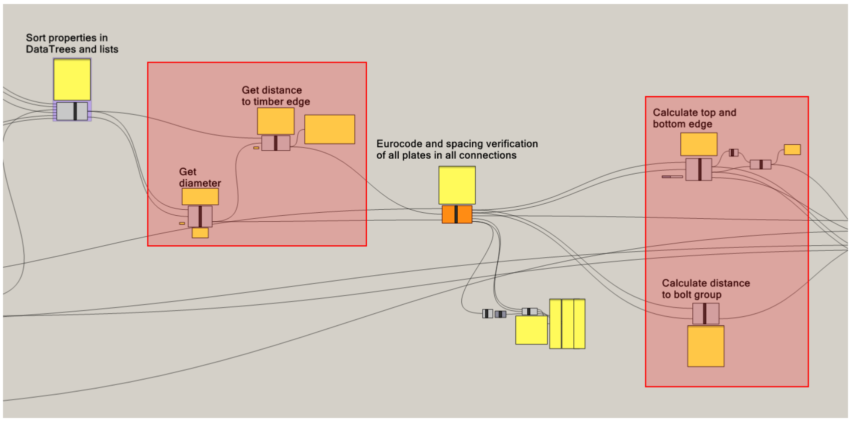

Sort properties in DataTrees and lists
Get distance to timber edge
Get diameter
Eurocode and spacing verification of all plates in all connections
Calculate top and bottom edge
Calculate distance to bolt group

Figure 4.16: The Constructional Limitations Calculations are computed in the components surrounded by the red rectangles.

Figure 4.16 shows the four problems that the spatial limitations sub-process has to solve. The *Get Cylinder Diameter*-process works out a minimum outer cylinder diameter to avoid the gusset plates from overlapping. The *Get Distance To Timber*-component calculates how far the timber beams have to be placed from the node origin to prevent intersecting beams. The *Calculate Top and Bottom Edge of Cylinder*-component calculate how tall the top and bottom edge of the cylinder has to be to ensure that all of the plates are connected to the cylinder surface. The *Calculate Distance To Bolt Group*-component calculates the necessary distance to the center of the bolt group from the node center based on the required timber spacing and the required spacing of bolt. All the information is appended to Plate Properties Data Tree and Node Properties Data Tree as shown in Table 4.3 and 4.4. Note that the indices leap, due to some of the components running after the structural verification and optimization component.

Table 4.3: Information appended to Plate Properties Data Tree from Construction Limitations.

Index	Information
5	Necessary distance to timber edge [m]
19	Distance to Bolt Group Center [m]

Table 4.4: Information appended to Node Properties Data Tree from Construction Limitations.

Index	Information
6	Cylinder diameter [m]
7	Cylinder thickness [mm]
8	Top Edge of Cylinder [m]
9	Bottom Edge of Cylinder [m]

Space Requirements for Beams

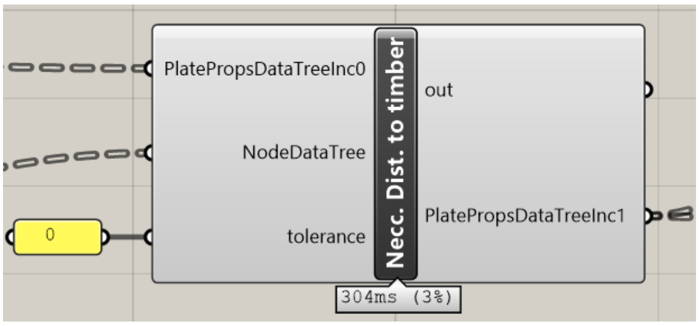


Figure 4.17: The component for calculating space requirements for beams.

When connecting the beams to the gusset plates, there are multiple practical challenges to solve. It is, for instance, essential that the beams do not intersect or collide with each other and that the beams are spaced in such a configuration that it is easy to insert bolts and tighten the nuts.

The latter issue has been decided to be out of scope. This due to the decision in the case study to use connections with four plates, which means that there will always be one plate with an adjacent plate more than 90 degrees to it. The result is that there will always be at least one way of assembling the connections.

Regarding the first issue, by dividing the volume around the node into sections corresponding to each of the beams, it is possible to constrain the beams to stay inside the section. A convenient way of defining the sections is by spherical wedges, as in

Figure 4.18. Figure 4.19 shows the concept from a planar view. The following set of equations were found through the derivation in Appendix C.1

$$l_1 = \frac{w_2 + w_1 \cdot \cos\theta}{\sin\theta} \tag{4.1}$$

$$l_2 = \frac{w_1 + w_2 \cdot \cos\theta}{\sin\theta} \tag{4.2}$$

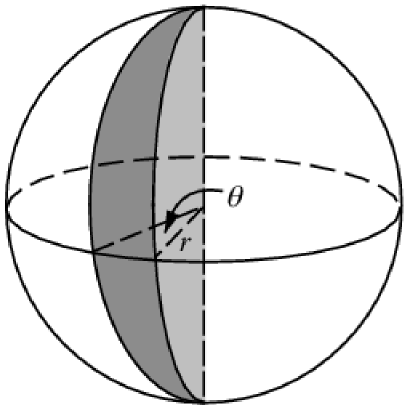


Figure 4.18: The geometry of a spherical wedge, [42].

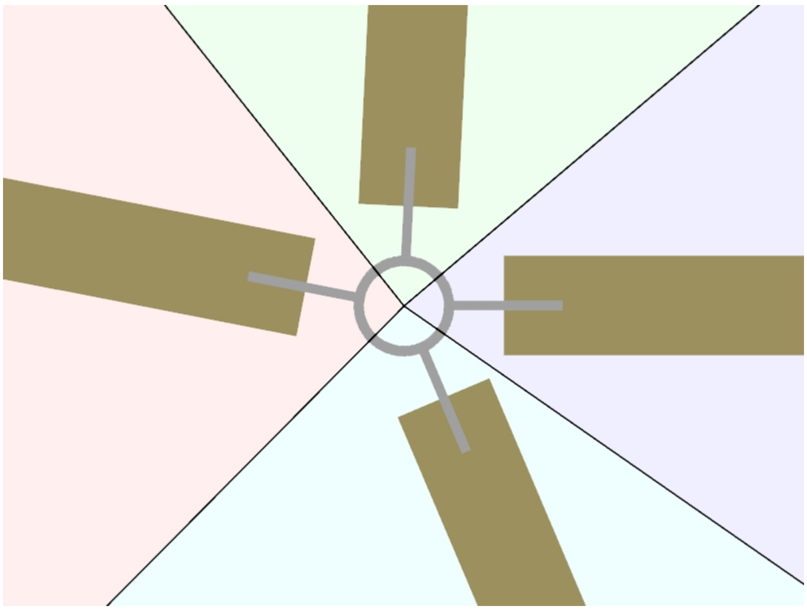


Figure 4.19: Planar view of the spherical wedges defining the allowed space for each beam.

Equation C.5 and C.6 describe the necessary distance from the nodal coordinate's z-axis to the closest edge of the beams. All adjacent beams have to be checked to verify that the length is sufficient, something that necessitates sorting the yaw angles to identify adjacency.

When pitching the plates, the timber edges get pulled closer to the cylinder. To counteract this effect, it is necessary to lengthen the plates relative to the pitch angle. The effect calculated in Appendix C.4, gives us the following formula:

$$l_{1,\text{edit}} = l_1 \sec\phi + h/2 \cdot \tan\phi \tag{4.3}$$

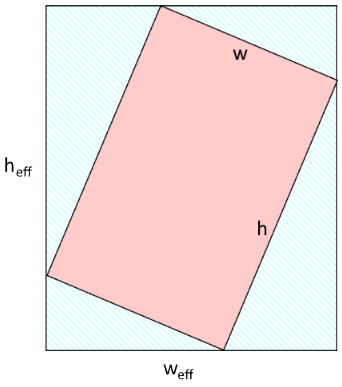


Figure 4.20: Effective cross section of a rolled rectangular cross section.

The roll of the plate will also create issues when it comes to spacing. The corners of the cross-section might occupy the adjacent space. A simple way of solving this problem is to define an effective rectangular cross-section where the rectangle is oriented orthogonal to the nodal coordinate system. This is not a solid cross-section, but rather a rectangular space to constrain the real cross-section. If the section fits in this effective section, it will also not have problems with intersecting other beams. Figure 4.20 shows how a rolled rectangular cross-section (e.g., a glulam beam) can be fitted optimally in the effective rectangle. This means that w_{eff} and h_{eff} can be calculated using:

$$w_{\text{eff}} = |w\cos\omega| + |h\sin\omega| \tag{4.4}$$

$$h_{\text{eff}} = |h\cos\omega| + |w\sin\omega| \tag{4.5}$$

Figure 4.21 displays a flowchart of the process of calculations. The corresponding script can be found in Appendix B.2.

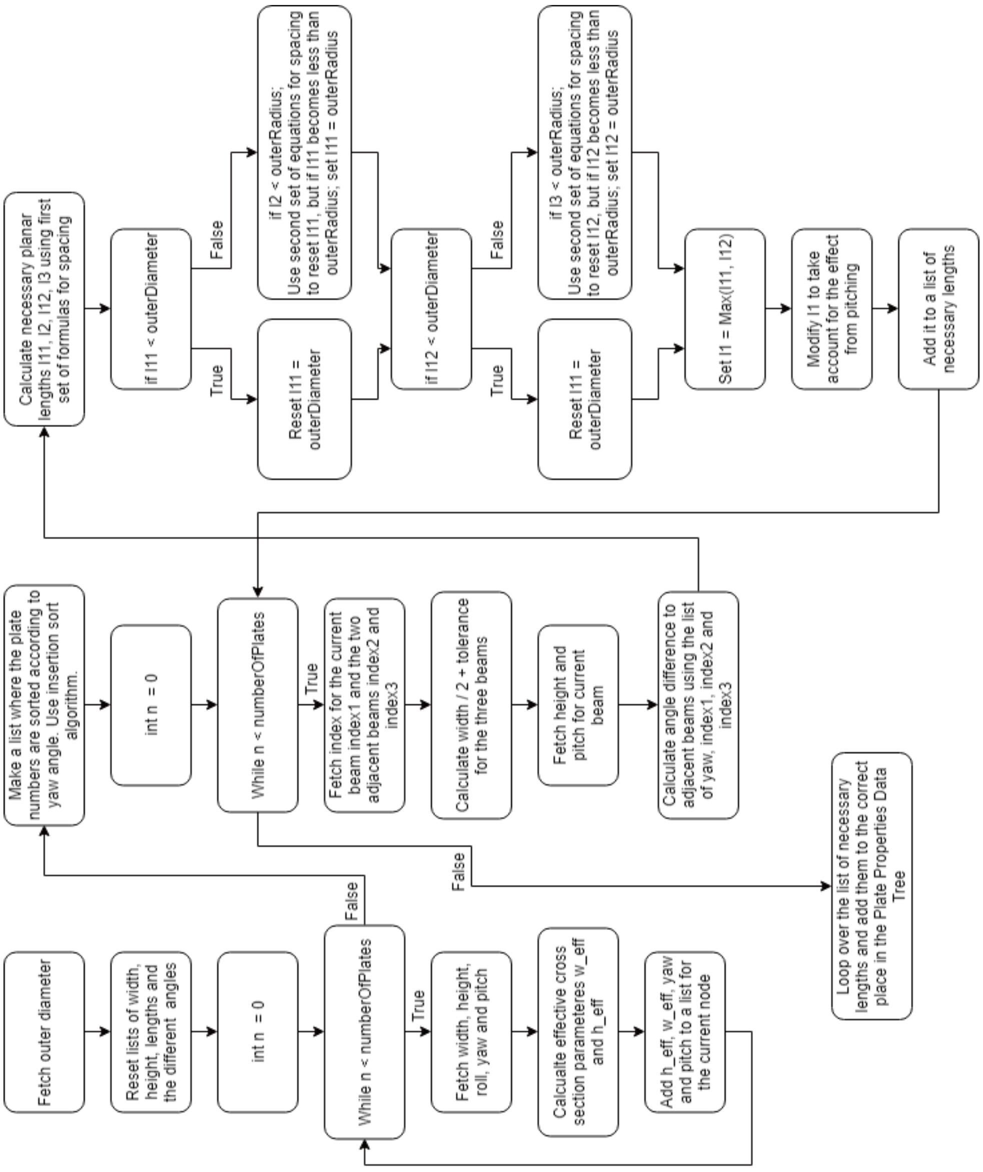

Fetch outer diameter
Reset lists of width, height, lengths and the different angles
int n = 0
While n < numberOfPlates
True
Fetch width, height, roll, yaw and pitch
Calcualte effective cross section parameteres w_eff and h_eff
Add h_eff, w_eff, yaw and pitch to a list for the current node
False
Make a list where the plate numbers are sorted according to yaw angle. Use insertion sort algorithm.
int n = 0
While n < numberOfPlates
True
Fetch index for the current beam index1 and the two adjacent beams index2 and index3
Calculate width / 2 + tolerance for the three beams
Fetch height and pitch for current beam
Calculate angle difference to adjacent beams using the list of yaw, index1, index2 and index3
False
Loop over the list of necessary lengths and add them to the correct place in the Plate Properties Data Tree
Calculate necessary planar lengths l11, l2, l12, l3 using first set of formulas for spacing
if l11 < outerDiameter
True
Reset l11 = outerDiameter
False
if l2 < outerRadius; Use second set of equations for spacing to reset l11, but if l11 becomes less than outerRadius; set l11 = outerRadius
if l12 < outerDiameter
True
Reset l11 = outerDiameter
False
if l3 < outerRadius; Use second set of equations for spacing to reset l12, but if l12 becomes less than outerRadius; set l12 = outerRadius
Set l1 = Max(l11, l12)
Modify l1 to take account for the effect from pitching
Add it to a list of necessary lengths

Figure 4.21: The algorithm for spacing the beams.

Minimum Cylinder Diameter

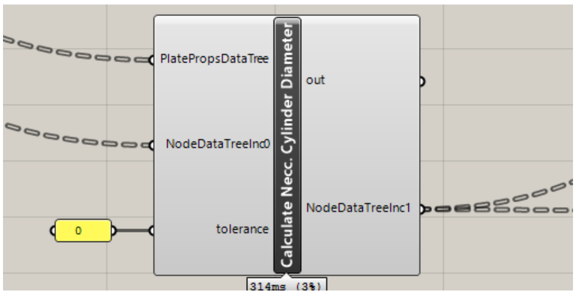


Figure 4.22: The component for calculating minimum cylinder diameter.

The attachment of the plates to the cylinder must happen in such a way that none of the plates are in contact with each other. This ensures that the forces go through the cylinder, and does not find any spurious paths between the plates. Although the gusset plates are very thin, many of the plates are rolled slightly which increases the effective width of the plates. In this case, the minimum cylinder diameter might be controlled by the top or bottom parts of the rolled plates and whether they intersect with adjacent beams.

The strategy used is similar to the strategy for finding the distance to the timber edge, namely the quadrilateral strategy used in Appendix C.1. The space around the node is divided into spherical wedges and distributed between the gusset plates. Now the cylinder diameter is the same for all the plates, while the distance from the cylinder to the timber edge is individual. This means that calculating the minimum cylinder diameter involves calculating a diameter for all of the plates and then choosing the maximum value. The diameter of the cylinder has to be chosen such that the plates do not intersect, which means that the radius is the line to the intersection point, between the plate edges. This makes some changes in the equations (as shown in Appendix C.2). The set of equations can, for instance, be reduced to one single equation:

$$r = \csc\theta \sqrt{w_1^2 + w_2^2 + 2w_1 w_2 \cos\theta} \tag{4.6}$$

where w_1 and w_2 are the effective section width for the plates, while θ is the difference in yaw angle between the plates.

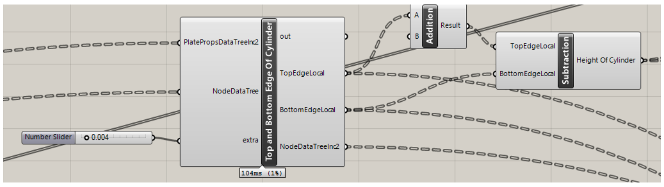


Figure 4.23: The component for calculating top and bottom height of the cylinder.

Calculation of the Top and Bottom Height of the Cylinder

The height of the cylinder is primarily decided by the footprint of the plates on the outer surface of the cylinder. By looping over all the plates connected to the node and finding the highest and lowest points of the footprint, the top and bottom of the cylinder can easily be calculated. Figure 4.23 shows the component used in the calculations. The footprint effect is derived in Appendix C.3 and scripted in Appendix B.4.

Calculation of the Distance to the Bolt Group

The component finds the distance between the origin in plate coordinates and the origin in node coordinates, using the equation:

$$D_{\text{bolt group}} = D_{\text{timber edge}} + a_3 + a_1 \cdot \left(n_{\text{bolts,x}} - 1\right)/2 \tag{4.7}$$

where $D_{\text{timber edge}}$ is the distance to the timber edge l, calculated in Chapter 4.4.2. The component is shown in Figure 4.24 and the script can be found in Appendix B.5.

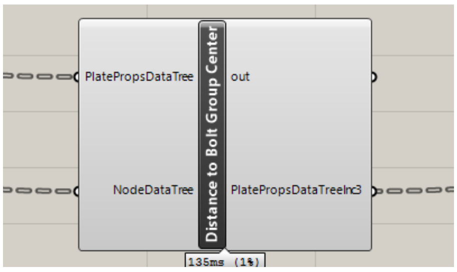


Figure 4.24: The component for calculating distance to the center of the bolt group in node coordinates.

4.4.3 Structural Verification and Optimization

The structural verification and optimization are processed in one component. The component runs an incremental optimization strategy with multiple nested loops increasing one parameter at a time until it reaches a valid solution. Upon the reach of a valid solution the current configuration properties are appended to the plate properties, node properties and bolt properties data trees as described in Table 4.5, 4.6 and 4.7.

Table 4.5: Information appended to Plate Properties Data Tree from structural verification and optimization. The spacing parameters are according to Chapter 3.

Index	Information
6	Number of Bolts in x-direction
7	Number of Bolts in z-direction
8	Thickness Gusset Plate [mm]
9	Boltsize [mm]
10	Utilization of Plate
11	Spacing Parameter, a1 [mm]
12	Spacing Parameter, a2 [mm]
13	Spacing Parameter, a3 [mm]
14	Spacing Parameter, a4 [mm]
15	Spacing Parameter, e1 [mm]
16	Spacing Parameter, e2 [mm]
17	Spacing Parameter, p1 [mm]
18	Spacing Parameter, p2 [mm]

Table 4.6: Information appended to Node Properties Data Tree from structural verification and optimization.

Index	Information
7	Thickness of cylinder [mm]

Table 4.7: Information appended to Bolt Properties Data Tree from structural verification and optimization.

Index	Information
0	x-coordinate of bolt in plate coord. system [m]
1	z-coordinate of bolt in plate coord. system [m]

The structural verification is split into different code blocks in the script according to the failure mode. Table 4.8 lists all the different failure modes with corresponding failure mode index and a reference to the chapter covering the calculation methods used in the script. The methods are displayed in the flowchart in Figure 4.25 and the corresponding script can be found in Appendix B.1.

Table 4.8: The different failure modes for the plates as output from the structural verification and optimization component.

Notation	Failure Mode	Ref:
A1	Johansen Theory: Capacity of Individual Bolts	3.3.1
A2	Johansen Theory: Effective Capacity for Row of Bolts	3.3.3
B1X	Bearing Resistance Bolt Group x-dir.	3.4.2
B1Z	Bearing Resistance Bolt Group z-dir.	3.4.2
B2	Cut-Off Capacity Bolts	3.4.1
C	Timber Splitting	3.3.4
D1	Gusset Plate Tension	3.4.3
D2	Gusset Plate Compression	3.4.3
D3	Gusset Plate Shear	3.4.3
D4	Gusset Plate Bending Moment, Axial and Shear Forces	3.4.3
D5	Gusset Plate Buckling	3.4.5
E	Cylinder: Punching Shear or Cylinder Face Failure	3.5

Incremental Improvements Optimization Approach

The incremental optimization approach is based on the concept of starting with the minimum configuration (e.g. 1x2 M12 bolts or 12 mm gusset plates), checking if the configuration is sufficient and if not, improving one parameter at a time until the configuration is sufficient in regards to the chosen verification. The method builds on the concept of *brute force algorithms* – where the algorithm tests every possible configuration, one at a time. The space to be searched can be considered a 5-dimensional space since the algorithm only varies five different parameters.

The approach can be seen as a hierarchic approach to optimization. The different properties are layered in different loops, prioritizing some parameters over others. In general, choosing prioritization for parameters should be based on the economic impact, as the cost of a project is an important success factor. In this case, changing the position of the bolt has close to none economical impact while increasing the number of bolts might have a significant impact. Therefore it is advantageous to prioritize bolt placement ahead of the number of bolts.

The prioritization of parameters can lead to cases where the optimized solution does not converge to the best solution, particularly if a lower prioritized parameter increment can lead to an economically better solution. It is also important to notice that setting high priority for parameters with little need for bigger dimensions might make the process run considerably faster.

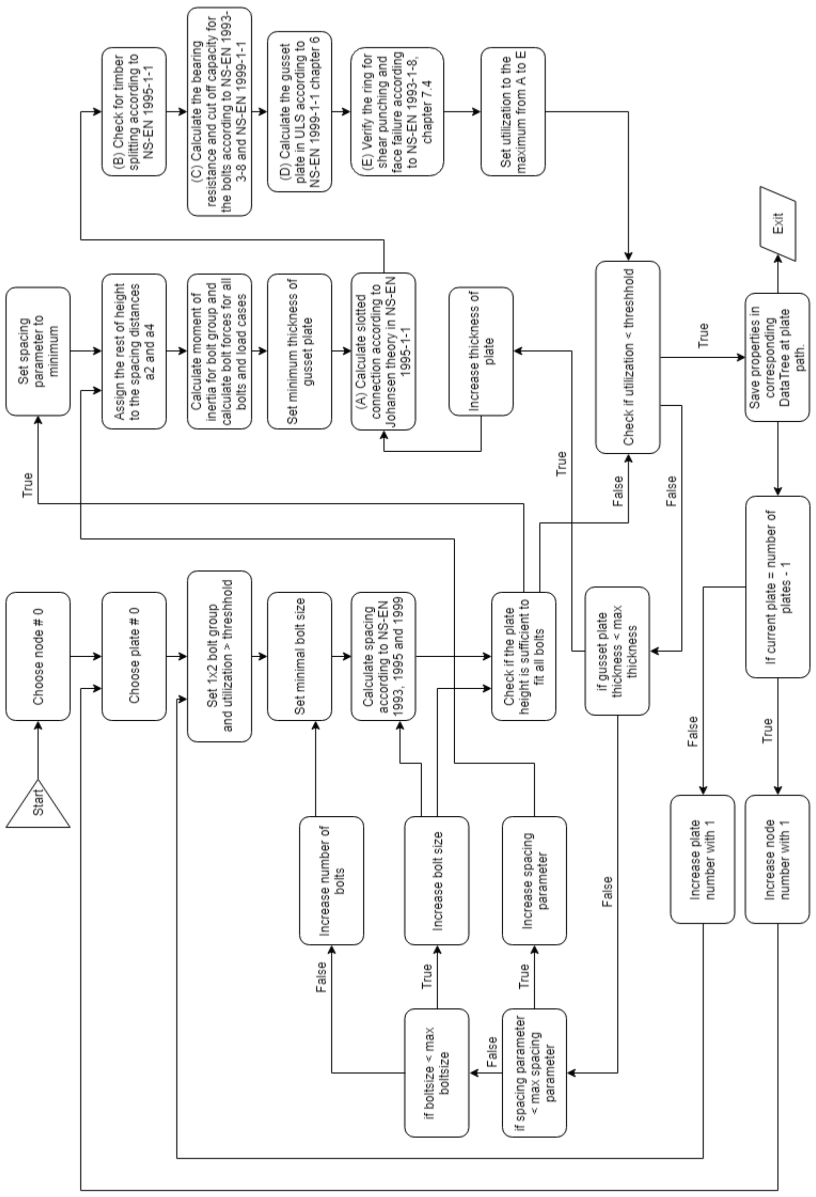

Start
Choose node # 0
Choose plate # 0
Set 1x2 bolt group and utilization > threshold
Set minimal bolt size
Calculate spacing according to NS-EN 1993, 1995 and 1999
Check if the plate height is sufficient to fit all bolts
if gusset plate thickness < max thickness
If current plate = number of plates - 1
Set spacing parameter to minimum
Assign the rest of height to the spacing distances a2 and a4
Calculate moment of inertia for bolt group and calculate bolt forces for all bolts and load cases
Set minimum thickness of gusset plate
(A) Calculate slotted connection according to Johansen theory in NS-EN 1995-1-1
Increase thickness of plate
Check if utilization < threshold
Save properties in corresponding DataTree at plate path.
Exit
(B) Check for timber splitting according to NS-EN 1995-1-1
(C) Calculate the bearing resistance and cut off capacity for the bolts according to NS-EN 1993-3-8 and NS-EN 1999-1-1
(D) Calculate the gusset plate in ULS according to NS-EN 1999-1-1 chapter 6
(E) Verify the ring for shear punching and face failure according to NS-EN 1993-1-8, chapter 7.4
Set utilization to the maximum from A to E
Increase number of bolts
Increase bolt size
Increase spacing parameter
if boltsize < max boltsize
if spacing parameter < max spacing parameter
Increase plate number with 1
Increase node number with 1
True
False

Figure 4.25: The process of optimizing the connection.

Table 4.9 shows how the different parameters are prioritized. Notice that the cylinder thickness has top priority. This is because it has a minimal impact on the rest of the model. By quickly searching the 1-dimensional space for cylinder thickness, the optimization process will not have to check the much larger 4-dimensional space of solutions that are invalid.

Table 4.9: Hierarchy of parameters.

Priority	Parameter
High	Cylinder Thickness
	Bolt Spacing Vertically
	Bolt Size
	Thickness of Gusset Plate
Low	Number of Bolts

Genetic Algorithm Optimization Approach

The genetic algorithm was also examined as a potential optimization method. The method was rejected due to slow convergence and efficiency, but for completeness, it deserves a brief review.

The genetic algorithm is potentially the most general and open approach to optimization in comparison to the incremental improvements approach. Applying the explanation proposed by Gen and Cheng [43] in the context of this thesis, the genetic algorithm can be described as an approach based on creating a population of individual connections, with random parameters. The different individuals are given a fitness score based on specifications, in this case, weight and utilization. A new generation of individuals is created based on the best solutions from the previous generation. The new population is generated by mixing the parameters of the previous generation. Repeating this process over many generations will, in theory, converge towards an optimal design.

The algorithm did however not produce results as efficiently as wanted. It was often more efficient to alter the parameters than letting the algorithm do it manually. There are possibly many reasons why the algorithm did not work as efficiently as wanted, and suggesting reasons is out of scope for this thesis. The algorithm did have some advantages regarding weight optimizing, something the incremental improvements approach does not allow – although going from simple to complex design often lead to light connection solutions.

4.4.4 3D-Models

The 3D-models were created of four different types of parts: the center cylinder, the gusset plates, the bolts and the beam end. The process of generating the 3D-models consists of generating each of the components in local plate coordinates, rotating the geometry to the correct configuration according to spherical node coordinates, moving the geometry to the correct location according to the global coordinates, removing excess volume and merging the geometry to solid geometries. Figure 4.26 shows the outlines of the 3D-processing.

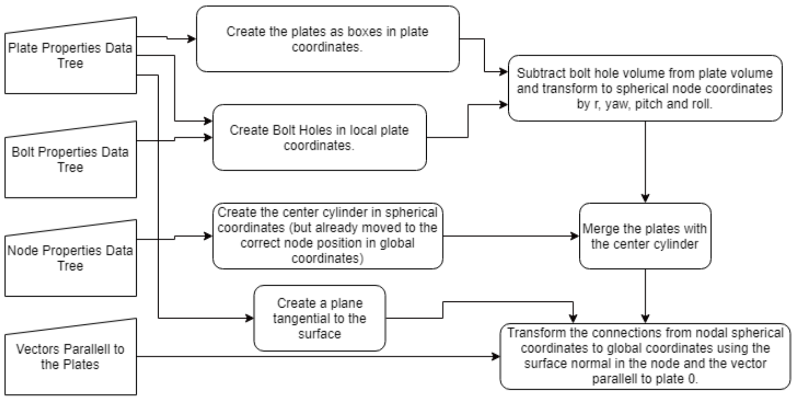


Plate Properties Data Tree
Bolt Properties Data Tree
Node Properties Data Tree
Vectors Parallell to the Plates
Create the plates as boxes in plate coordinates.
Create Bolt Holes in local plate coordinates.
Create the center cylinder in spherical coordinates (but already moved to the correct node position in global coordinates)
Create a plane tangential to the surface
Subtract bolt hole volume from plate volume and transform to spherical node coordinates by r, yaw, pitch and roll.
Merge the plates with the center cylinder
Transform the connections from nodal spherical coordinates to global coordinates using the surface normal in the node and the vector parallell to plate 0.

Figure 4.26: The process of generating 3D-geometries.

Plates

The plates are generated as box-objects in custom C#-components. The boxes are defined from two points; close to the origin, and a point distanced the necessary length of the plate from the origin. It is important to note that the plate shall not intersect with the nodal origin, due to problems trying to merge boundary representations with intersecting faces or edges. If the boundaries intersect, the components might return an error.

Center Cylinder

The approach for creating the cylinder consists of making a massive solid cylinder with outer radius and merging it with the gusset plates. The excess volume from the geometry is removed by creating a massive cylinder with the inner radius and subtracting it from the connection geometry. This also removes excess plate volume.

Bolt Holes

The bolt holes are generated as cylinder-objects defined with radius according to necessary bolt holes and tolerance – this means the bolt holes should be 2 mm wider than the bolt size.

Bolts and Timber Ends

For completeness of the 3D-model and visualizing the spacing of beams, the bolts and timber ends are generated. The bolts are created as solid cylinders as a simplified visualization of their positioning. It is important to note that the bolts will have a smaller diameter than the bolt holes due to tolerance requirements. Therefore, it is not possible to use the same geometry to create bolts and subtract bolt holes from timber ends and gusset plates – they have to be generated individually. The timber edge uses much of the same algorithm as the algorithm creating the plates. The only differences are the coordinates where the beams are made in plate coordinates and the width of the beams.

4.5 Parametric Model Verification

To verify the parametric nature of the model, assume a simple example with one connection. The node is defined in the global origin, and with n number of cantilever beams connected to it.

The materials used in the verification are nickel-aluminum-copper alloy connections (as in Appendix D), steel bolts of material quality 8.8 and GL32c glulam timber beams. If not specified, the cross-section is 220 x 115 mm. All tolerance criterion is set to 0 to test the algorithm to its most extreme.

4.5.1 Constructional Challenges

Figure 4.27 shows how the parametric model reacts to a planar node with 2, 3, 4, 5, 6 and 16 plates connected to it. It is clear that there are no problems with the spacing of the beams or any overlapping plates when increasing the number of beams. It is also clear that the spacing algorithm for the beams works well when increasing the number of beams since no beams intersect each other. For the case with two beams, it is clear that the distance to the timber edge equals the outer radius of the cylinder. This shows that the minimum criteria are activated upon possible intersection between the beam and the cylinder.

It is although worth to notice that some of the fictive bolts intersect for the connection with 16 beams. Since we decided to let this issue be out of scope, the model will not notice this. It is although easy to add a tolerance (as shown in Chapter 4.4.2) when spacing the beams, to take the bolts into account.

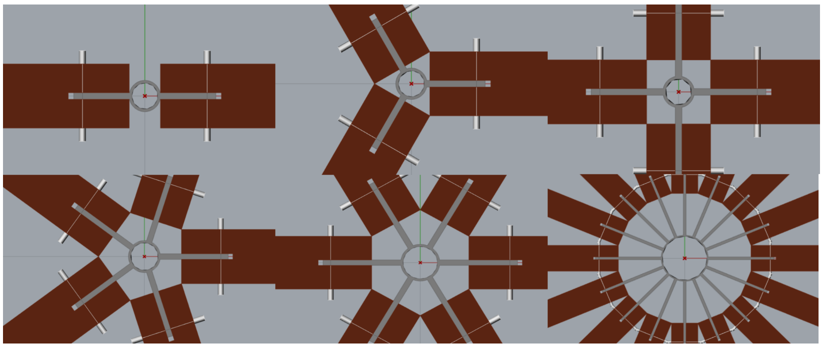


Figure 4.27: Validating different number of beams connected to the node.

To test the model's reaction to pitch, one of the cantilever beams is pitched a bit upwards. Figure 4.28 shows how the top edge of the cylinder is raised when the footprint from the gusset plate shifts. By pitching the cantilever even more, a slight gap appears, as shown in Figure 4.29. This is due to the definition of the plates, as from the center of the cylinder and rotated around the same point. This is a weakness of the model and method of creating the geometry, rather than the calculations. By also applying a roll to the plate, the diameter increases and the gap disappears, as seen in Figure 4.30 and 4.31. It can be derived that the gap might occurs when the angle ϕ becomes larger than $\arctan\left(d/h \cdot \cos\phi\right)$.

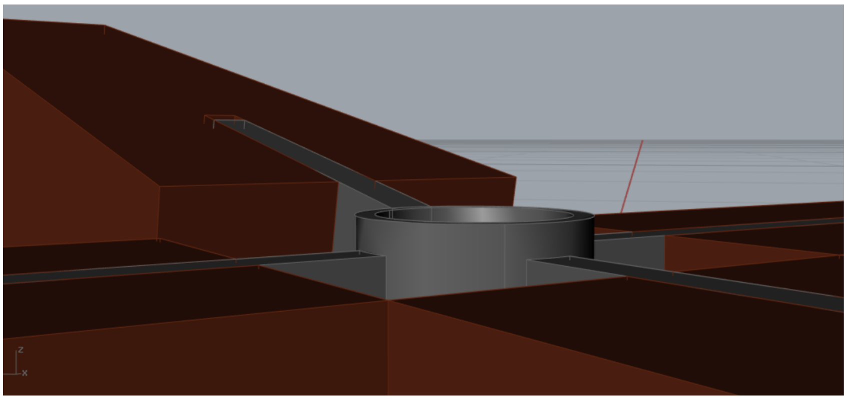


Figure 4.28: By pitching one of the beams, the top edge of the cylinder raised to react to the shifted contact surface.

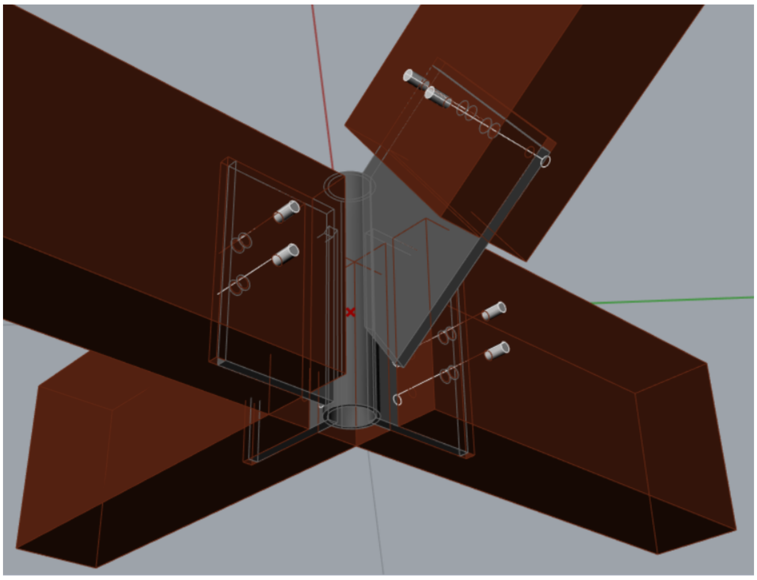


Figure 4.29: If the pitch compared to the diameter of the cylinder becomes too high, there might be a problem with generating the connections and the plates does not get sufficient contact with the cylinder.

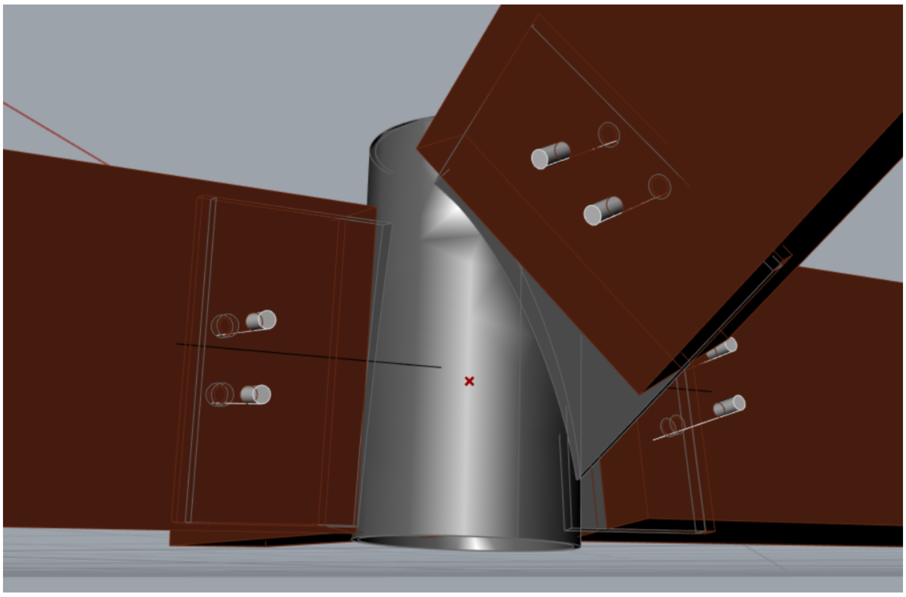


Figure 4.30: When the plate is rolled, the diameter increases and the problem in Figure 4.29 do not appear for low pitch angles.

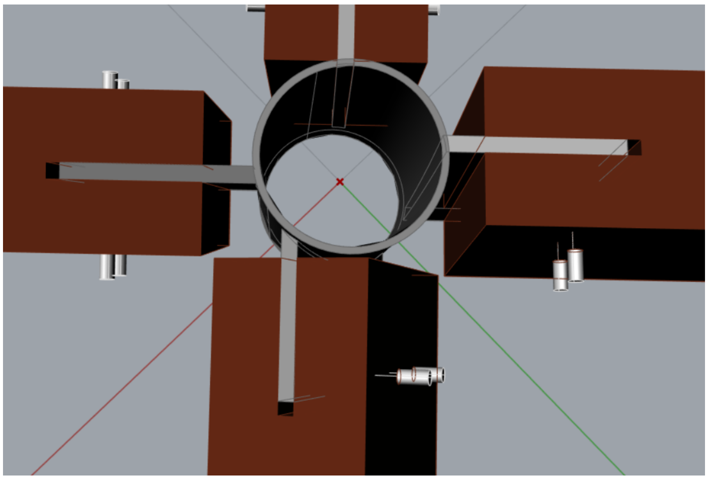


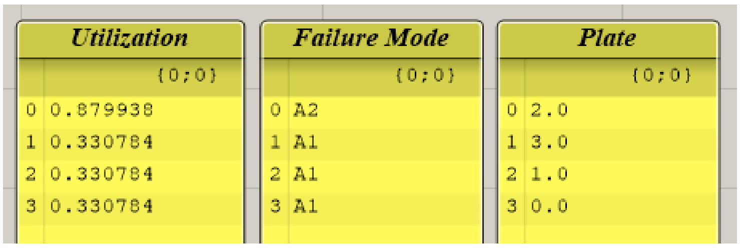
Figure 4.31: By rolling one of the plates, the cylinder diameter increases to avoid plate overlapping.

4.5.2 Structural Stability

To verify that the parametric model adapts to increased loads, a test with increasing compression loads was carried out. Figure 4.33 shows the effect on the planar node as a compressive force applied on plate 2 vary from 10 kN to 40 kN.

From 0 to 10 kN, there was no effect on the connections, and the minimal configuration was sufficient. For 20 kN to 35 kN, the bolt size, thickness of the plate, thickness of the cylinder and bolt spacing increases, as predicted.

For 40 kN, the bolt group makes a leap in the number of bolts from 2 to 6, skipping bolt groups with 4 bolts. The reason for this is the reduced effect of having multiple bolts in a row parallel to the fibers in the timber. This effect is described in Chapter 3.3.1. Figure 4.32 shows how the failure mode has changed from A1 to A2, according to the notation in Table 4.8.



```
Utilization       Failure Mode      Plate
     {0;0}             {0;0}             {0;0}
0 0.879938        0 A2              0 2.0
1 0.330784        1 A1              1 3.0
2 0.330784        2 A1              2 1.0
3 0.330784        3 A1              3 0.0
```

Figure 4.32: Failure modes and utilization for 40 kN compression on plate 2.

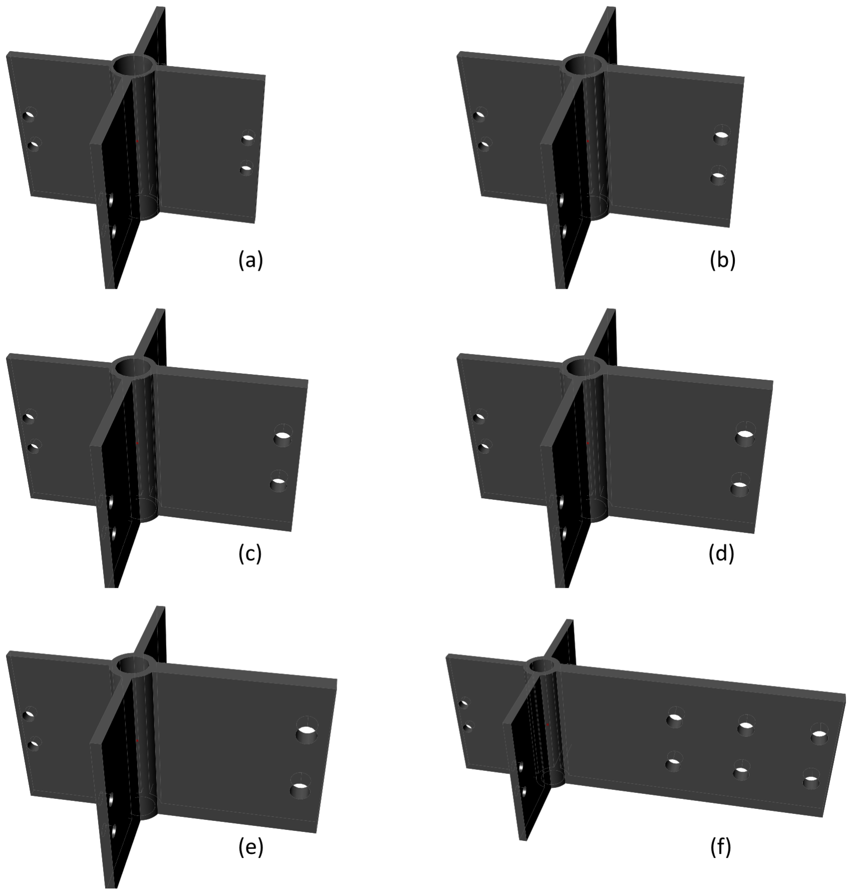


Figure 4.33: Planar node with compression force applied on the right plate: (a) 10 kN, (b) 20 kN, (c) 25 kN, (d) 30 kN, (e) 35 kN and (f) 40 kN.

To get connections with 2x2 bolt groups, the connections should instead be checked for a combination of shear forces and bending moment, since this will induce minimal bolt forces in the direction of fibers. Figure 4.34 shows that the application of a vertical shear force of 3 kN and a bending moment of 1.697 kNm creates a plate with 4 bolts.

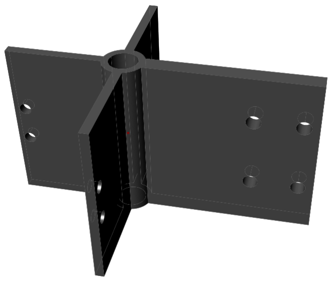


Figure 4.34: Planar node with shear force of 3 kN and bending moment of 1.697 kNm.

4.5.3 Applied in a Gridshell

Consider a 3x3 point grid, quadratic interpolated between the points to create a double curved surface, as shown in Figure 4.35. For simplicity, assume no applied loads and 90x90 mm glulam cross-section.

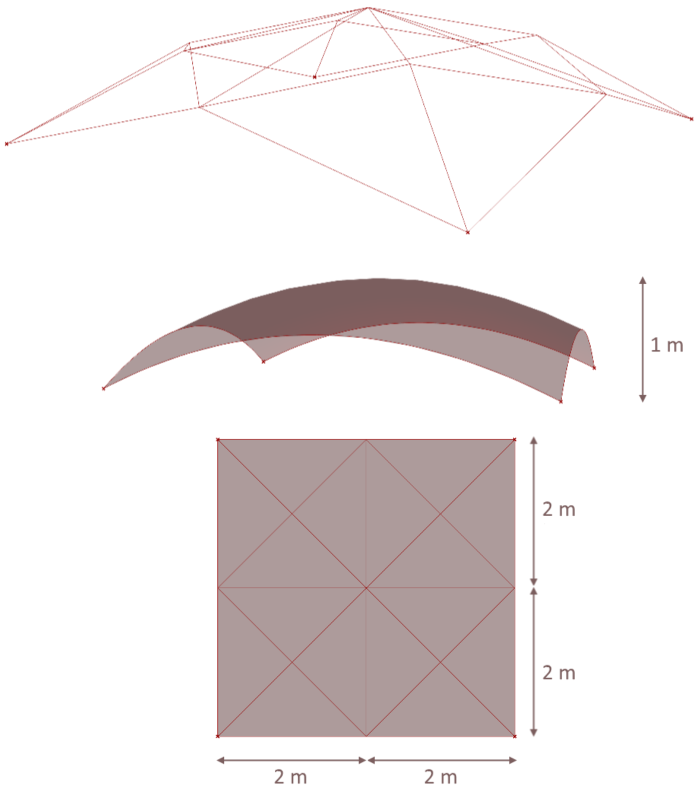


Figure 4.35: The geometry for the test grid shell.

The resulting 3D-geometries are generated and visualized in Figure 4.36. All the connections were generated successfully. The beams and plates are also sufficiently spaced according to the constructional limitations, e.g. beam spacing, plate spacing, bolt spacing. Note that the bolts are placed in a row parallel to the fiber direction despite the optimization algorithm checking the opposite as advantageous. This is due to timber beam not having enough height to fit two bolts vertically. If the cross-section height was increased, the bolts would be spaced differently.

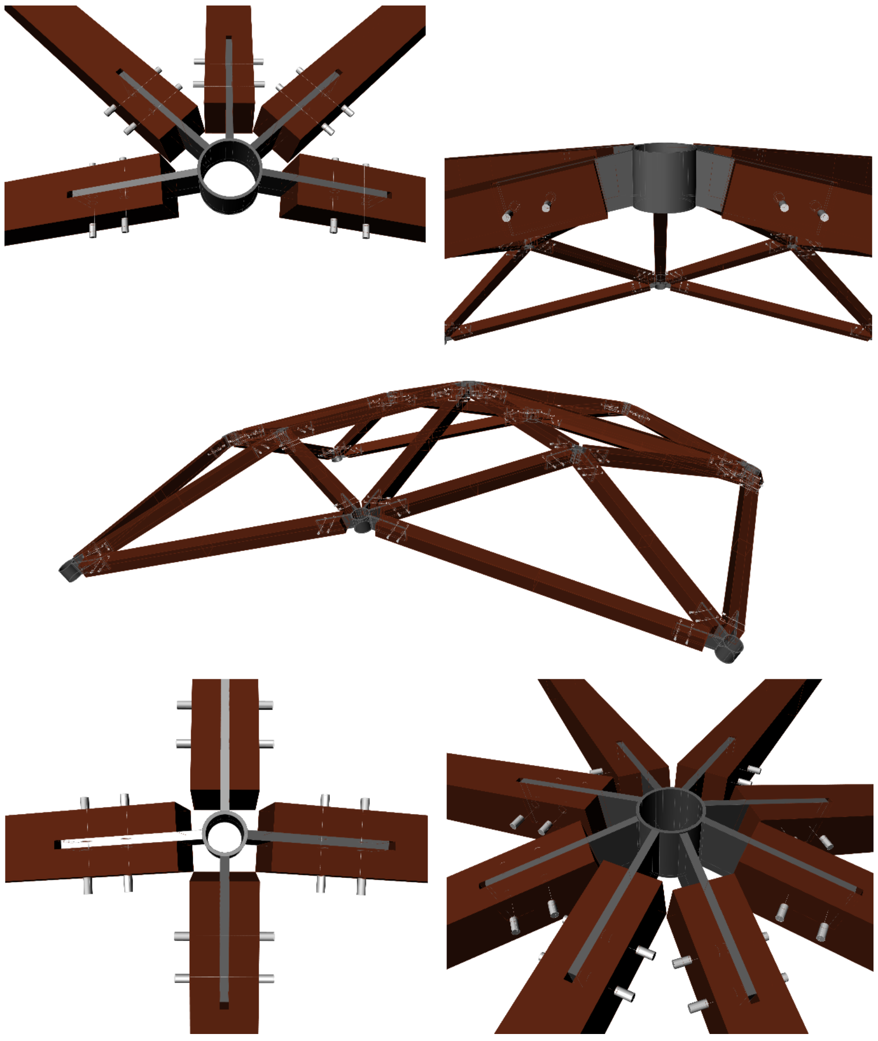


Figure 4.36: The 3D-model generated through the digital workflow.

Chapter 5

Case Study: Grid Shell Student Cabin in Norway

5.1 General About the Project

As a part of this thesis, the digital workflow developed in the previous chapters was implemented into an actual project to test the methodology of design. The case of interest involved the design of a student cabin in Fosen, Norway. The Conceptual Structural Design Group (CSDG) at the Norwegian University of Science and Technology (NTNU) was engaged by the student sports organization at NTNU (NTNUI) to research the possibility of creating a cabin with a spatial roof structure. The decision of designing a timber gridshell is motivated by previous research done by CSDG.

The case study has been a collaboration between CSDG and two groups of master students at NTNU; Huseby and Eliassen [44] and the authors of this thesis. The two groups of master students have looked at different parts of the digital workflow concerning design and engineering in the project. Huseby and Eliassen [44] studied how a digital workflow oriented around a parametric model can be used in the design of global shape in spatial structures and used their results to propose shape, grid pattern and connection placement, together with PhD candidate Steinar Dyvig.

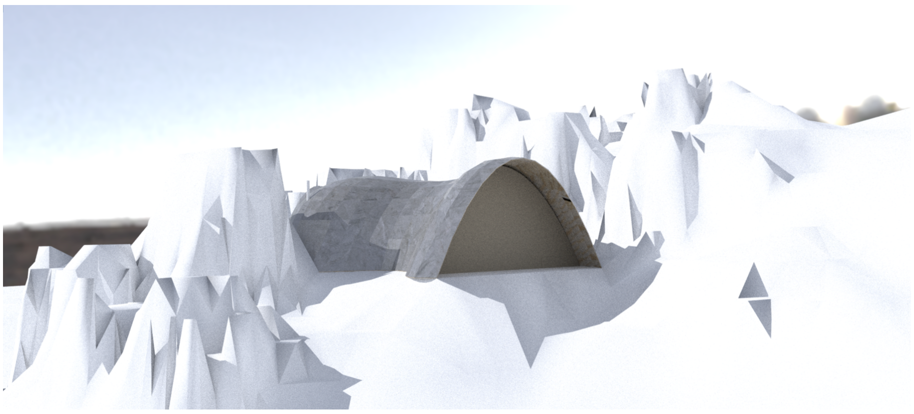


Figure 5.1: A basic render of the architectural concept behind the cabin (render by PhD candidate Steinar Dyvig).

The shape of the roof follows a double curved surface seemingly mimicking a barrel vault (illustrated in Figure 5.1 and 5.2). The grid is quadrilateral meshed and will feature a primary structural system connected to a secondary structural system that supports the cladding. The secondary structural system has been deemed out of scope for this research due to the complexity.

5.2 Basis for Optimization

The parametric model in this thesis has been developed to connect to the parametric model in Huseby and Eliassen [44]. This makes it easy to fetch the necessary information without manually inputting it. Figure 5.3 shows where the information is collected from the parametric model, fetching the following data: lines corresponding to the beams, the surface of the gridshell, the forces and moments for the beams and the cross-section used in the model. By attaching this information to the components shown in Chapter 4, this should be enough to generate connection designs automatically.

A result of the integration of the two models was the ability to experiment with different assumptions and study the effect. This enabled research of how different cross-sections, different grids, or different joint definitions (in Karamba) influence the necessary design of connections.

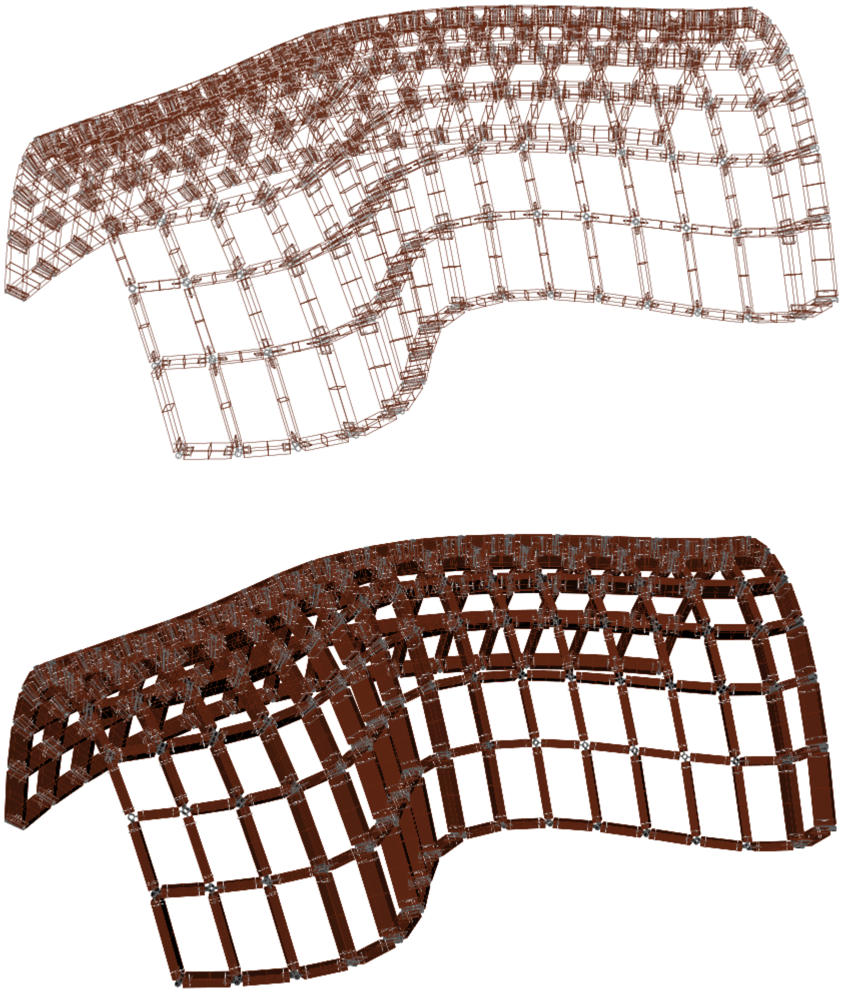


Figure 5.2: The goal of the model is to produce a valid connection design for all connections and corresponding 3D-models.

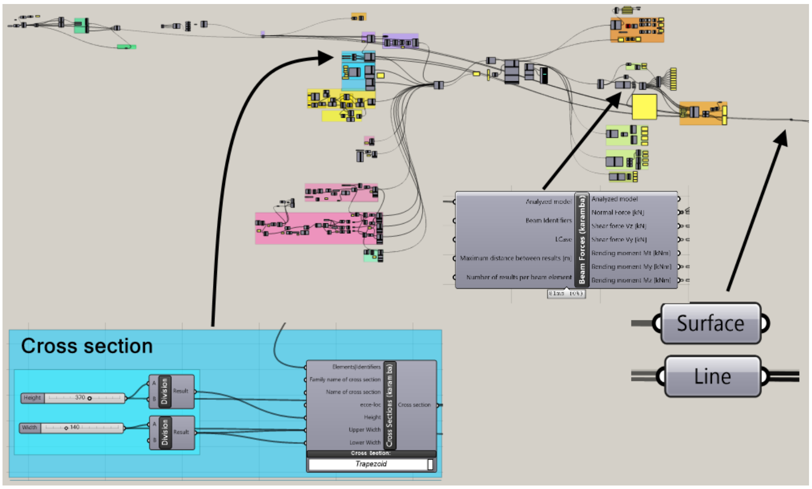


Figure 5.3: Extracting information from the parametric model of Huseby and Eliassen [44].

5.2.1 Joint Definition Assumptions

A study of different rotational definitions for the joints was carried out by testing the grid for:

1. No in-plane rotational stiffness ($C_{r,x} = 0$ and $C_{r,z} = 0$)
2. Fixed against in-plane rotation

The first definition reflects the properties of the connection design more accurately, while the latter simulates a situation where the connection has been strengthened. The study is particularly interesting in order to verify whether the connections have to be made stiffer due to the rectangular grid pattern or not. Figure 5.4 shows how the first joint type is defined in Karamba.

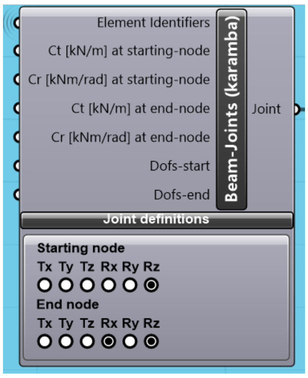

Element Identifiers
Ct [kN/m] at starting-node
Cr [kNm/rad] at starting-node
Ct [kN/m] at end-node
Cr [kNm/rad] at end-node
Dofs-start
Dofs-end
Beam-Joints (karamba)
Joint
Joint definitions
Starting node
Tx Ty Tz Rx Ry Rz
○ ○ ○ ○ ○ ●
End node
Tx Ty Tz Rx Ry Rz
○ ○ ○ ● ○ ●

Figure 5.4: The joint definition in Karamba that most realistically reflects the properties of the connection.

5.3 Results from Optimization

As the essence of a parametric model is its ability to verify multiple solutions quickly, the set of dimensions for glulam beams presented in Table 5.1 was tested in the model. The dimensions are according to some of the standard glulam beam dimensions from the distributor Moelven [45]. The table also presents if the different cross-sections have a valid solution or not. Based on the results, the following two cross-sections were chosen for further investigation: 360 x 140 mm and 225 x 115 mm.

For completeness, the different beams where also tested for a triangular grid, also proposed by Huseby and Eliassen [44], and visualized in Figure 5.14.

Table 5.1: Checking for success in convergence to a valid clamped or free connection design for rectangular grid or free connection design for triangular grid. O = Model converged to solution, X = Model did not find a solution.

	Clamped			Free			Triangle		
H/W	90	115	140	90	115	140	90	115	140
90	X	X	X	X	X	X	X	X	X
115		X			X			X	
135	X	X	X	X	X	X	X	X	X
180	X	X	X	X	X	X	X	X	X
225	X	O	O	X	X	X	O	O	O
270	O	O	O	X	X	X	O	O	O
315	O	O	O	X	X	X	O	O	O
360	O	O	O	X	X	O	O	O	O
405	O	O	O	X	O	O	O	O	O
450	O	O	O	X	O	O	O	O	O
495	O	O	O	X	O	O	O	O	O

5.3.1 Different Cross Sections

360 mm x 140 mm GL32c Cross Sections

This cross-section does not have noticeable problems with the capacity. The verification and optimization component gives a valid configuration with all plates when the limit is set to 100 % utilization. When all joints are clamped, the failure mode for all gusset plates is according to Johansen theory. By releasing the rotational freedom in the surface plane, the failure modes of some joints change to vertical bearing resistance.

The freedom of rotation due to the connection design creates some issues for the connections. Some particular areas are exposed to large forces when the surface is curved. Particularly the inner parts of the convex area need high connection capacity, as seen in Figure 5.5.

The freedom of rotation also causes a few problems regarding the space between the connections. As shown in Figure 5.13, many beams have limited space to fit the gusset plates and bolts groups. This is not a possible solution.

For the clamped joint definition, all bolt groups use two bolts positioned vertically, and the height of the beam enables the joints to position the bolts far from each other. This creates extra bending capacity for the bolt group and makes smaller bolts possible, which shortens the gusset plates. This decreases the need for space. The bolt placement close to the top and bottom edge of the gusset plate might also be the reason why bearing resistance becomes the failure mode for some of the connections.

The height of the cross-section might create problems regarding the applicability of beam theory. Some of the beams are very thick in comparison to the length, as illustrated in Figure 5.6. This could create issues regarding the transfer of forces through the bolts. It might be problematic if the forces in the timber find more efficient paths because of the spurious geometry. Hence some bolts take more forces than expected. This is further discussed in Chapter 5.3.4.

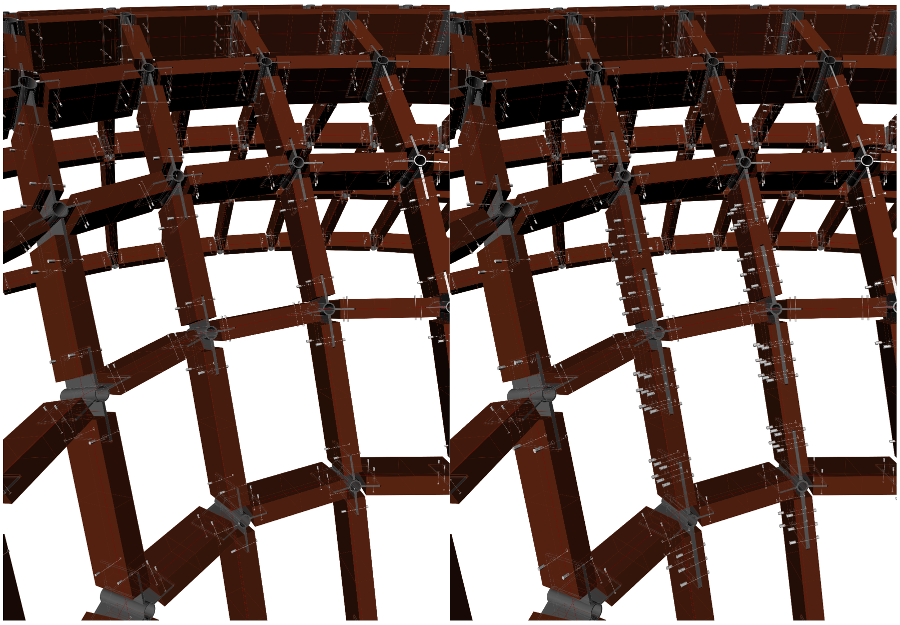


Figure 5.5: The different connections configurations for 360x140 based on joint definition; clamped to the left and free to the right.

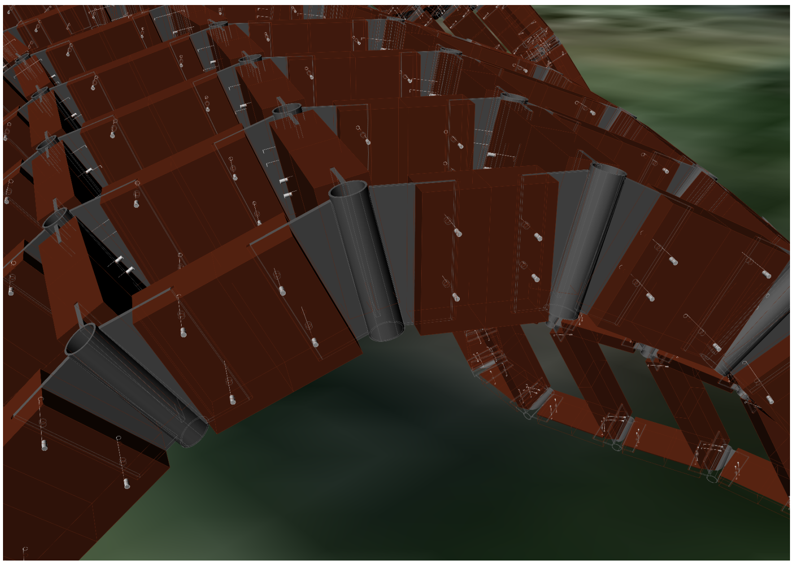
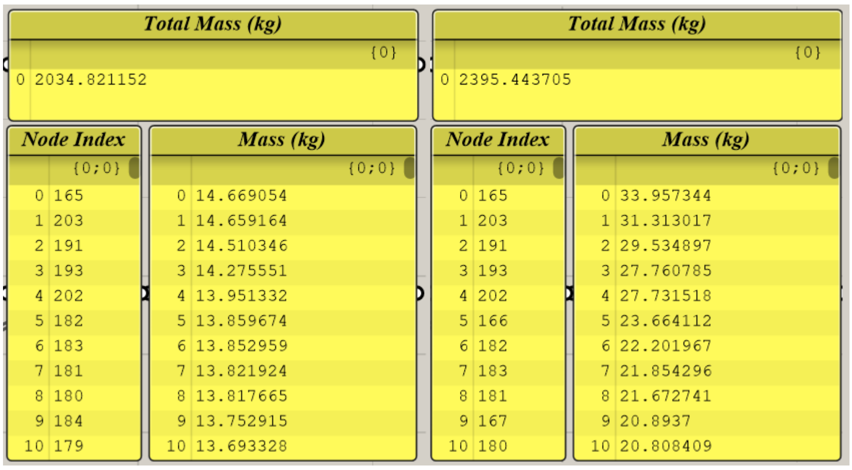


Figure 5.6: Beam 346 from the rectangular grid is actually taller than its length for 360x140 mm beams.

Figure 5.7 shows the weight of the ten heaviest connections in the grid for clamped connections and pinned connections. Both joint definitions (Chapter 5.2.1) are around 2 metric tonnes in total, but the pinned connection has larger extreme values than the fixed one. The parametric model used approximately 4 minutes to generate the 3D-models and 15 seconds to optimize.



Total Mass (kg) {0}

0 2034.821152

	Node Index {0;0}	Mass (kg) {0;0}
0	165	14.669054
1	203	14.659164
2	191	14.510346
3	193	14.275551
4	202	13.951332
5	182	13.859674
6	183	13.852959
7	181	13.821924
8	180	13.817665
9	184	13.752915
10	179	13.693328

Total Mass (kg) {0}

0 2395.443705

	Node Index {0;0}	Mass (kg) {0;0}
0	165	33.957344
1	203	31.313017
2	191	29.534897
3	193	27.760785
4	202	27.731518
5	166	23.664112
6	182	22.201967
7	183	21.854296
8	181	21.672741
9	167	20.8937
10	180	20.808409

Figure 5.7: The weight of the connections for 360 x 140 mm cross-sections. The left side is for clamped while the right side is for the real joint definition.

225 x 115 mm GL32c Cross Sections

The chosen cross-section is only structurally valid for the rectangular grid, when the connections are clamped. It does, however, have massive problems with fitting all the plates and bolts due to the low timber profiles. As shown in Figure 5.8, the plates often intersect with adjacent connection, which is unacceptable.

The weight of the connections is relatively high. Figure 5.9 shows the weight of the 10 heaviest connections, almost reaching 20 kg. The total weight of all the connections is still around 2 metric tonnes.

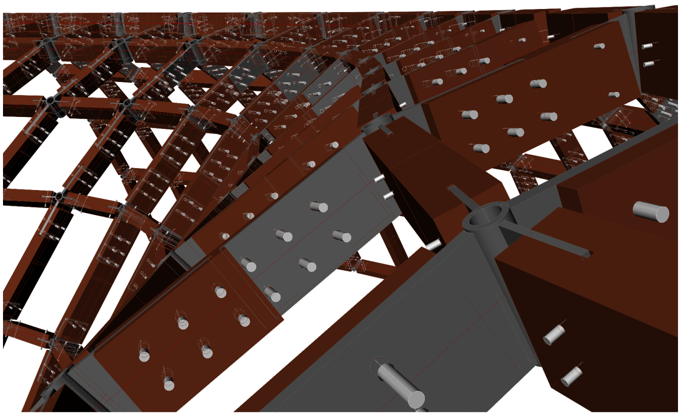
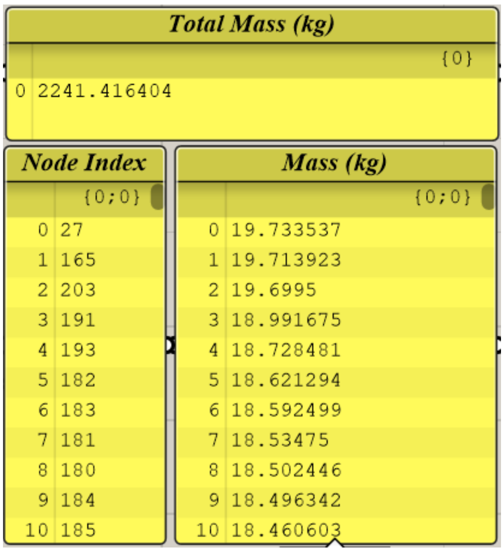


Figure 5.8: 225 x 115 mm cross-section for the rectangular grid. Many beams and bolts intersect with adjacent connections.



Total Mass (kg)

	{0}
0	2241.416404

Node Index

	{0;0}
0	27
1	165
2	203
3	191
4	193
5	182
6	183
7	181
8	180
9	184
10	185

Mass (kg)

	{0;0}
0	19.733537
1	19.713923
2	19.6995
3	18.991675
4	18.728481
5	18.621294
6	18.592499
7	18.53475
8	18.502446
9	18.496342
10	18.460603

Figure 5.9: Mass of the connections for the 225 x 115 mm cross-section for the rectangular grid.

The design used 59 seconds for structural optimization of the connections while the geometry modelling took approximately 3.5 minutes. The optimization was computer costly due to the number of bolts required. Structurally, close to all plates fail by Johansen theory before any other failure modes.

5.3.2 Triangular Grid

By applying the digital workflow on the triangular grid proposed in Figure 5.14, it appears to be no problems with the grid when it comes to structural stability, spacing the beams or overlapping plates from adjacent nodes. Figure 5.10 shows no problems with too little bolt group distance which occurred with the rectangular grid. The screenshot is taken close to the position of beam 346 from the rectangular grid, and there is clear evidence that it does not suffer the same issues.

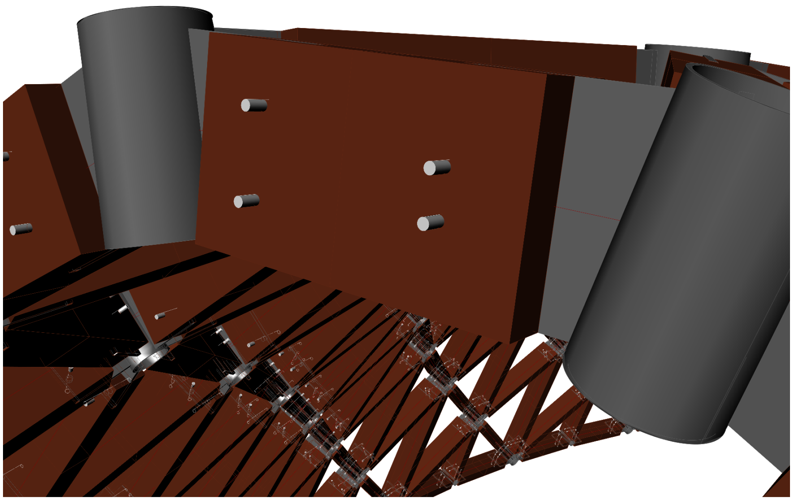
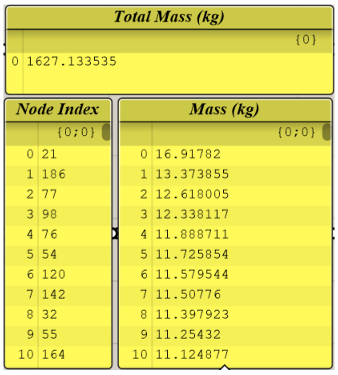


Figure 5.10: There are no indications for the same problems as in Figure 5.6 for the 225 x 115 mm cross-section in a triangular grid.

As shown in Figure 5.11, due to the extra (but lighter) gusset plates necessary to achieve the triangular grid, the heaviest connection mostly weigh around 12 kg, with the exception of one connection at 17 kg. It is although clear that the grid type saves around 20 % off the total mass of all the connections.

The program use 2.4 seconds for the optimization, which is a lot faster than the previous cases. It however used 4 minutes to generate the 3D-models.



```
Total Mass (kg)
{0}
0 1627.133535
```

```
Node Index
{0;0}
 0 21
 1 186
 2 77
 3 98
 4 76
 5 54
 6 120
 7 142
 8 32
 9 55
10 164
```

```
Mass (kg)
{0;0}
 0 16.91782
 1 13.373855
 2 12.618005
 3 12.338117
 4 11.888711
 5 11.725854
 6 11.579544
 7 11.50776
 8 11.397923
 9 11.25432
10 11.124877
```

Figure 5.11: Mass of the connections for the 225 x 115 mm cross-section for the triangular grid.

5.3.3 About the Lack of Shear Stability

The grid proposed in Huseby and Eliassen [44] is built using only rectangular shapes. This creates some challenges when it comes to lateral stability. If the grid were flat, there would be no resistance to the shear forces, and such a grid would collapse. The double curved surface will, however, have some stiffness because of its shape, but barely enough to be significant. The choice of the rectangular grid will therefore create difficulties with the type of connection proposed in this approach, especially since it demands very large dimensions for both glulam beams and the connections.

5.3.4 About the Lack of Room between Connections

The effect of short and tall beams and the large bolt groups might cause problems regarding the spatial orientation of the beams and bolt groups. In the worst case of the current configuration, some beams are as short as 0.5 meters, while being 140 mm wide.

Even in situations where the plates of adjacent connections do not intersect, there still might be problems with bolt groups placed too close together. Particularly the shear forces will behave with local effects, as seen in Figure 5.12. A simple experiment of a beam with two bolt groups in Abaqus CAE shows how fixing one of the bolt groups while displacing the other rigidly and vertically induces different stress-pattern in a short and a long beam. All parameters are equal except the length between the bolt groups. length corresponds to the total length of the plate. It is clear that the two beams have different load distributions, and as a consequence, the bolt forces are highly unpredictable and should not be evaluated according to classic beam theory for slender beams.

The grid proposed in Huseby and Eliassen [44] has beams as short as 0.568 m, something that can be problematic, particularly with beams higher than 200 mm. Figure 5.13 shows an actual situation where the forces will not behave as they would in a slender beam. Due to this, all beams with the possibility of less predictable shear force distribution, should be analyzed further, through a more detailed shell or solid FEA. Then is will be possible to map the local effects from the shear forces and give a realistic picture of the actual stress distribution.

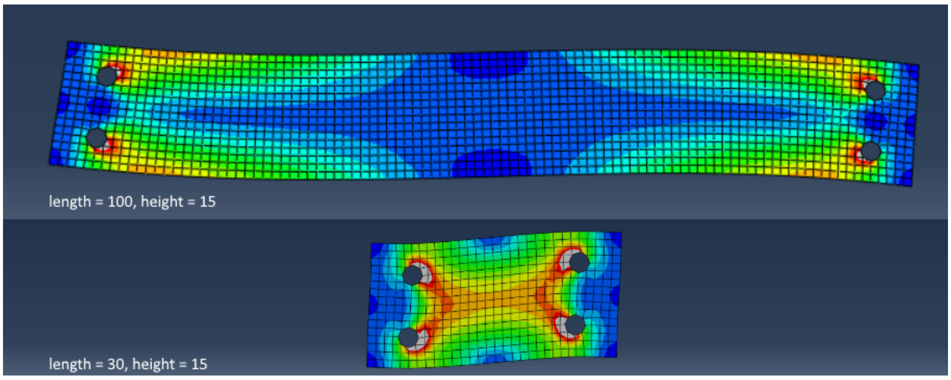


Figure 5.12: A simple experiment in Abaqus showing the local effect on stress due to thick beams in comparison to slender beams when subject to shear forces. Red or gray indicates large stress while dark blue indicates small stress.

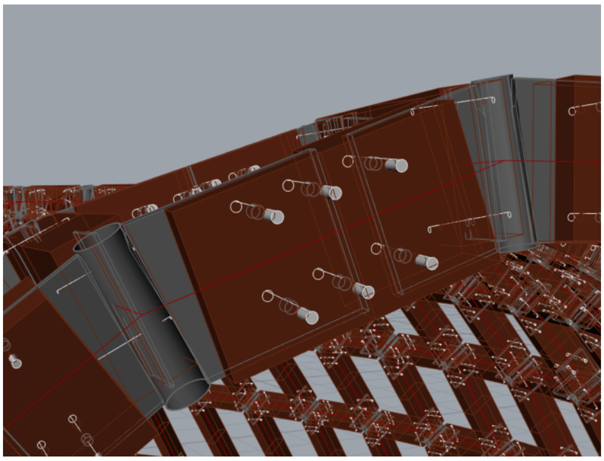


Figure 5.13: When the beams become very high in comparison to the length of the beams, ordinary beam theory might not apply.

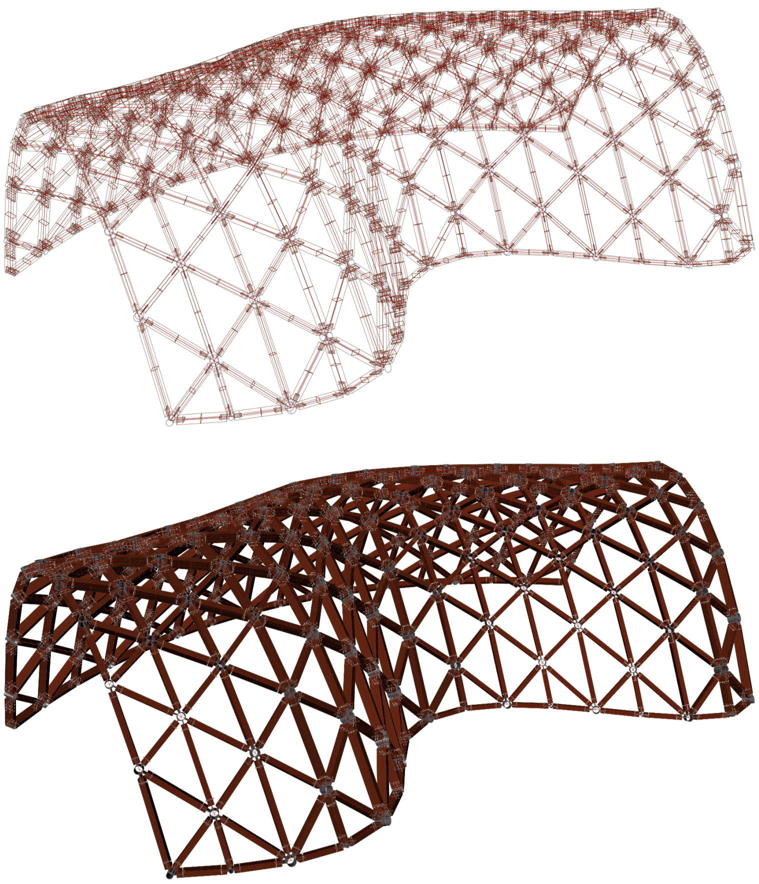


Figure 5.14: A 3D-model of the triangular grid, also proposed in Huseby and Eliassen [44].

5.4 Structural Verification in Abaqus CAE

To verify the results from the parametric model, one of the connections was further analyzed using FEM in Abaqus CAE Software. Node number 83 was arbitrarily chosen as the representative connection for the structure. Load case 3 was also chosen since this yields the largest design load.

From the fact that the Johansen approach and verification methods for bolts are well known through the Eurocodes, as well as the theory elaborated in Chapter 3.5, it was concluded that structural verification of the aluminum part of the connection should be in focus.

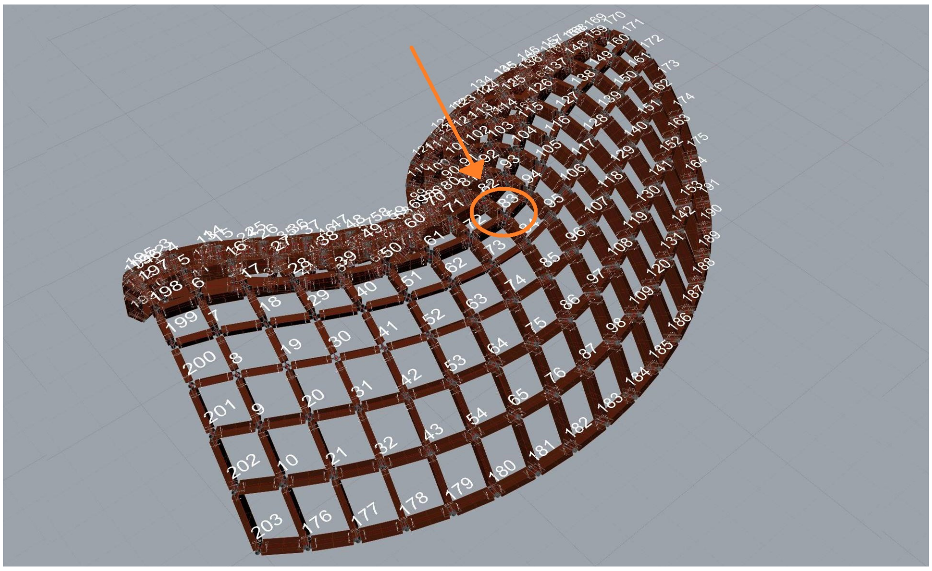


Figure 5.15: Position of node 83 in the gridshell.

5.4.1 Abaqus CAE Software

Abaqus is a FEA software developed by Dassault Systems. It is very flexible in solving different types of physical situations and enables the user to model a vast amount of different phenomenon in mechanics, thermomechanics, and viscomechanics. The software is a low-level software, where the interaction with the model happens on an elementary level, enabling the user to solve most engineering problems. Parts of the software are also implemented with advanced tools for solving problems, without having to interact on detail basis.

The basis of FEM and the general use of the Abaqus software is considered as general knowledge and is therefore not elaborated any further. Thus, it is important to point out that this chapter will only present some basic theory of special topics of interest in the use of FEM.

5.4.2 FEA Characteristics

The connection is modeled as a solid geometry in Rhino. The geometry is directly imported as individual parts from Rhino into Abaqus. The linear analysis is done in Abaqus Standard.

Discretization

To obtain as accurate and trustworthy results as possible using FEM, it is essential to define a good mesh. There are lots of different types of elements available in Abaqus, which makes it possible to mesh a wide range of geometries and structures. All the element types can be characterized by the number of nodes along the edges. Elements consisting only of corner nodes are called first-order elements (or linear elements). The first-order element is limited to linear interpolation along the edge because of the number of known nodes to extract values. If an edge-node is introduced, the element becomes a second-order element (or quadratic element) due to the possibility of extracting values from three nodes, and thereby quadratically interpolating over the edge.

A hexahedron element is categorized as a *brick*-element. It is a polyhedron consisting of six faces, eight corners, and twelve edges. It is used to model three-dimensional solids. The first and second-order hexahedron elements are illustrated in Figure 5.16. Three-dimensional triangular and tetrahedral elements are used in a lot of automatic mesh generators, including Abaqus CAE, as it is a convenient element for complex shapes. However, hexahedrons have a better convergence rate and usually provides a solution with the same accuracy at less cost. In case of very large distortions, the brick element will in general be more sensitive to this than the tetrahedral.

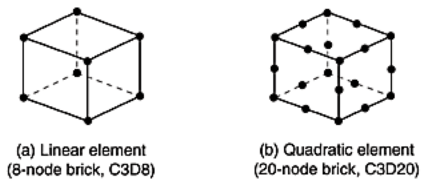


Figure 5.16: (a) First- and (b) second-order hexahedral elements [46].

For the model the chosen element type is C3D8R; 8-node Linear Brick Element with Reduces Integration and Hourglass control. Reduced integration implies that an integration scheme of a lower order is used. For first-order elements, this means integrating over only one point per element. With this comes the risk of spurious zero-energy modes because of the lack of integration points. In Abaqus, this is solved by introducing an hourglass control. In the Abaqus Manual [46], it is also recommended that these kinds of elements is used with reasonably fine meshes. The hexahedron elements used for this analysis has three degrees of freedom in each node and has approximate element size of 4 mm. The meshed model is shown in Figure 5.17.

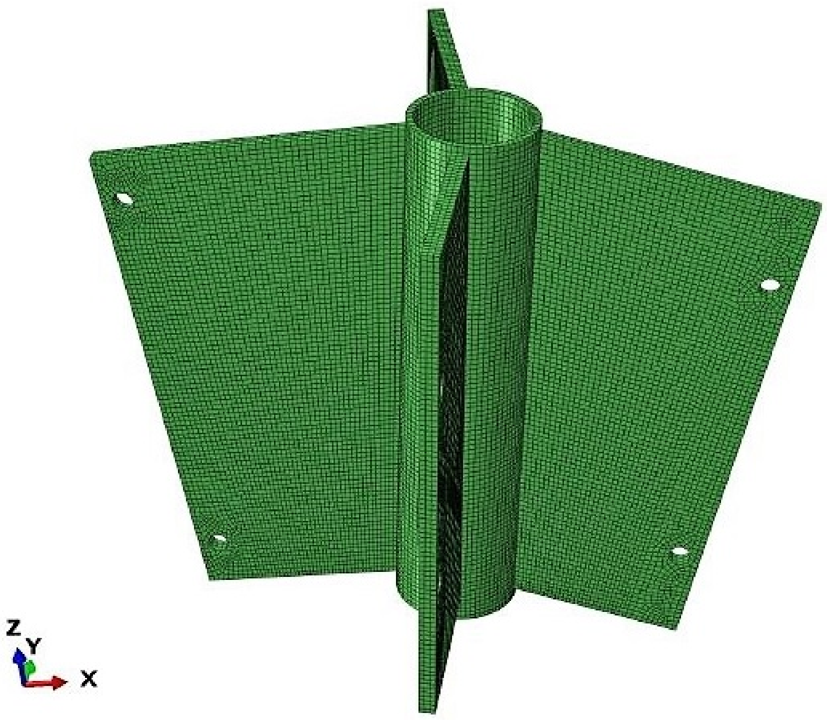


Figure 5.17: Mesh of the solid model.

Model Properties

Connection properties for node 83 used in the FEA are given in Table 5.2 and 5.3.

Table 5.2: Node properties for node 83, all distances in meter.

Node Properties	
Diameter of cylinder	0.0927
Thickness of cylinder	4.8
Top edge of cylinder coord.	0.189
Bottom edge of cylinder coord.	-0.204
Height of cylinder	0.370

Table 5.3: Plate properties for node 83, all distances in meter.

Plate Properties	Connecting Node			
	72	94	82	84
Width of timber	0.140	0.140	0.140	0.140
Height of timber	0.370	0.370	0.370	0.370
Rotation roll [Rad]	6.268	0.017	0.025	0.019
Rotation yaw [Rad]	4.728	1.842	3.149	0.017
Rotation pitch [Rad]	-0.023	-0.029	-0.201	-0.186
Distance to timber edge	0.078	0.102	0.136	0.109
nBoltsX	1	1	1	1
nBoltsZ	2	2	2	2
Thickness of gusset plate	0.012	0.012	0.012	0.012
Boltsize	0.012	0.012	0.012	0.012
Utilization	92 %	99 %	95 %	91 %
Spacing parameter a1	0.060	0.060	0.060	0.060
Spacing parameter a2	0.161	0.161	0.298	0.236
Spacing parameter a3	0.084	0.084	0.084	0.084
Spacing parameter a4	0.104	0.104	0.036	0.067
Spacing parameter e1	0.016	0.016	0.016	0.016
Spacing parameter e2	0.016	0.016	0.016	0.016
Spacing parameter p1	0.029	0.029	0.029	0.029
Spacing parameter p2	0.031	0.031	0.031	0.031
Distance to Bolt Group Center	0.162	0.186	0.220	0.193

Material Properties

The material data used for the connections is corresponding to the Nickel-Aluminum-Bronze alloy C95500, produced by *Green Alloys* (further presented in Appendix D). It is defined as an Elastic Isotropic material with Young's Modulus 110 GPa and Poisson's Ratio 0.32. The yield strength is 290 MPa while the ultimate tensile strength is 655 MPa.

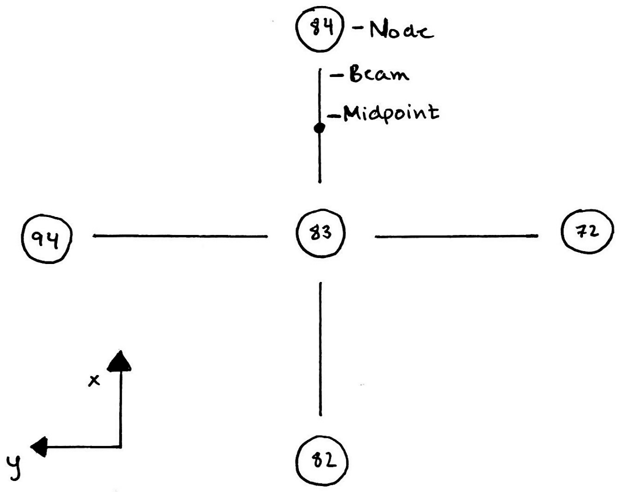


Figure 5.18: Node 83 with connecting nodes and beams.

Application of Load

The loads are applied to the model as rigid displacements of the bolt hole. Grasshopper node 83 is connected to four other nodes in the grid (see Figure 5.18). To achieve a conservative estimate, the displacement of the bolt group is fetched at the midpoint of the beams (also illustrated in Figure 5.18), where the long distance between measuring points will lead to a larger displacement difference between the bolts. The values of the displacement are derived from a Karamba analysis in the parametric model of Huseby and Eliassen [44]. The displacements are then applied around the inner surface of the bolt hole. All the displacements are presented in Table 5.4.

Table 5.4: Applied displacements for the individual plates from the digital workflow in the direction of the given connected nodes.

Connecting Node		Displacement [m]
72	X	-0.001566
	Y	-0.000041
	Z	-0.000517
94	X	-0.001581
	Y	-0.00004
	Z	-0.000539
82	X	-0.001565
	Y	-0.000031
	Z	-0.00026
84	X	-0.001678
	Y	-0.000054
	Z	-0.002369

Results

With reference to the plot of the Von Mises stress in Figure 5.19, the stress is mainly gathered close to the bolt holes, but also perceptible in the cylinder. It is clear that the stresses do not exceed the yielding strength of the material in any areas (Equation (5.1)). The continuous flow of stress between the opposing plates implies an accurate distribution and strengthens the validity of this model.

$$\begin{aligned} \sigma_{\text{Model,max}} &\leq \frac{\sigma_{\text{y,C95500}}}{\gamma_{\text{M0}}} \\ 171 &\leq \frac{290}{1.15} = 252 \end{aligned} \tag{5.1}$$

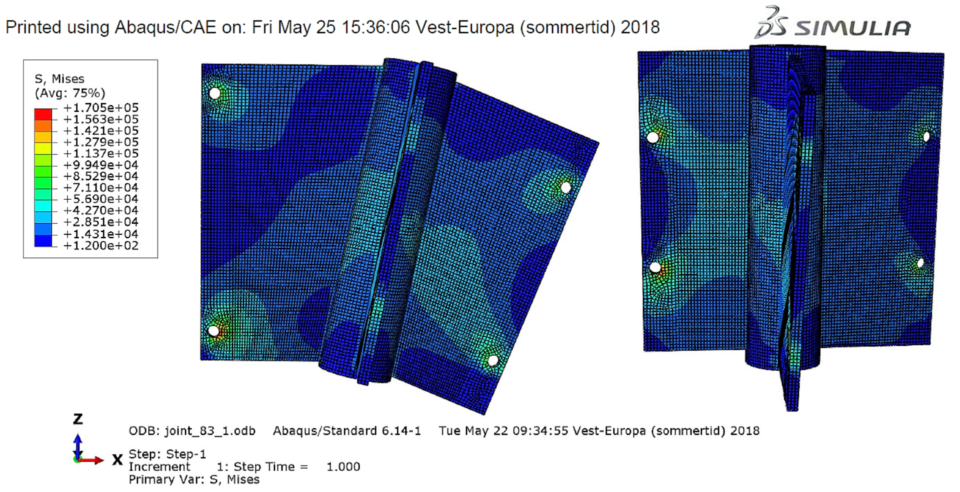

Printed using Abaqus/CAE on: Fri May 25 15:36:06 Vest-Europa (sommertid) 2018
SIMULIA
S, Mises
(Avg: 75%)
+1.705e+05
+1.563e+05
+1.421e+05
+1.279e+05
+1.137e+05
+9.949e+04
+8.529e+04
+7.110e+04
+5.690e+04
+4.270e+04
+2.851e+04
+1.431e+04
+1.200e+02
Z
X
ODB: joint_83_1.odb Abaqus/Standard 6.14-1 Tue May 22 09:34:55 Vest-Europa (sommertid) 2018
Step: Step-1
Increment 1: Step Time = 1.000
Primary Var: S, Mises

Figure 5.19: Overview of the Von Mises stress distribution [kPa].

As all models, this should be interpreted on the basis of the given model features. As this is a simplified FEA, it is necessary to point out that application of the displacement throughout the bolt holes will not give an accurate result near these. This kind of application will lead to an overly constrained model without the necessary freedom of rotation in the bolt holes, hence the stresses in the areas close to the bolt holes will be unreliable.

The stress distribution in the other parts of the model will, on the other hand, give a more accurate indication of the behavior. This includes the concentration of shear forces, between the cylinder and the gusset plate, which could result in punching shear failure, and generally how the plates and cylinder behave together. The interesting area is then where the forces is transferred between the opposing plates through the cylinder. According to the model, this is not critical for node 83.

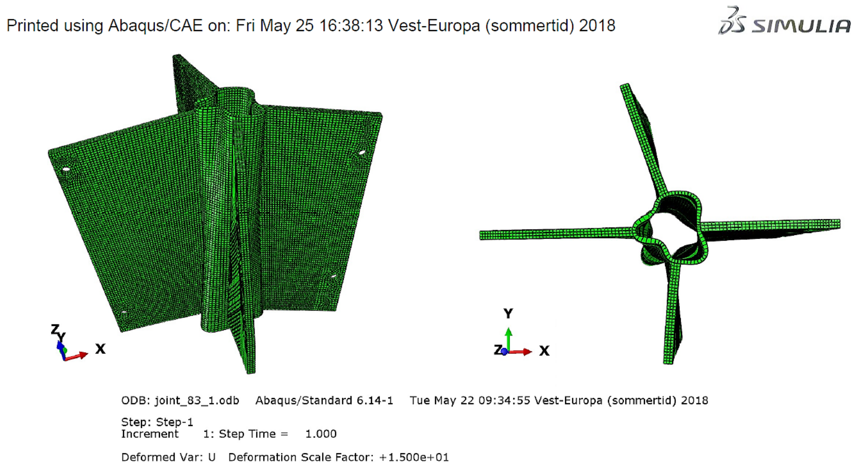

Printed using Abaqus/CAE on: Fri May 25 16:38:13 Vest-Europa (sommertid) 2018
SIMULIA
Z
Y
X
Y
Z
X
ODB: joint_83_1.odb Abaqus/Standard 6.14-1 Tue May 22 09:34:55 Vest-Europa (sommertid) 2018
Step: Step-1
Increment 1: Step Time = 1.000
Deformed Var: U Deformation Scale Factor: +1.500e+01

Figure 5.20: Deformation with scale factor 15.

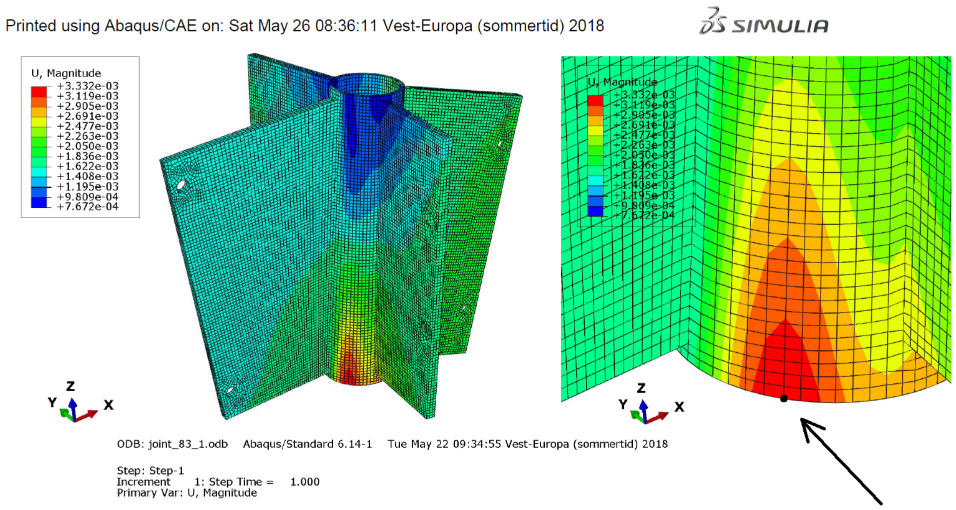

Printed using Abaqus/CAE on: Sat May 26 08:36:11 Vest-Europa (sommertid) 2018
SIMULIA
U, Magnitude
+3.332e-03
+3.119e-03
+2.905e-03
+2.691e-03
+2.477e-03
+2.263e-03
+2.050e-03
+1.836e-03
+1.622e-03
+1.408e-03
+1.195e-03
+9.809e-04
+7.672e-04
Z
Y
X
ODB: joint_83_1.odb Abaqus/Standard 6.14-1 Tue May 22 09:34:55 Vest-Europa (sommertid) 2018
Step: Step-1
Increment 1: Step Time = 1.000
Primary Var: U, Magnitude

Figure 5.21: Deformation contour plot.

The deformation of the connection is illustrated in both Figure 5.20 and 5.21. The maximum deformation appears in the node pointed out in Figure 5.21, and has a magnitude of 3,3 mm. The relative displacement to the 4,8 mm thick cylinder is then quite noticeable. Nevertheless, considering both the conservative application of the load, and the fact that this load case it the worst possible case, this deformation will be considered as reasonable. The deformation is illustrated in 1:1 scale in Figure 5.22.

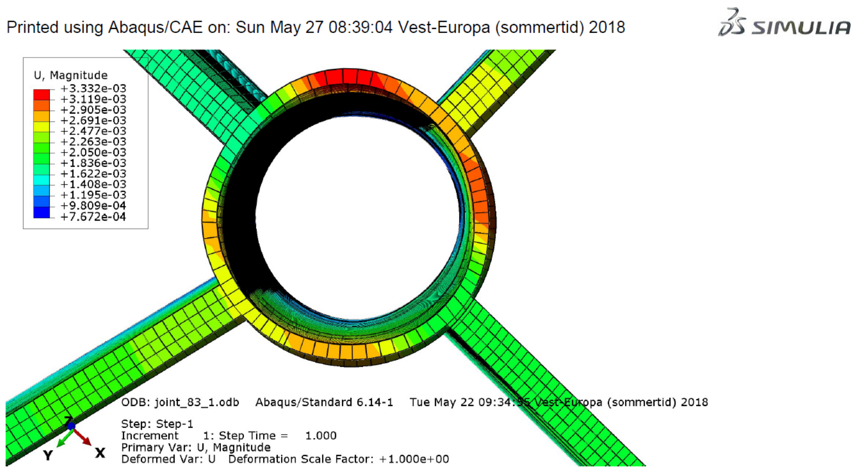

Printed using Abaqus/CAE on: Sun May 27 08:39:04 Vest-Europa (sommertid) 2018
SIMULIA
U, Magnitude
+3.332e-03
+3.119e-03
+2.905e-03
+2.691e-03
+2.477e-03
+2.263e-03
+2.050e-03
+1.836e-03
+1.622e-03
+1.408e-03
+1.195e-03
+9.809e-04
+7.672e-04
ODB: joint_83_1.odb Abaqus/Standard 6.14-1 Tue May 22 09:34:35 Vest-Europa (sommertid) 2018
Step: Step-1
Increment 1: Step Time = 1.000
Primary Var: U, Magnitude
Deformed Var: U Deformation Scale Factor: +1.000e+00
Y
X

Figure 5.22: Deformation contour plot with scale factor 1.

5.4.3 Comparison of Parametric Results and FEA

In the parametric model (see Chapter 4), two different failure modes for the cylinder is validated; the punching shear capacity and the cylinder face failure capacity. By extracting from the *Structural Verification and Optimization* component, the maximum utilization derived for node 83 is 14.94 %, and is caused by cylinder face failure. The maximum utilization for shear punching is 2.63 %.

Considering the shear punching, it is clear that this is not a critical failure mode according to the Grasshopper model. Looking at the stress distribution in Figure 5.23 from the Abaqus model, the concentration of forces that characterize shear punching is not present in the stress distribution. Comparing this to the 2.67 % utilization value from Grasshopper gives reason to consider the values from Abaqus as reasonable, but not evident that the calculations are correct.

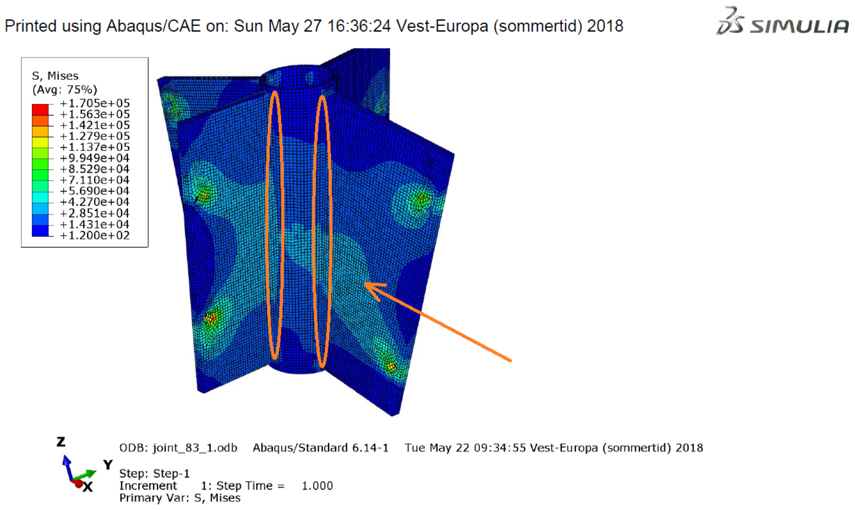

Printed using Abaqus/CAE on: Sun May 27 16:36:24 Vest-Europa (sommertid) 2018
S, Mises
(Avg: 75%)
+1.705e+05
+1.563e+05
+1.421e+05
+1.279e+05
+1.137e+05
+9.949e+04
+8.529e+04
+7.110e+04
+5.690e+04
+4.270e+04
+2.851e+04
+1.431e+04
+1.200e+02
ODB: joint_83_1.odb Abaqus/Standard 6.14-1 Tue May 22 09:34:55 Vest-Europa (sommertid) 2018
Step: Step-1
Increment 1: Step Time = 1.000
Primary Var: S, Mises

Figure 5.23: Illustration of where the stressed areas should appear in case of shear punching.

To compare the cylinder face failure utilization at 14.94 % with the results from Abaqus, the stress in an element in the cylinder wall is derived from one of the most stressed areas. The value of the Von Mises stress at this point (see Figure 5.24) is 32 MPa, which gives a dimensioning utilization of 12.7 %. The Grasshopper gives a higher utilization rate, which indicates that the Grasshopper model calculations are more conservative than the Abaqus model.

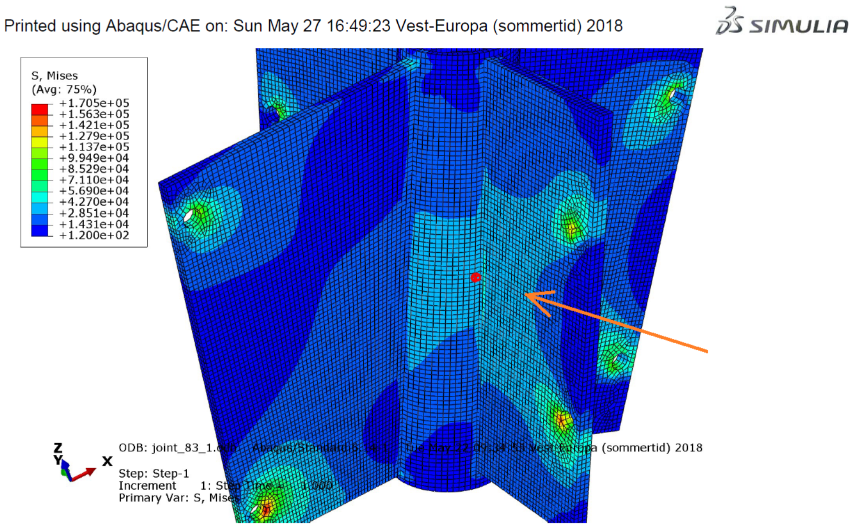

Printed using Abaqus/CAE on: Sun May 27 16:49:23 Vest-Europa (sommertid) 2018
S, Mises
(Avg: 75%)
+1.705e+05
+1.563e+05
+1.421e+05
+1.279e+05
+1.137e+05
+9.949e+04
+8.529e+04
+7.110e+04
+5.690e+04
+4.270e+04
+2.851e+04
+1.431e+04
+1.200e+02
ODB: joint_83_1.odb Abaqus/Standard 6.14-1 Tue May 22 09:34:55 Vest-Europa (sommertid) 2018
Step: Step-1
Increment 1: Step Time = 1.000
Primary Var: S, Mises

Figure 5.24: Element used for collecting values from the cylinder wall.

As it is clear that the two models are highly comparable, it is interesting to discuss the accuracy of both. The Abaqus model has applied displacements and not forces. These displacements are derived from the midpoint of the glulam beam. As glulam is a lot softer than the alloy C95500, applying the displacement from the middle of the glulam beam to the bolt holes is a very conservative assumption. Additionally, the application method makes it meaningless to check for bearing capacity in the bolt holes.

There are also conditions which affect the Abaqus model non-conservatively. As mentioned, the application method of the load results in an overly constrained model. The results can often be unrealistically high stresses. The application of the load could also be less conservative as the same displacement is applied in both bolt holes for all the plates. On the one hand, this is preventing a possible bending moment in the plate caused by rotation of the bolts. On the other hand, it also introduces extra shear forces throughout the connection since the connection no longer can rotate.

In Grasshopper, the calculations are implemented from the Eurocodes, which contains several conservative safety factors. In addition to this, certain simplifications have been made to save both time, work and computational cost. E.g. the algorithm for spacing is conservatively calculating the minimum spacing needed for the beams without taking into account the possibility to translate the beams different distances from the connection center. The multiple safety factors, combined with the simplifications made in Grasshopper, result in a conservative model.

From the considerations above, the FEA in Abaqus has given a good indication of the accuracy of the cylinder calculations. The level of the stresses from the two models seems to resemble each other, and the utilization from the FEA is slightly lower than the utilization from the Grasshopper model. There are too many influence factors to verify the certain calculations, but this gives a good indication that the digital workflow offers rational values, and calculates slightly more conservative than the FEM model. The deformation of 3.3 mm also indicates what kind of tolerance to use for the spacing algorithm.

Chapter 6

Discussion

6.1 Critical Components of the Structure

The results from the parametric model and the case study show that failure according to the Johansen theory often is the limiting failure mode for the connections. Only the bearing resistance in the aluminum plate occasionally appears as the failure mode for gusset plates with small distance between the edge and bolt hole. As the Johansen theory is a well known experimental method and the implementation of it in this connection design follows traditional use, the results should be accurate and reliable.

An issue propounded in the case study in Chapter 5 is regarding the connections' resistance against in-plane rotation. As previously stated, the rectangular grid shape combined with the current connection design induces problems with the global static stability of the construction. There are primarily two solutions that fix this issue while keeping most of the simplicity of the design: making the grid triangular instead of rectangular or making the connections stiffer.

The issue raised is of particular interest due to the conflict of interest between the two fields of architecture and engineering. The triangular grid makes sense from an engineering perspective while strengthening the connections showed to be a better option from the architectural perspective. An engineer's ability to consider the architectural concept is essential for a productive discussion climate. Since the connections can be strengthened, it is a more reasonable option to strengthen them rather than changing the grid design, hence keeping the concept of the architect.

Also put forward in Chapter 5 is the issue of spatial configuration of beams and plates.

The beams and connections have problems fitting in the grid when the cross-section or bolt group becomes large. This is mainly a problem because of the stability problem, described in Chapter 5.3.4, redistributing the related forces in less advantageous ways. From the model, it also seems like having rows with two or more bolts often create issues due to the necessary spacing distance between the bolt holes. It is therefore better to expand the bolt groups perpendicular to the surface rather than parallel to it, by ,e.g., increasing the height of the beam cross-section. If the decision is to alter the design, the bolts should be placed as close to the connection as possible.

6.2 Flexibility

The parametric model, proposed in this thesis, worked efficiently on different gridshells, as shown in Chapter 4.5. The shorter processing time of the parametric model, in comparison to traditional CAD, enables quick and low-cost simulations of the gridshell, again making the digital workflow between CAD and CAE more efficient. It became clear during the case study that the 3D-generation of the model was much more computer costly than the optimization process. This made it practical to disconnect the 3D-modelling process when researching the structural behavior of different grids, and only activate it when a visual inspection was necessary. It was also often advantageous to isolate the connections of particular interest, and only display that particular connection to save computing time. Clever use of data structures, e.g. data trees, can potentially increase efficiency, since storing the properties and geometries of the connections separately and organized enables the user easy access.

For the connection design in consideration, the model is flexible and fully parametric. The model automatically treats the individual parts (e.g., bolts, steel elements, and beam members) differently and calculates according to the material. The parametric model was also successfully connected to the parametric model of Huseby and Eliassen [44]. By close collaboration (both digitally and through meetings), the two teams of engineers could communicate and discuss the specifications required for such an interaction. The data transferred between the model was kept to the absolute minimum to ensure compatibility.

6.3 Troubleshooting of the Parametric Model

Having an on-demand visualization of the structure helps discovering errors more quickly. The implementation of methodology will always reflect the user's ability to visualize and identify problems. The use of computer technology to enhance the visual experience should therefore lead to more accuracy in the model. An iterative approach of implementation and visual control can create a feedback loop, incrementally improving the model (see Figure 6.1). Many errors from hand calculation in traditional engineering are not discovered, as the presentation only contains numbers. Once these numbers are directly connected to a parametric model, they are represented and visualized in 3D. The different components are visualized simultaneously, which makes abnormalities easier to discover. This applies to, e.g., dimensions or the orientation of the individual parts. A complex parametric model often contains multiple coordinate systems which need to be organized.

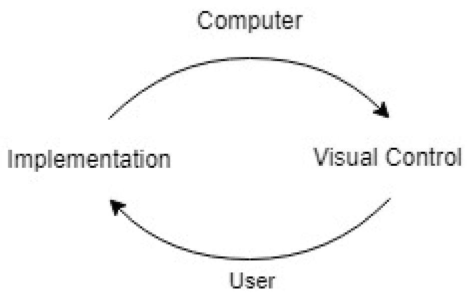


Figure 6.1: Principle of visual control.

An interesting perspective that reflects the feedback loop is the global overview of the model behavior, and the realistic consideration of the model. An example of a discovered problem from the case study was intersecting plates (see Chapter 5.3.4). At a point during the development of the gridshell connections, the input grid to the parametric model was highly refined, which was problematic due to the small distance between the connections. For some of the beams, the slotted plates from both sides of the beam intersected. Due to this, the design decision of the refinement of the grid had to be taken into account during the optimization process. This error would probably be discovered during other design processes as well, but it is worth mentioning that this was a trivial error to discover during this process.

Validation of both the structural calculations and the geometrical algorithms in the model is also done through testing of several input grids. Even if the model seems to behave correctly in one case, does not imply that the model is verified. In this sense, it is difficult to make a final verification, as it will be impossible to test an infinite number of input grids. Testing the model with different input grids will nevertheless give an increasingly improved model.

Another key point when it comes to troubleshooting is the problem with an undiscovered error. For a complex parametric model, an error can influence the whole model. This, for instance, could be a logic mistake in the code or the design approach, implemented calculation methods or software bugs. During the process of developing such a digital workflow, it is therefore essential to implement the different features as accurate as possible. Therefore, the possibility of developing customized verification methods to ensure the validity of the methods of design should be examined. Currently, the process is limited when it comes to checking the validity, which increases the probability of undiscovered errors. The troubleshooting is mostly based on visual control in Rhino and engineering experience.

6.4 Reduction of Redundant Work through Generalization of Code

Despite the model working well with the developed connection type, it is important to point out that the digital workflow used is customized for this specific connection type, but generalized for every input grid. This is considered to be necessary as the process of making a generalized parametric model for multiple connection types would be very time-consuming. Although the customized model facilitates an efficient process of designing the digital workflow, the simplicity can also be a problem. As uncovered in Chapter 5, the problems of the selected grid, makes the chosen connection type lack in compatibility with the grid. Completely altering the design concept can be problematic with the selected workflow. It requires reprogramming of Grasshopper-components for structural verification and spatial limitations, in addition to changing the methods of structural analysis. This is particularly pointed out by the lack of stability of the gridshell for in-plane transverse motion, where the connections have to be stiffened. This shows the importance of establishing design requirements, as in Chapter 2.

It is advantageous to make digital code reusable for other projects, particularly regarding structural verification and spatial calculations which is particularly exposed to simple calculation mistakes. Creating generalized calculation tools is therefore potentially very profitable, particularly if direct implementation to a new model is possible. A detailed study about generalized code was considered out of scope for this thesis.

Still, many choices in the parametric model makes the code generalized, e.g., due to the choice of treating any cross-sections as inscribed in a rectangle during spatial calculations, the choice of cross-section or the choice of connection type does not alter the code or method of calculation. This quality makes researching different designs or making alterations potentially more efficient and profitable.

6.5 AEC Collaboration

Developing the connections and being a part of the NTNUI cabin project has proven to be a good example of working in the intersection between engineering and architecture. The engineer, as well as the architect, has to take responsibility for the form of the shell geometry. This is, among other things, because of the structural challenges related to the smoothness of the gridshell. The angle of the connection will to a high degree influence the design. As found in Chapter 5.3.4, it is disadvantageous to have connected beams with too much angular difference, as this can complicate the connection geometry, as illustrated in Figure 4.29.

The model's ability to give solutions for different shell geometries increases the possibilities for improved cross-disciplinary discussion between architect and engineer. There should always be a balance between the architectural opinion about the form, and the structural considerations of the engineer. Using a digital workflow raised the communication to a higher level. The communication has the possibility to evolve around more complex and detailed subjects where both the engineer and the architect, with a common digital platform, easily can visualize how new input parameters may change the whole design (see Figure 6.2).

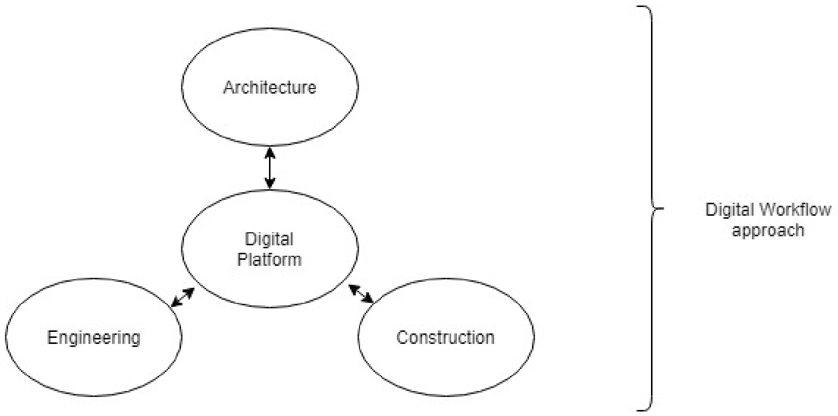


Figure 6.2: Visualization of digital workflow approach.

Even though the benefits of the digital workflow in the case study are many, the model does not work perfectly with the specific requirements from the architects as explained above. Much of the reason for this is the incompatible terms between engineers and architects. A possible solution to this could be to generalize the digital workflow and disconnect it from a specific design. This would require a digital platform with a more general framework between the three teams, that also has the possibility to evaluate different solutions. Possible solutions could involve the use of GA, as briefly introduced in Chapter 4.4.3, but could also be solved with the traditional approach.

Chapter 7

Conclusion and Recommendations for Further Work

7.1 Conclusion

Using a digital workflow has shown to entail great advantages. Mostly because of the on-hand visual control that creates a feedback loop for the participants. This leads to increased probability of discovering errors and less redundant work. The digital workflow approach creates a common digital platform built on common terms where different participants can experiment with changes, and quickly communicate opinions across disciplines.

The digital workflow is based on simple and established theories. It is based on a traditional engineering approach to structural problems, where the computer is instructed to mimic engineering decisions. For the case study, the connection design in the digital workflow did not work perfectly with the gridshell design proposed. The workflow still performed very well with terms established early in the process. The incompatibility between the terms of the architect and the engineers could be solved with better communication from early in the project, and in this case, understanding the impact on the connection design of different grid types.

The case study project consisting of groups with different perspectives create a foundation for a realistic process with many participants. It is reasonable to think that a process of bigger scale can increase in complexity and the demand for efficient communication. The experience through the case study shows that the use of a digital workflow involving a parametric model creates an enhanced work culture. The results acquired in this project therefore suggests that an efficient digital workflow can be a success factor for bigger projects involving spatial structures.

7.2 Recommendations for Further Work

This master thesis builds the foundation for further research on the following topics:

Implementing FEA-software in the Digital Workflow

To improve the digital workflow, further researching could be extended with individual FEA-software verification from the parametric model. Verifying critical parts of the connection could be essential for the results, and should be done for more than a single arbitrary node to ensure validity. This could be done by scripting in Abaqus CAE. The amount of work needed is hard to predict, but the computational cost of the digital workflow will increase drastically. The verification could be done either for a few or all connections.

Topology Optimization of Connections

Designing connections with broader possibilities of free-form could improve the customizability of the connection design. The parametric model in Grasshopper can be a suitable platform to implement topology optimization. The methodolgy presented in this thesis will not be a suitable foundation for such an approach, but certain parts concerning the timber to aluminum attachment could be applied to the model.

GA in Gridshell Connection Optimization

GA has a huge potential when it comes to the optimization process. This technology will be more effective and efficient with increasing computer capacity, and it is therefore essential with a future-oriented approach to the subject. Grasshopper comes with the Galapagos plugin, which offers easy connection to the parametric model. Although this thesis barely scratched the surface of the topic, there still exist a vast amount of possibilities regarding the subject.

Implement Gridshell Assembly Instruction

This parametric model is mainly oriented towards collaboration between architects and engineers, but it is also interesting to look more at the interaction between design and assembly. The assembly process for a gridshell with the individual, optimized connections for every joint is a comprehensive and complicated process.

As a method to simplify the process, a procedure for an optimized assembly process could be produced from the digital workflow. Some of the aspects it could contain are numbering of beams and connections, verifying that the assembly of bolts and beams is feasible, propose an optimal order of assembly regarding practical issues and provide further documentation of the parts.

Bibliography

[1] National Institute of Building Sciences. Frequently asked questions about the national bim standard.
URL: https://www.nationalbimstandard.org/faqs [Retrieved 2018-05 02].

[2] IPENZ. Improving collaboration between architects and engineers.
URL:
https://www.engineeringnz.org/documents/111/Improving_Collaboration_Between_Architects_and_Engineers.pdf [Retrieved 2018-05-01].

[3] Robert D. Cook. *Concepts and Applications of Finite Element Analysis*. 4th edition, 2002.

[4] Robert Hooke. *A Description of Helioscopes, and Some Other Instruments*. London, UK, 1676.

[5] John Ochsendorf and Philippe Block. Exploring shell forms. In *Shell Structures for Architecture*. Routledge, New York, NY, 2014.

[6] Laurent Ney and Sigrid Adriaenssens. Shaping forces. In *Shell Structures for Architecture*. Routledge, New York, NY, 2014.

[7] Foster and Partners. Great court at the british museum.
URL: https://www.fosterandpartners.com/projects/great-court-at-the-british-museum/
[Retrieved 2018-05-22].

[8] Bendik Manum. Trondheim gridshell (2015).
URL: https://www.ntnu.edu/kt/research/csdg/research/trondheim-gridshell
[Retrieved 2018-05-01].

[9] Michael Ramange. The wood from the trees: The use of timber in construction. In *Renewable and Sustainable Energy Reviews*. Elsevier, Cambridge, UK, 2017.

[10] AD Editorial Team. This wooden geodesic dome contains the world's largest planetarium.
URL: https://www.archdaily.com/891857/geodesic-dome-with-worlds-largest-planetarium-inside
[Retrieved 2018-05 02].

[11] Luca Caneparo. *Digital Fabrication in Architeture, Engineering and Construction.* Springer, 1st edition, 2014.

[12] Lalit Narayan, K. Mallikarjuna Rao, and M. M. M. Sarcar. *Computer Aided Design and Manufacturing.* Prentice-Hall of India Private Limited, New Delhi, 2008.

[13] Kerry Stevenson. Mx3d reveals first section of 3d printed steel bridge.
URL: http://www.fabbaloo.com/blog/2017/9/28/mx3d-reveals-first-section-of-3d-printed-steel-bridge
[Retrieved 2018-05-22].

[14] MX3D. Mx3d bridge.
URL: http://mx3d.com/projects/bridge-2/ [Retrieved 2018-05-22].

[15] Chee Kai Chua, Kah Fai Leong, and Chu Sing Lim. *Rapid Prototyping: Principles and Applications.* World Scientific Publishing Co. Pte. Ltd., Singapore, 2010.

[16] ASTM. Standard terminology for additive manufacturing technologies, 2012.

[17] Shapeways HQ. Aluminum.
URL: https://www.shapeways.com/materials/aluminum [Retrieved 2018-04 03].

[18] Nasdaq: MTLS. i.materialise.
URL: https://i.materialise.com/blog/how-3d-printing-in-aluminum-works/
[Retrieved 2018-05-01].

[19] Zhang Jianlong. *Global Design and Local Materialization.* Springer, Shanghai, China, 1st edition, 2013.

[20] EnginSoft. Aluminum sand casting.
URL: http://www.enginsoft.com/technologies/metal-process-simulation/aluminum-sand-casting/ [Retrieved 2018-04 03].

[21] Dr Dmitri Kopeliovich. Sand casting.
URL: http://www.substech.com/dokuwiki/doku.php?id=sand_casting
[Retrieved 2018-05 02].

[22] Michael Forde. *Construction and Building Materials*, volume 9th. Elsevier, 2010.

[23] Dragos-Iulian Naicu. Geometry and performance of timber gridshells. Master's thesis, The University of Bath, 10 2012.

[24] John Chilton and Gabriel Tang. *Timber Gridshells: Architecture, Structure and Craft*, volume 1st. Routledge, 2017.

[25] Woodays. Multihalle mannheim.
URL: http://www.wooddays.eu/sk/wood-architecture/projekt/detail/multihalle-mannheim/ [Retrieved 2018-05-22].

[26] Cullinan Studio. Downland gridshell.
URL: http://cullinanstudio.com/project/downland-gridshell
[Retrieved 2018-05-18].

[27] GAR Parke and P. Disney. *Space Structures 5*, volume 1st. Thomas Telford Publishing, London, UK, 2002.

[28] Iasef Md Rian. Frac shell.
URL: https://www.karamba3d.com/projects/frac-shell/ [Retrieved 2018-03-18].

[29] Andrei Nejur. Roppe bridge, .
URL: http://www.grasshopper3d.com/photo/roppe-bridge-18/prev?context =albumalbumId=2985220%3AAlbum%3A433364 [Retrieved 2018-05 02].

[30] Andrei Nejur. Roppe bridge, .
URL: https://www.scribd.com/document/110789878/ROPPE
[Retrieved 2018-05].

[31] SITD City Form Lab. Sitd city form lab poster, 2013.

[32] ArchDaily. Sutd library pavilion / city form lab.
URL: https://www.archdaily.com/387696/sutd-library-pavilion-city-form-lab
[Retrieved 2018-04-20].

[33] MASS Design Group. Design biennial boston.
URL: https://massdesigngroup.org/work/design/design-biennial-boston
[Retrieved 2018-03-11].

[34] Hans Larsen and Vahik Enjily. *Practical design of timber structures to Eurocode 5*. ICE Publishing, London, UK, 1st edition, 2009.

[35] Roberto Crocetti and Johan Fröbel. *Limtreboka*, volume 1st. Norske Limtreprodusenters Forening, 2015.

[36] Nicholas Williams, Daniel Prohasky, Jane Burry, Kristof Crolla, Martin Leary, Milan Brandt, Mike Xie, and Hamed Seifi. Challenges of scale modelling material behaviour of additive-manufactured nodes. In *Modelling Behaviour*. Springer International Publishing, Melbourne, Australia, 2015.

[37] CEN and Standard Norge. Eurocode 9: Design of aluminium structures - part 1-1: General common rules, 2009.

[38] CEN and Standard Norge. Eurocode 3: Design of steel structures - part 1-8: Design of joints, 2009.

[39] CEN and Standard Norge. Eurocode 5: Design of timber structures - part 1-1: General common rules and rules for buildings, 2010.

[40] K. W. Johansen. Theory of timber connections. *IABSE publications*, 1949.

[41] Jack Porteous and Abdy Kermani. *Structural Timber Design to Eurocode 5*. Wiley-Blackwell, West Sussex, UK, 2nd edition, 2013.

[42] J. W. Harris and H. Stocker. *Handbook of Mathematics and Computational Science*. Springer-Verlag, New York, USA, 1998.

[43] Mitsuo Gen and Runwei Cheng. *Genetic Algorithms and Engineering Optimization*. John Wiley Sons, Inc., 1st edition, 2000.

[44] Aashild Huseby and Marie Eliassen. The digital workflow of parametric structural design - developing grid shells in a nordic climate. masters thesis, NTNU, Trondheim, Norway, June 2018.

[45] Moelven. Standard limtre.
URL: https://www.moelven.com/no/Produkter-og-tjenester/Limtre-og-Kerto/Standard-Limtre/ [Retrieved
2018-06-07].

[46] Dassault Systèmes. Abaqus manual.
URL: https://www.sharcnet.ca/Software/Abaqus610/Documentation/docs/v6.10/books/usb/default.htm?startat=pt06ch25s01alm01.html
[Retrieved 2018-05-01].

[47] Karl Rottmann. *Matematisk Formelsamling*. Spektrum Publishers, Oslo, Norway, 13th edition, 2013.

Appendix A

Acronyms

AEC	Architecture, Engineering and Construction
AM	Additive Manufacturing
BIM	Building Information Model
CAD	Computer Aided Design
CAE	Computer Aided Engineering
CAM	Computer Aided Manufacturing
CNC	Computer Numerical Control
CSDG	Conceptual Structural Design Group at NTNU
DDM	Direct Digital Manufacturing
DMLS	Direct Metal Laser Sintering
EWP	Engineered Wood Product
FDM	Fused Deposition Modelling
FEA	Finite Element Analysis
FEM	Finite Element MEthod
GA	Genetic Algorithms
HSE	Health, Safety and Environment
NC	Numerical Control
RP	Rapid Prototyping

SLM Selective Laser Melting

SLS Selective Laser Sintering

Appendix B

Custom Components for the Parametric Model with C# Script

B.1 Structural Verification and Optimization

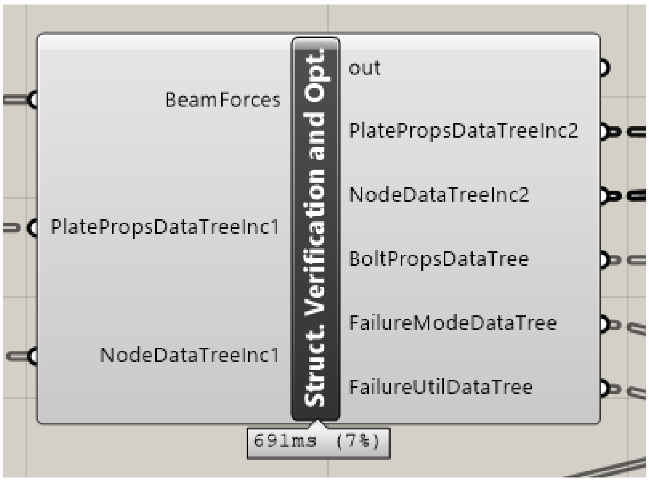


Figure B.1: Structural verification and optimization component

```
private void RunScript(DataTree<double> BeamForces, DataTree<double>
    PlatePropsDataTreeInc1, DataTree<double> NodeDataTreeInc1, ref object
    PlatePropsDataTreeInc2, ref object NodeDataTreeInc2, ref object
```

```
      BoltPropsDataTree, ref object FailureModeDataTree, ref object
      FailureUtilDataTree)
  {
    GH_Path GH_Path_Plate = new GH_Path();
    GH_Path GH_Path_Node = new GH_Path();
    GH_Path GH_Path_BeamForceLC = new GH_Path();
    GH_Path GH_Path_BoltForceLC = new GH_Path();
    GH_Path GH_Path_Bolt = new GH_Path();
    DataTree<int> nBoltsX_ = new DataTree<int>();
    DataTree<int> nBoltsZ_ = new DataTree<int>();
    DataTree<double> SpacingDataTree_ = new DataTree<double>();
    DataTree<double> BoltForces_ = new DataTree<double>();
    DataTree<double> XCoord = new DataTree<double>();
    DataTree<double> ZCoord = new DataTree<double>();
    DataTree<double> BoltPropsDataTree_ = new DataTree<double>();
    DataTree<string> FailureModeDataTree_ = new DataTree<string>();
    DataTree<double> FailureUtilDataTree_ = new DataTree<double>();
    double anglePitch;
    int maxIterationsNBolts = 5, nBoltsIterations;
    int nPlates, nNodes, nLCs, nBoltsX, nBoltsZ, nBolts = 0, nBoltsTemp,
         nBoltsZ_max, counter;
    double utilization, tempUtilization, tempUtilizationPunching, maxUtilization =
         1.00, ratioUtil;
    double forceAngle = 0;
    double boltsize = 0, A_bt, e1 = 0, e2 = 0, p1 = 0, p2 = 0, a1 = 0, a2, a2edit =
         0, a3 = 0, a4, a4edit = 0, d, d0, d_w, eta, Ip_sum, h_eff, N_Ed, V_Ed,
         M_Ed, R_90d;
    double extraHeight, thicknessGussetPlate = 0, t_1, widthTimber, heightTimber,
         thicknessRing, thicknessRingPlate, thicknessRingOutput, ringDiameter;
    double h_boltGroup, w_boltGroup, x_, z_, phi, areaGussetPlate, W_el_GussetPlate;
    double fxBolt, fzBolt, fBolt, fAngleBolt;
    double F1_axRd, F2_axRd, F_axRk, M_yRk, k_90, f_h0k, f_hak, F_vRk_f, F_vRk_g,
         F_vRk_h, MinJohansenPart, n_eff;
    double F_vRd_cutOffBolt, f_vRd_x, f_vRd_z;
    double alpha_d, alpha_b, k_1;
    bool innerBoltX, innerBoltZ;
    double N_1_Ed, M_ip_1_Ed, N_1_Rd, M_ip_1_Rd;

    // -------------- Plate capacity -----------------
    double rho, relativeSlenderness, imperfectionfactor = 0.32, ksi, Phi, lambda_1,
         N_Cr, N_bRd, N_cRd, N_tRd, N_plRd, N_uRd, V_plRd, M_NRd, M_plRd;
    double areaGussetPlate_net, I_weak, I_strong, W_pl_strong, DistToBeamEdge, L_Cr;

    // -------------------------------

    // Material values
    double rho_k = 410; // [kg/m^3] // Must check!
    double k_mod = 0.6;
    double f_c90k = 2.5; // [MPa]
    double f_ukb = 490; // [MPa]
    double fy_alu = 290; // [MPa] // https://www.concast.com/c95500.php
    double fu_alu = 655; // [MPa] // https://www.concast.com/c95500.php
    double kp = 1;
    double alpha_v = 0.6;
    double E_Alu = 110000; // [MPa]
    double G_Alu = E_Alu / (1 + 0.32); // [MPa]
    double epsilon = Math.Sqrt(235 / fy_alu); // MUST BE CHANGED!!
    double kappa = 1.0; // Weakening of wielding

    // Material safety factors according to NS-EN 1993-1-1, NS-EN 1995-1-1 and
         NS-EN 1999-1-1
    // Timber
    double gamma_M_Glulam = 1.15;
    double gamma_M_Connection = 1.3;

    // Alu
    double gamma_M0_Alu = 1.10;
    double gamma_M1_Alu = 1.10;
    double gamma_M2_Alu = 2.0;
    double gamma_M5_Alu = 1.10;

    // Bolt
    double gamma_M2_Bolt = 1.25;

    // Count number of nodes
    nNodes = 0;
    GH_Path_Plate = new GH_Path(0, 0);
    while (PlatePropsDataTreeInc1.PathExists(GH_Path_Plate))
      GH_Path_Plate = new GH_Path(++nNodes, 0); //

    // Count number of load cases
    nLCs = 0;
    GH_Path_BeamForceLC = new GH_Path(0, 0, 0);
    while (BeamForces.PathExists(GH_Path_BeamForceLC))
      GH_Path_BeamForceLC = new GH_Path(0, 0, ++nLCs);

    for (int m = 0; m < nNodes; m++) // Loop over all nodes
    {
      GH_Path_Node = new GH_Path(m);

      // Count number of plates per node
      nPlates = 0;
      GH_Path_Plate = new GH_Path(m, 0);
      while (PlatePropsDataTreeInc1.PathExists(GH_Path_Plate))
        GH_Path_Plate = new GH_Path(m, ++nPlates);

      ringDiameter = NodeDataTreeInc1.Branch(GH_Path_Node)[6];
      thicknessRingOutput = 0;

      for (int n = 0; n < nPlates; n++) // Loop over all plates in the current node
      {
        // Address for output data and input from 'PlatePropsDataTree'
        GH_Path_Plate = new GH_Path(m, n);
        widthTimber = PlatePropsDataTreeInc1.Branch(GH_Path_Plate)[0];
        heightTimber = PlatePropsDataTreeInc1.Branch(GH_Path_Plate)[1];
        counter = 0;
        nBoltsX = 1;
        nBoltsZ = 1;
        utilization = 2;
        nBoltsIterations = 0;
        thicknessRingPlate = 0;
        anglePitch = PlatePropsDataTreeInc1.Branch(GH_Path_Plate)[4];

        while (utilization >= maxUtilization && nBoltsIterations <=
             maxIterationsNBolts)
        {
          nBoltsIterations++;

          // -------------------------------------------------------
          // Increase number of bolts if utilization is insufficient:
          // -------------------------------------------------------
          switch (counter)
          {
            case 0:
              nBoltsZ++;
              break;
            case 1:
              nBoltsTemp = nBoltsZ;
              nBoltsZ = nBoltsX;
              nBoltsX = nBoltsTemp;
              break;
            case 2:
              nBoltsZ++;
              break;
          }
          counter = (counter + 1) % 3;

          // -------------------------------------------------------
          // Change thickness of plate until utilization is insufficient:
          // -------------------------------------------------------

          // Set starting thickness of gusset plate to minimum for a thick steel
               plate (t >= d) - the step value (For example start value = 5 mm and
               step value = 1 mm neccessitates thicknessGussetPlate to be equal to 4
               mm!!)
          thicknessGussetPlate = 10; // [mm]

          while (utilization >= maxUtilization && thicknessGussetPlate <= 30)
          {
            // Increase thickness of gusset plate:
```

```
          thicknessGussetPlate += 2;

          areaGussetPlate = heightTimber * thicknessGussetPlate;
          W_el_GussetPlate = 1d / 6d * thicknessGussetPlate *
               Math.Pow(heightTimber, 2);
          W_pl_strong = 1d / 4d * thicknessGussetPlate * Math.Pow(heightTimber, 2);
          I_strong = 1d / 12d * thicknessGussetPlate * Math.Pow(heightTimber, 3);
          I_weak = 1d / 12d * heightTimber * Math.Pow(thicknessGussetPlate, 3);

          // Set starting value for boltsize
          boltsize = 10; // Radius [mm]

          while ((utilization >= maxUtilization && boltsize <=
               thicknessGussetPlate - 2) && boltsize <= 22)
          {
            // --------------------------------------------------------
            // Change bolt size if utilization is insufficient:
            // --------------------------------------------------------
            boltsize += 2;

            d = boltsize;
            d0 = d + 1;
            d_w = 3 * d;

            if (d == 12)
              A_bt = 84;
            else if (d == 14)
              A_bt = 115;
            else if (d == 16)
              A_bt = 157;
            else if (d == 18)
              A_bt = 192;
            else if (d == 20)
              A_bt = 245;
            else if (d == 22)
              A_bt = 303;
            else if (d == 24)
              A_bt = 353;
            else
              A_bt = 0;

            eta = heightTimber / d0;

            // --------------------------------------------------
            // Calculate neccessary spacing acording to EC3/EC9 and EC5 given beam
                 forces and bolt size
            // --------------------------------------------------

            // Reset variables from last iteration:
            a1 = 0;
            a2 = 0;
            a2edit = 0;
            a3 = 0;
            a4 = 0;
            a4edit = 0;

            // EC3 / EC9:
            e1 = 1.2 * d0;
            e2 = 1.2 * d0;
            p1 = 2.2 * d0;
            p2 = 2.4 * d0;

            // EC5:
            a2 = 4 * d;

            for (int q = 0; q < nLCs; q++)
            {
              GH_Path_BeamForceLC = new GH_Path(m, n, q);
              forceAngle = (Math.Atan2(BeamForces.Branch(GH_Path_BeamForceLC)[1],
                   BeamForces.Branch(GH_Path_BeamForceLC)[0]) + 4 * Math.PI) % (2 *
                   Math.PI);

              a1 = Math.Max(a1, (4 + Math.Abs(Math.Cos(forceAngle))) * d);

              if ((forceAngle >= 0 && forceAngle <= Math.PI / 2) || (forceAngle >=
                   270 / 180 * Math.PI && forceAngle <= 2 * Math.PI))
                a3 = Math.Max(a3, Math.Max(7 * d, 80));
              else if (forceAngle >= Math.PI / 2 && forceAngle < 150 / 180 *
                   Math.PI)
                a3 = Math.Max(a3, (1 + 6 * Math.Sin(forceAngle)) * d);
              else if (forceAngle >= 150 / 180 * Math.PI && forceAngle < 210 / 180
                   * Math.PI)
                a3 = Math.Max(a3, 4 * d);
              else if (forceAngle >= 210 / 180 * Math.PI && forceAngle <= 270 /
                   180 * Math.PI)
                a3 = Math.Max(a3, (1 + 6 * Math.Abs(Math.Sin(forceAngle))) * d);

              // Although the idle edge distance can be smaller than for the
                   strained, it is advantageous calculationwise to keep them equal
                   to avoid eccentric load on the bolt group!
              a4 = Math.Max(a4, Math.Max((2 + 2 * Math.Abs(Math.Sin(forceAngle)))
                   * d, 3 * d));
            }

            // ------------------------------------------------
            // Shear resistance per shear plane (assume bolt class 8.8) (EC3-1-8
                 Table 3.4)
            // ------------------------------------------------
            F_vRd_cutOffBolt = (alpha_v * f_ukb * Math.PI * Math.Pow(d0 / 2, 2)) /
                 gamma_M2_Bolt;
```

```

// ----------------------------------------------
// Check if the number of bolts demands taller beams than available:
// ----------------------------------------------

nBoltsZ_max = (int) Math.Floor((heightTimber - 2 * a4) / a2) + 1;

if (nBoltsZ <= nBoltsZ_max)
{
  // We have SUFFICIENT beam height for the current configuration!

  // -----------------------------------------------------
  // Loop over different spacing configurations:
  //
  // phi describes how extra space is distributed between a2 and a4.
  // phi = 1 means distance from edge bolts to the edge is at minimum of allowed distance!
  // phi = 0 implies minimum distance between the bolts in z-direction!
  // -----------------------------------------------------

  extraHeight = heightTimber - (2 * a4 + (nBoltsZ - 1) * a2);

  phi = -0.25;
  while (utilization >= maxUtilization && phi <= 0.75) // Loop over different distributions of spacing
  {
    phi += 0.25;
    if (nBoltsZ != 1)
    {
      a2edit = a2 + phi * extraHeight / (nBoltsZ - 1);
      a4edit = a4 + (1 - phi) * extraHeight / 2;
    }
    else
    {
      a2edit = a2;
      a4edit = a4 + extraHeight / 2;
    }

    // -------------------------------------------------------
    // Calculate the corresponding bolt forces for all LCs:
    // -------------------------------------------------------

    // Total height and length of bolt group:
    h_boltGroup = (nBoltsZ - 1) * a2edit;
    w_boltGroup = (nBoltsX - 1) * a1;

    Ip_sum = 0;

    XCoord = new DataTree<double>();
    ZCoord = new DataTree<double>();
```

```

                nBolts = nBoltsX * nBoltsZ;

                for (int i = 0; i < nBoltsZ; i++)
                {
                  for (int j = 0; j < nBoltsX; j++)
                  {
                    // Find x-coordinate
                    if (nBoltsX == 1)
                      x_ = 0;
                    else
                      x_ = w_boltGroup * ((double) j) / ((double) (nBoltsX - 1)) -
                           w_boltGroup / 2;

                    // Find z-coordinate
                    if (nBoltsZ == 1)
                      z_ = 0;
                    else
                      z_ = -h_boltGroup * ((double) (i)) / ((double) (nBoltsZ - 1))
                           + h_boltGroup / 2;

                    // Store coordinates in a DataTree
                    XCoord.Add(x_, GH_Path_Plate);
                    ZCoord.Add(z_, GH_Path_Plate);

                    // Add contribution to Ip
                    Ip_sum = Ip_sum + Math.Pow(x_, 2) + Math.Pow(z_, 2);
                  }
                }
                BoltForces_ = new DataTree<double>();

                for (int k = 0; k < nBolts; k++)
                {
                  // Loop over all load cases:
                  for (int q = 0; q < nLCs; q++)
                  {
                    GH_Path_BoltForceLC = new GH_Path(q, k);
                    GH_Path_BeamForceLC = new GH_Path(m, n, q);

                    fxBolt = 1000 * BeamForces.Branch(GH_Path_BeamForceLC)[0] /
                         nBolts + 1000000 *
                         BeamForces.Branch(GH_Path_BeamForceLC)[2] *
                         ZCoord.Branch(GH_Path_Plate)[k] / Ip_sum;
                    fzBolt = 1000 * BeamForces.Branch(GH_Path_BeamForceLC)[1] /
                         nBolts + 1000000 *
                         BeamForces.Branch(GH_Path_BeamForceLC)[2] *
                         XCoord.Branch(GH_Path_Plate)[k] / Ip_sum;
                    fBolt = Math.Sqrt(Math.Pow(fxBolt, 2) + Math.Pow(fzBolt, 2));

                    if (fxBolt != 0)
```

```
      fAngleBolt = (Math.Atan(-fzBolt / fxBolt) * 180 / Math.PI + 360) % (360);
    else
      fAngleBolt = Math.PI / 2;

    BoltForces_.Add(fxBolt, GH_Path_BoltForceLC);
    BoltForces_.Add(fzBolt, GH_Path_BoltForceLC);
    BoltForces_.Add(fBolt, GH_Path_BoltForceLC);
    BoltForces_.Add(fAngleBolt, GH_Path_BoltForceLC);
  }
}
areaGussetPlate_net = areaGussetPlate - nBoltsZ * d0 * thicknessGussetPlate;
tempUtilization = 0;

// --------------------------------------
// Verify the Timber-Aluminium Connection by calculating with Johansen's Approach - part 1 (independant of LCs)
// --------------------------------------

// Bending moment at yield, 8.5.1.1, Equation (8.30)
M_yRk = 0.3 * f_ukb * Math.Pow(d, 2.6);

// 8.5.1.1, Equation (8.33) (For soft wood)
// d in [mm]
k_90 = 1.35 + 0.015 * d;

// 8.5.1.1, Equation (8.32)
// rho_k in [kg/m^3]
f_h0k = 0.082 * (1 - 0.01 * d) * rho_k;

t_1 = (widthTimber - thicknessGussetPlate) / 2;

// Calculating rope effect from threaded bolts
// (Given by the maximum of the axial capacity of the bolt and the yield strength of the timber under compression from the washer)
F1_axRd = 0.9 * f_ukb * (1 / gamma_M2_Alu) * A_bt;
F2_axRd = 3 * f_c90k * (Math.PI / 4) * (Math.Pow(d_w, 2) - Math.Pow(d0, 2)) * (k_mod / gamma_M_Glulam);//NB: d+1 mm
F_axRk = Math.Min(F2_axRd, F1_axRd) * (gamma_M_Glulam / k_mod);

for (int q = 0; q < nLCs; q++) // Loop over all loadcases
{
  GH_Path_BeamForceLC = new GH_Path(m, n, q);
  for (int k = 0; k < nBolts; k++) // Loop over all bolts
  {
    GH_Path_BoltForceLC = new GH_Path(q, k);

    // Get correct bolt force in simpler variable names
    fxBolt = BoltForces_.Branch(GH_Path_BoltForceLC)[0];
```

```
                fzBolt = BoltForces_.Branch(GH_Path_BoltForceLC)[1];
                fBolt = BoltForces_.Branch(GH_Path_BoltForceLC)[2];
                fAngleBolt = BoltForces_.Branch(GH_Path_BoltForceLC)[3];

                // --------------------------------------
                // Verify the Timber-Aluminium Connection by calculating with
                     Johansen's Approach - part 2 (dependant of LCs)
                // --------------------------------------

                // 8.5.1.1, Equation (8.31)
                // Calculating f_hak for every bolt with corresponding angle
                     for the bolt force
                f_hak = f_h0k / (k_90 * Math.Pow(Math.Sin(-fAngleBolt * Math.PI
                     / 180), 2) + Math.Pow(Math.Cos(-fAngleBolt * Math.PI /
                     180), 2));

                // 8.2.3 For a thick steel plate in double shear, see mode f,
                     g, and h:
                F_vRk_f = 2 * f_hak * t_1 * d;
                F_vRk_g = 2 * f_hak * t_1 * d * (Math.Sqrt(2 + (4 * M_yRk /
                     (f_hak * Math.Pow(t_1, 2) * d))) - 1) + F_axRk / 4;
                F_vRk_h = 2 * 2.3 * Math.Sqrt(M_yRk * f_hak * d) + F_axRk / 4;

                // 8.2.2 (2) Verify that the Johansen part of the equations is
                     larger than 25% of the rope effect contribution
                MinJohansenPart = Math.Min(2 * f_hak * t_1 * d * (Math.Sqrt(2 +
                     (4 * M_yRk / (f_hak * Math.Pow(t_1, 2) * d))) - 1), 2 *
                     2.3 * Math.Sqrt(M_yRk * f_hak * d));

                ratioUtil = Math.Max(F_axRk / MinJohansenPart, Math.Abs(fBolt /
                     (k_mod / gamma_M_Glulam * Math.Min(Math.Min(F_vRk_f,
                     F_vRk_g), F_vRk_h))));

                if (ratioUtil > tempUtilization)
                {
                  FailureModeDataTree_.RemovePath(GH_Path_Plate);
                  FailureModeDataTree_.Add("A", GH_Path_Plate);
                  tempUtilization = ratioUtil;
                }

                // ------------------------------------
                // Verify Shear Resistance for Bolt:
                // ------------------------------------

                ratioUtil = Math.Abs(fBolt / (2 * F_vRd_cutOffBolt));
                if (ratioUtil > tempUtilization)
                {
                  FailureModeDataTree_.RemovePath(GH_Path_Plate);
                  FailureModeDataTree_.Add("C2", GH_Path_Plate);
                  tempUtilization = ratioUtil;
```

```
}
tempUtilization = Math.Max(tempUtilization, fBolt / (2 *
    F_vRd_cutOffBolt));

// ---------------------------------
// Calculate Bearing Resistance for Bolt
// ---------------------------------

//checking inner/outer bolt
if ((k - 1) % nBoltsX == 0)
  innerBoltX = false;
else
  innerBoltX = true;

if ((k >= nBolts - nBoltsX) || (k <= nBoltsX))
  innerBoltZ = false;
else
  innerBoltZ = true;

// checking x-direction
// finding alpha_b (in the direction of load transfer)
if (innerBoltX == true)
  alpha_d = (a1 / (3 * d0)) - (1 / 4);
else
  alpha_d = e1 / 3 * d0;
alpha_b = Math.Min(Math.Min(alpha_d, (f_ukb / fu_alu)), 1.0);

// finding k_1 (perpendicular to the direction of load transfer)
if (innerBoltZ == true)
  k_1 = Math.Min(Math.Min(2.8 * (a4edit / d0) - 1.7, 1.4 *
      (a2edit / d0)), 2.5);
else
  k_1 = Math.Min(1.4 * (a2edit / d0), 2.5);
f_vRd_x = (k_1 * alpha_b * fu_alu * d * thicknessGussetPlate) /
    gamma_M2_Alu;

if (Math.Abs(fxBolt / f_vRd_x) > tempUtilization)
{
  FailureModeDataTree_.RemovePath(GH_Path_Plate);
  FailureModeDataTree_.Add("C1X", GH_Path_Plate);
  tempUtilization = Math.Abs(fxBolt / f_vRd_x);
}

// checking z-direction
// finding alpha_b (in the direction of load transfer)
if (innerBoltZ == true){
  alpha_d = (a2edit / (3 * d0)) - 1 / 4;
}
else{
  alpha_d = a4edit / 3 * d0;
```

```
          }
          alpha_b = Math.Min(Math.Min(alpha_d, (f_ukb / fu_alu)), 1.0);

          // finding k_1 (perpendicular to the direction of load transfer)
          if (innerBoltX == true){
            k_1 = Math.Min(Math.Min(2.8 * (e1 / d0) - 1.7, 1.4 * (a1 /
                d0)) - 1.7, 2.5);
          }
          else{
            k_1 = Math.Min(1.4 * (a1 / d0) - 1.7, 2.5);
          }
          f_vRd_z = (k_1 * alpha_b * fu_alu * d * thicknessGussetPlate) /
              gamma_M2_Alu;

          ratioUtil = Math.Abs(fzBolt / f_vRd_z);
          if (ratioUtil > tempUtilization)
          {
            FailureModeDataTree_.RemovePath(GH_Path_Plate);
            FailureModeDataTree_.Add("C1Z", GH_Path_Plate);
            tempUtilization = ratioUtil;
          }
        }

        // ------------------------------------
        // Verify the Timber-Aluminium Connection by calculating with
            Johansen's Approach - Looking at the effect of a row of bolts
        // ------------------------------------

        double F_vRk_single, F_vRd_single, F_vRd_group, F_vEd_group,
            F_vEd_group1, F_vEd_group2;

        if (nBoltsX > 1) //
        {
          // Calculates effective number of bolts for a single row
          n_eff = Math.Min(Math.Pow(nBoltsX, 0.9) * Math.Pow(a1 / (13 *
              d), 0.25), nBoltsX);

          // Calculate Force from top and bottom row and find the
              maximum: F_vEd_group
          F_vEd_group1 = 0;
          F_vEd_group2 = 0;

          // Assumes that the maximum bolt group force is either the top
              row or the bottom row. Finds the maximum of the two from
              all the load cases:
          for (int k = 0; k < nBoltsX; k++)
          {
            GH_Path_BoltForceLC = new GH_Path(q, k);
            F_vEd_group1 = F_vEd_group1 +
                BoltForces_.Branch(GH_Path_BoltForceLC)[0];

            GH_Path_BoltForceLC = new GH_Path(q, (int) nBoltsX * (int)
                nBoltsZ - k - 1);
            F_vEd_group2 = F_vEd_group2 +
                BoltForces_.Branch(GH_Path_BoltForceLC)[0];
          }
          F_vEd_group = Math.Max(Math.Abs(F_vEd_group1),
              Math.Abs(F_vEd_group2));

          // Calculate capacity from group action EC5 8.2.3(3) Equation
              8.11
          F_vRk_f = f_h0k * t_1 * d;
          F_vRk_g = f_h0k * t_1 * d * (Math.Sqrt(2 + (4 * M_yRk / (f_h0k
              * Math.Pow(t_1, 2) * d))) - 1) + F_axRk / 4;
          F_vRk_h = 2.3 * Math.Sqrt(M_yRk * f_h0k * d) + F_axRk / 4;
          F_vRk_single = Math.Min(F_vRk_f, Math.Min(F_vRk_g, F_vRk_h));
          F_vRd_single = k_mod / gamma_M_Glulam * F_vRk_single;
          F_vRd_group = n_eff * F_vRd_single;

          // Calculates the ratio between dimensioning force and capacity
              for group action
          ratioUtil = Math.Abs(F_vEd_group / F_vRd_group);
          if (ratioUtil > tempUtilization)
          {
            FailureModeDataTree_.RemovePath(GH_Path_Plate);
            FailureModeDataTree_.Add("B", GH_Path_Plate);
            tempUtilization = ratioUtil;
          }
        }

        // -----------------------------------------------
        // Calculate splitting (EC5-1-1, 8.1.4)
        // -----------------------------------------------

        // Calculate effective height for splitting of timber
        h_eff = heightTimber - a2edit;

        // dimensioning shear forces
        N_Ed = BeamForces.Branch(GH_Path_BeamForceLC)[0] * 1000; //
            Converts from kN to N
        V_Ed = BeamForces.Branch(GH_Path_BeamForceLC)[1] * 1000; //
            Converts from kN to N
        M_Ed = BeamForces.Branch(GH_Path_BeamForceLC)[2] * 1000; //
            Converts from kN to N

        // EC5 8.1.4 (2) Equation (8.4)
        R_90d = k_mod / gamma_M_Connection * 14 * widthTimber *
            Math.Sqrt(h_eff / (1 - h_eff / heightTimber));

        ratioUtil = Math.Abs(V_Ed / R_90d);
        if (ratioUtil > tempUtilization)
        {
          FailureModeDataTree_.RemovePath(GH_Path_Plate);
          FailureModeDataTree_.Add("D", GH_Path_Plate);
          tempUtilization = ratioUtil;
        }

        // -----------------------------------------------
        // Calculate gusset plate (Use EC3-1-1, chapter 6 and EC3-1-8,
            chapter 3)
        // -----------------------------------------------

        // EC3-1-1 6.2.3 Tension

        N_plRd = areaGussetPlate * fy_alu / gamma_M0_Alu;
        N_uRd = 0.9 * areaGussetPlate_net * fu_alu / gamma_M2_Alu;
        N_tRd = Math.Min(N_plRd, N_uRd);

        if (N_Ed > 0)
        {
          ratioUtil = Math.Abs(N_Ed / N_tRd);
          if (ratioUtil > tempUtilization)
          {
            FailureModeDataTree_.RemovePath(GH_Path_Plate);
            FailureModeDataTree_.Add("F1", GH_Path_Plate);
            tempUtilization = ratioUtil;
          }
        }

        // EC3-1-1 6.2.4 Compression

        N_cRd = areaGussetPlate * fy_alu / gamma_M0_Alu;
        if (N_Ed < 0)
        {
          ratioUtil = Math.Abs(N_Ed / N_cRd);
          if (ratioUtil > tempUtilization)
          {
            FailureModeDataTree_.RemovePath(GH_Path_Plate);
            FailureModeDataTree_.Add("F2", GH_Path_Plate);
            tempUtilization = ratioUtil;
          }
        }

        // EC3-1-1 6.2.6 Shear

        V_plRd = areaGussetPlate * (fy_alu / Math.Sqrt(3)) / gamma_M0_Alu;
        ratioUtil = Math.Abs(V_Ed / V_plRd);

        if (ratioUtil > tempUtilization)
        {
          FailureModeDataTree_.RemovePath(GH_Path_Plate);
          FailureModeDataTree_.Add("F3", GH_Path_Plate);
          tempUtilization = ratioUtil;
        }

        // EC3-1-1 6.2.10 Combination of bending moment, axial forces and
            shear

        if (V_Ed > 0.5 * V_plRd)
          rho = Math.Pow(2 * V_Ed / V_plRd - 1, 2);
        else
          rho = 0;

        M_plRd = W_pl_strong * (1 - rho) * fy_alu / gamma_M0_Alu;
        M_NRd = M_plRd * (1 - Math.Pow(N_Ed / N_plRd, 2)); // NB! NB!
            This is for cross section WITHOUT bolt holes. Is there a way
            to calculate WITH bolt holes??

        ratioUtil = Math.Abs(M_Ed / M_NRd);

        if (ratioUtil > tempUtilization)
        {
          FailureModeDataTree_.RemovePath(GH_Path_Plate);
          FailureModeDataTree_.Add("F4", GH_Path_Plate);
          tempUtilization = ratioUtil;
        }

        // EC3-1-1 6.3.1 Column type buckling

        if (N_Ed < 0)
        {
          DistToBeamEdge =
              PlatePropsDataTreeInc1.Branch(GH_Path_Plate)[5] *
              Math.Cos(anglePitch) - heightTimber / 1000 / 2 *
              Math.Sin(anglePitch);
          L_Cr = 1 * (DistToBeamEdge - ringDiameter / 1000 / 2) * 1000;
          lambda_1 = Math.PI * Math.Sqrt(E_Alu / fy_alu);
          N_Cr = Math.Pow(Math.PI, 2) * E_Alu * I_weak / Math.Pow(L_Cr,
              2);
          relativeSlenderness = Math.Sqrt(areaGussetPlate * fy_alu /
              N_Cr);
          Phi = 0.5 * (1 + imperfectionfactor * (relativeSlenderness -
              0.00) + Math.Pow(relativeSlenderness, 2));
          ksi = Math.Min(1 / (Phi + Math.Sqrt(Math.Pow(Phi, 2) -
              Math.Pow(relativeSlenderness, 2))), 1);
          N_bRd = kappa * ksi * areaGussetPlate * fy_alu / gamma_M1_Alu;

          ratioUtil = Math.Abs(N_Ed / N_bRd);
          if (ratioUtil > tempUtilization)
          {
```

```
          FailureModeDataTree_.RemovePath(GH_Path_Plate);
          FailureModeDataTree_.Add("F5", GH_Path_Plate);
          tempUtilization = ratioUtil;
        }
      }

      // ------------------------------------------------
      // Calculate shear punching (Use EC3-1-8, 7.4)
      // ------------------------------------------------

      tempUtilizationPunching = 10;
      thicknessRing = 0.4 * thicknessGussetPlate - 1;
      while (tempUtilizationPunching >= maxUtilization && thicknessRing
           <= ringDiameter * 1000 / 2 - 1)
      {
        tempUtilizationPunching = 0;
        thicknessRing += 1;

        // Get data from DataTrees
        N_1_Ed = BeamForces.Branch(GH_Path_BeamForceLC)[0] * 1000; //
             From kN to N
        M_ip_1_Ed = BeamForces.Branch(GH_Path_BeamForceLC)[2] *
             1000000; // From kN*m to N*mm

        // Check for ring face failure, EC3 Table 7.4 - case #2

        N_1_Rd = 5 * kp * fy_alu * Math.Pow(thicknessRing, 2) * (1 +
             0.25 * eta) / gamma_M5_Alu;
        M_ip_1_Rd = heightTimber * N_1_Rd;

        tempUtilizationPunching = Math.Max(tempUtilizationPunching,
             N_1_Ed / N_1_Rd + Math.Pow(M_ip_1_Ed / M_ip_1_Rd, 2));

        // Check for punching shear failure in the ring
        tempUtilizationPunching = Math.Max(tempUtilizationPunching,
             Math.Abs(((N_1_Ed / areaGussetPlate + M_ip_1_Ed /
             W_el_GussetPlate) * thicknessGussetPlate)) / (2 *
             thicknessRing * (fy_alu / Math.Sqrt(3)) / gamma_M5_Alu));
      }
      ratioUtil = Math.Max(tempUtilizationPunching, tempUtilization);
      if (ratioUtil > tempUtilization)
      {
        FailureModeDataTree_.RemovePath(GH_Path_Plate);
        FailureModeDataTree_.Add("G", GH_Path_Plate);
        tempUtilization = ratioUtil;
      }
      thicknessRingPlate = thicknessRing;

      // Choose max utilization to verify connection
      utilization = tempUtilization;
```

```

                  }
                }
              }
            }
          }
        }

        for (int k = 0; k < nBolts; k++)
        {
          GH_Path_Bolt = new GH_Path(m, n, k);
          BoltPropsDataTree_.Add(XCoord.Branch(GH_Path_Plate)[k], GH_Path_Bolt);
          BoltPropsDataTree_.Add(ZCoord.Branch(GH_Path_Plate)[k], GH_Path_Bolt);
        }

        thicknessRingOutput = Math.Max(thicknessRingOutput, thicknessRingPlate);

        PlatePropsDataTreeInc1.Add(nBoltsX, GH_Path_Plate);
        PlatePropsDataTreeInc1.Add(nBoltsZ, GH_Path_Plate);
        PlatePropsDataTreeInc1.Add(thicknessGussetPlate, GH_Path_Plate);
        PlatePropsDataTreeInc1.Add(boltsize, GH_Path_Plate);
        PlatePropsDataTreeInc1.Add(utilization, GH_Path_Plate);
        FailureUtilDataTree_.Add(utilization, GH_Path_Plate);
        PlatePropsDataTreeInc1.Add(a1, GH_Path_Plate);
        PlatePropsDataTreeInc1.Add(a2edit, GH_Path_Plate);
        PlatePropsDataTreeInc1.Add(a3, GH_Path_Plate);
        PlatePropsDataTreeInc1.Add(a4edit, GH_Path_Plate);
        PlatePropsDataTreeInc1.Add(e1, GH_Path_Plate);
        PlatePropsDataTreeInc1.Add(e2, GH_Path_Plate);
        PlatePropsDataTreeInc1.Add(p1, GH_Path_Plate);
        PlatePropsDataTreeInc1.Add(p2, GH_Path_Plate);
      }
      NodeDataTreeInc1.Add(thicknessRingOutput, GH_Path_Node);
    }
    PlatePropsDataTreeInc2 = PlatePropsDataTreeInc1;
    BoltPropsDataTree = BoltPropsDataTree_;
    NodeDataTreeInc2 = NodeDataTreeInc1;
    FailureModeDataTree = FailureModeDataTree_;
    FailureUtilDataTree = FailureUtilDataTree_;
  }
```

B.2 Necessary Length To Timber Edge

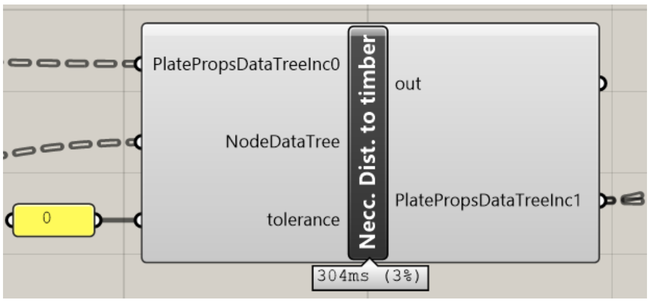


Figure B.2: Component for calculating necessary distance to timber

```
private void RunScript(DataTree<double> PlatePropsDataTreeInc0, DataTree<double>
     NodeDataTree, double tolerance, ref object PlatePropsDataTreeInc1)
{
    List<double> lengths_ = new List<double>();
    List<double> lengths_2 = new List<double>();
    List<double> WidthOfTimber = new List<double>();
    List<double> AnglesYaw = new List<double>();
    List<double> AnglesPitch = new List<double>();
    List<double> heights = new List<double>();
    List<int> Indicies = new List<int>();
    GH_Path GH_Path_Plate = new GH_Path();
    GH_Path GH_Path_Node = new GH_Path();
    double angle_1 = 0;
    double angle_2 = 0;
    double l11, l12, l2, l3, w1, w2, w3, outerRadius, angleRoll, anglePitch, h;
    int iTemp = 0;
    int index1 = 0;
    int index2 = 0;
    int index3 = 0;
    int nNodes = 0;
    int nPlates = 0;

    // Find number of nodes by testing how many data paths that exist:
    nNodes = 0;
    GH_Path_Plate = new GH_Path(0, 0);
    while (PlatePropsDataTreeInc0.PathExists(GH_Path_Plate))
      GH_Path_Plate = new GH_Path(++nNodes, 0);

    // Loop over all nodes:
    for (int m = 0; m < nNodes; m++)
    {
      GH_Path_Node = new GH_Path(m);

      // Find number of plates by testing how many data paths that exist:
      nPlates = 0;
      GH_Path_Plate = new GH_Path(m, 0);
      while (PlatePropsDataTreeInc0.PathExists(GH_Path_Plate))
        GH_Path_Plate = new GH_Path(m, ++nPlates);

      // Collect the outer radius of the ring (in meters)
      outerRadius = NodeDataTree.Branch(GH_Path_Node)[6] / 2;

      // Reset list of width, height, yaw and pitch for the current plate:
      WidthOfTimber = new List<double>();
      AnglesYaw = new List<double>();
      AnglesPitch = new List<double>();
      heights = new List<double>();

      // Prepare lists for the current number of plates for the current node:
      lengths_ = new List<double>();
      lengths_ = Enumerable.Repeat(0d, nPlates).ToList();
      lengths_2 = new List<double>();
      lengths_2 = Enumerable.Repeat(0d, nPlates).ToList();

      // Collect the width of timber for all plates by looping over all plates:
      for (int n = 0; n < nPlates; n++)
      {
        // Define the data path where data will be collected from or stored:
        GH_Path_Plate = new GH_Path(m, n);

        // Collect width and height of timber:
        w1 = PlatePropsDataTreeInc0.Branch(GH_Path_Plate)[0] / 1000;
        h = PlatePropsDataTreeInc0.Branch(GH_Path_Plate)[1] / 1000;

        // Collect roll of the plate:
        angleRoll = PlatePropsDataTreeInc0.Branch(GH_Path_Plate)[2];

        // Collect yaw and pitch of the plate and assign to a list so that its
             position in the list corresponds to the indexing in the data trees:
        AnglesYaw.Add(PlatePropsDataTreeInc0.Branch(GH_Path_Plate)[3]);
        AnglesPitch.Add(PlatePropsDataTreeInc0.Branch(GH_Path_Plate)[4]);

        // Calculate effective height and width of a
        heights.Add(h * Math.Abs(Math.Cos(angleRoll)) + w1 *
             Math.Abs(Math.Sin(angleRoll)));
        WidthOfTimber.Add(w1 * Math.Abs(Math.Cos(angleRoll)) + h *
             Math.Abs(Math.Sin(angleRoll)));
      }


      // Sort AnglesYaw to fix if the bars are in the wrong order clockwise using
           an insertion sort algorithm.
      Indicies = new List<int>();
      Indicies = Enumerable.Range(0, nPlates).ToList();
      iTemp = 0;

      for (int n = 0; n < nPlates - 1; n++)
      {
        for (int k = n + 1; k < nPlates; k++)
        {
          if (AnglesYaw[Indicies[n]] > AnglesYaw[Indicies[k]])
          {
            iTemp = Indicies[k];
            Indicies[k] = Indicies[n];
            Indicies[n] = iTemp;
          }
        }
      }

      // Find lengths for all plates by looping over all plates:
      for (int n = 0; n < nPlates; n++)
      {
        // Get the indicies from the sorted AnglesYaw. This will be used to fetch
             beam data from adjacent beams when checking minimum distance.
        index1 = Indicies[n];
        index2 = Indicies[(n - 1 + nPlates) % nPlates];
        index3 = Indicies[(n + 1) % nPlates];

        // Find half of the width of the chosen beam and bordering beams (plus
             tolerance) w1, w2 and w3:
        w1 = WidthOfTimber[index1] / 2 + tolerance;
        w2 = WidthOfTimber[index2] / 2 + tolerance;
        w3 = WidthOfTimber[index3] / 2 + tolerance;

        // Get height and pitch for current beam
        h = heights[index1];
        anglePitch = AnglesPitch[index1];

        // Find the angles to adjacent bars (1 = counter clockwise, 2 = clockwise):
        angle_1 = Math.Abs((Math.Abs(AnglesYaw[index1] - AnglesYaw[index2]) + 4 *
             Math.PI) % (2 * Math.PI));
        angle_1 = Math.Min(angle_1, 2 * Math.PI - angle_1);
        angle_2 = Math.Abs((Math.Abs(AnglesYaw[index1] - AnglesYaw[index3]) + 4 *
             Math.PI) % (2 * Math.PI));
        angle_2 = Math.Min(angle_2, 2 * Math.PI - angle_2);

        // Find temporary minimum distance to timber edge in plane
        l11 = (w2 + w1 * Math.Cos(angle_1)) / Math.Sin(angle_1);
        l2 = (w1 + w2 * Math.Cos(angle_1)) / Math.Sin(angle_1);
        l12 = (w3 + w1 * Math.Cos(angle_2)) / Math.Sin(angle_2);
        l3 = (w1 + w3 * Math.Cos(angle_2)) / Math.Sin(angle_2);

        // If one of the temporary minimum distances is smaller than the radius, check with second set of equations.
        if (l11 < outerRadius)
          l11 = outerRadius;
        else if (l2 < outerRadius)
        {
          l11 = w2 * Math.Sin(angle_1) + outerRadius * Math.Cos(angle_1);
          if (l11 < outerRadius) // If second set also fails, set the distance to outer radius.
            l11 = outerRadius;
        }
        if (l12 < outerRadius)
          l12 = outerRadius;
        else if (l3 < outerRadius)
        {
          l12 = w3 * Math.Sin(angle_2) + outerRadius * Math.Cos(angle_2);
          if (l12 < outerRadius) // If second set also fails, set the distance to outer radius.
            l12 = outerRadius;
        }

        // Find the corresponding lengths and minmize necessary length:
        lengths_[index1] = Math.Max(l11, l12);

        // Add the lengthening effect of pitching the plate
        lengths_2[index1] = (lengths_[index1] + h / 2 * Math.Abs(Math.Sin(anglePitch)) / Math.Abs(Math.Cos(anglePitch));
      }

      for (int n = 0; n < nPlates; n++) // Append the length to the Plate Properties Data Tree
      {
        GH_Path_Plate = new GH_Path(m, n);
        PlatePropsDataTreeInc0.Add(lengths_2[n], GH_Path_Plate);
      }
    }
    PlatePropsDataTreeInc1 = PlatePropsDataTreeInc0;
}
```

B.3 Minimum Cylinder Diameter

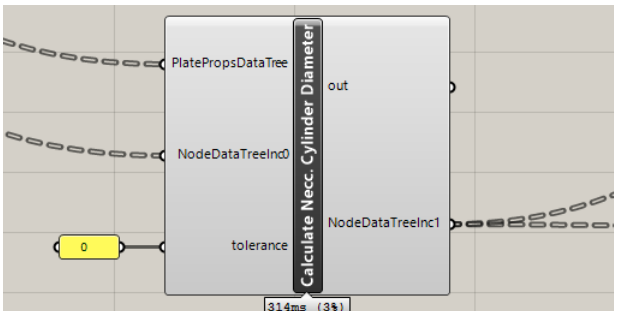


Figure B.3: Component for calculating minimum cylinder diameter.

```
private void RunScript(DataTree<double> PlatePropsDataTree, DataTree<double>
     NodeDataTreeInc0, double tolerance, ref object NodeDataTreeInc1)
  {
    GH_Path GH_Path_Plate = new GH_Path();
    GH_Path GH_Path_Node = new GH_Path();

    List<double> neccWidthForPlate = new List<double>();
    List<double> anglesYaw = new List<double>();
    List<double> anglesPitch = new List<double>();
    List<double> heights = new List<double>();

    List<int> Indicies = new List<int>();

    int nNodes, nPlates, iTemp, index1, index2, index3;

    double diameter, anglePitch, angleRoll, heightOfPlate,
         thicknessOfPlateConservative, newWidthOfPlate, newHeight, w1, w2, w3,
         angle_1, angle_2, psi_1, psi_2, r_1, r_2;

    // Find number of nodes by testing how many data paths that exist:
    nNodes = 0;
    GH_Path_Node = new GH_Path(0);
    while (NodeDataTreeInc0.PathExists(GH_Path_Node))
      GH_Path_Node = new GH_Path(++nNodes);

    // Loop over all nodes:
    for (int m = 0; m < nNodes; m++)
    {
      GH_Path_Node = new GH_Path(m);

      // Find number of plates by testing how many data paths that exist:
      nPlates = 0;
      GH_Path_Plate = new GH_Path(m, 0);
      while (PlatePropsDataTree.PathExists(GH_Path_Plate))
        GH_Path_Plate = new GH_Path(m, ++nPlates);

      // Reset lists for width, yaw, pitch and height:
      neccWidthForPlate = new List<double>();
      anglesYaw = new List<double>();
      anglesPitch = new List<double>();
      heights = new List<double>();

      for (int n = 0; n < nPlates; n++)
      {
        // Define the data path to the current plate
        GH_Path_Plate = new GH_Path(m, n);

        // Fetch pitch, roll and height of the current plate from the Plate
             Properties Data Tree for the relevant gusset plate
        anglePitch = PlatePropsDataTree.Branch(GH_Path_Plate)[4];
        angleRoll = PlatePropsDataTree.Branch(GH_Path_Plate)[2];
        heightOfPlate = PlatePropsDataTree.Branch(GH_Path_Plate)[1] / 1000;

        // Since the thickness of the gusset plate is not calculated yet, make a
             conservative estimat for the thickness of the plates of 40 mm.
        thicknessOfPlateConservative = 40d / 1000d;

        // Calculate the effect of rolling the plate on the height and width of the
             gusset plate footprint on the ring:
        newWidthOfPlate = heightOfPlate * Math.Abs(Math.Sin(angleRoll)) +
             thicknessOfPlateConservative * Math.Abs(Math.Cos(angleRoll));
        newHeight = heightOfPlate * Math.Abs(Math.Cos(angleRoll)) +
             thicknessOfPlateConservative * Math.Abs(Math.Sin(angleRoll));

        // Add the yaw, pitch, height and width for the current plate in the updated
             lists:
        anglesYaw.Add(PlatePropsDataTree.Branch(GH_Path_Plate)[3]);
        anglesPitch.Add(anglePitch);
        neccWidthForPlate.Add(newWidthOfPlate);
        heights.Add(newHeight);
      }

      // Sort anglesYaw to fix if the bars are in the wrong order clockwise.
           Looping over all plates using an insertion sort algorithm.
      Indicies = new List<int>();
      Indicies = Enumerable.Range(0, nPlates).ToList();
      iTemp = 0;

      for (int n = 0; n < nPlates - 1; n++)
      {
        for (int i = n + 1; i < nPlates; i++)
        {
          if (anglesYaw[Indicies[n]] > anglesYaw[Indicies[i]])
          {
            iTemp = Indicies[i];
            Indicies[i] = Indicies[n];
            Indicies[n] = iTemp;
          }
        }
      }

      // Reset diameter to an unrealistic low value
      diameter = 0;

      // Find lengths for all plates by looping over all plates:
      for (int n = 0; n < nPlates; n++)
      {
        // Get the indicies for current and adjacent plates from the sorted anglesYaw
        index1 = Indicies[n];
        index2 = Indicies[(n - 1 + nPlates) % nPlates];
        index3 = Indicies[(n + 1) % nPlates];

        // Find half of the width of the chosen plate and bordering plates + a
             tolerance (w1, w2 and w3):
        w1 = neccWidthForPlate[index1] / 2 + tolerance;
        w2 = neccWidthForPlate[index2] / 2 + tolerance;
        w3 = neccWidthForPlate[index3] / 2 + tolerance;

        // Fetch height and pitch for the current plate:
        heightOfPlate = heights[index1];
        anglePitch = anglesPitch[index1];

        // Find the angles to adjacent plates (1 = counter clockwise, 2 = clockwise):
        angle_1 = (Math.Abs(anglesYaw[index1] - anglesYaw[index2]) + 4 * Math.PI) %
             (2 * Math.PI);
        angle_1 = Math.Min(angle_1, 2 * Math.PI - angle_1);
        angle_2 = (Math.Abs(anglesYaw[index1] - anglesYaw[index3]) + 4 * Math.PI) %
             (2 * Math.PI);
        angle_2 = Math.Min(angle_2, 2 * Math.PI - angle_2);

        // Get ratios between the widths of the plates
        psi_1 = Math.Min(w1 / w2, w2 / w1);
        psi_2 = Math.Min(w1 / w3, w3 / w1);

        if (Math.Acos(-psi_1) > angle_1)
        {
          r_1 = 1d / Math.Sin(angle_1) * Math.Sqrt(Math.Pow(w1, 2) + Math.Pow(w2, 2)
               + 2 * w1 * w2 * Math.Cos(angle_1));
        }
        else
        {
          r_1 = Math.Max(w1, w2);
```

```
   }
   if (Math.Acos(-psi_2) > angle_2)
   {
    r_2 = 1d / Math.Sin(angle_2) * Math.Sqrt(Math.Pow(w1, 2) + Math.Pow(w3, 2) + 2 * w1 * w3 * Math.Cos(angle_2));
   }
   else
   {
    r_2 = Math.Max(w1, w3);
   }

   // Find the corresponding lengths and minmize necessary length:
   diameter = Math.Max(diameter, 2 * Math.Max(r_1, r_2));
   diameter = Math.Max(diameter, heightOfPlate / 4);
  }
  NodeDataTreeInc0.Add(diameter, GH_Path_Node);
 }
 NodeDataTreeInc1 = NodeDataTreeInc0;
}
```

B.4 Top and Bottom of Cylinder

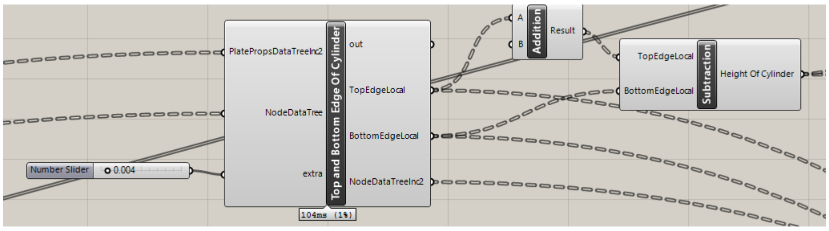


Figure B.4: Component calculating the top and bottom height of the cylinder.

```
private void RunScript(DataTree<double> PlatePropsDataTreeInc2, DataTree<double> NodeDataTree, double extra, ref object TopEdgeLocal, ref object BottomEdgeLocal, ref object NodeDataTreeInc2)
{
 GH_Path GH_Path_Plate = new GH_Path();
 GH_Path GH_Path_Node = new GH_Path();

 DataTree<double> BottomEdge_ = new DataTree<double>();
 DataTree<double> TopEdge_ = new DataTree<double>();

 double anglePitch, angleRoll, bottomEdge, topEdge, outerRadius, thicknessPlate, widthBeam, heightOfBeam, heightOfBeamEff;
```

```

    int nNodes, nPlates;

    nNodes = 0;
    GH_Path_Plate = new GH_Path(0, 0);

    while (PlatePropsDataTreeInc2.PathExists(GH_Path_Plate))
    {
      nNodes++;
      GH_Path_Plate = new GH_Path(nNodes, 0);
    }

    for (int m = 0; m < nNodes; m++)
    {
      GH_Path_Node = new GH_Path(m);

      nPlates = 0;
      GH_Path_Plate = new GH_Path(m, 0);

      outerRadius = NodeDataTree.Branch(GH_Path_Node)[6] / 2;

      while (PlatePropsDataTreeInc2.PathExists(GH_Path_Plate))
      {
        nPlates++;
        GH_Path_Plate = new GH_Path(m, nPlates);
      }

      bottomEdge = 1;
      topEdge = -1;

      for (int n = 0; n < nPlates; n++)
      {
        GH_Path_Plate = new GH_Path(m, n);

        anglePitch = PlatePropsDataTreeInc2.Branch(GH_Path_Plate)[4];
        angleRoll = PlatePropsDataTreeInc2.Branch(GH_Path_Plate)[2];

        heightOfBeam = PlatePropsDataTreeInc2.Branch(GH_Path_Plate)[1] / 1000;
        thicknessPlate = PlatePropsDataTreeInc2.Branch(GH_Path_Plate)[8] / 1000;
        widthBeam = PlatePropsDataTreeInc2.Branch(GH_Path_Plate)[0] / 1000;

        // Calcuate an effective height of the beam as an effect of rolling the
             rectangular cross-section:
        heightOfBeamEff = heightOfBeam * Math.Abs(Math.Cos(angleRoll)) + widthBeam *
             Math.Abs(Math.Sin(angleRoll));

        topEdge = Math.Max(topEdge, extra + outerRadius * Math.Tan(anglePitch) +
             heightOfBeamEff / 2 / Math.Cos(anglePitch));
        bottomEdge = Math.Min(bottomEdge, -extra + outerRadius *
             Math.Tan(anglePitch) - heightOfBeamEff / 2 / Math.Cos(anglePitch));
      }
      NodeDataTree.Add(topEdge, GH_Path_Node);
      NodeDataTree.Add(bottomEdge, GH_Path_Node);
      TopEdge_.Add(topEdge, GH_Path_Node);
      BottomEdge_.Add(bottomEdge, GH_Path_Node);
    }
    BottomEdgeLocal = BottomEdge_;
    TopEdgeLocal = TopEdge_;
    NodeDataTreeInc2 = NodeDataTree;
  }
```

B.5 Distance to Bolt Group

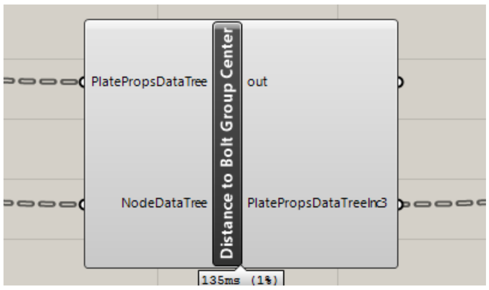


Figure B.5: Component calcualting the distance to the center of the bolt group.

```
  private void RunScript(DataTree<double> PlatePropsDataTree, DataTree<double>
      NodeDataTree, ref object PlatePropsDataTreeInc3)
  {
    GH_Path GH_Path_Plate = new GH_Path();
    GH_Path GH_Path_Node = new GH_Path();
    DataTree<double> LengthOfPlate_ = new DataTree<double>();
    DataTree<double> DistToCenterOfBoltGroup_ = new DataTree<double>();

    int nNodes = 0;
    int nPlates = 0;

    double a1, a3, nx, DistanceToTimber, LengthOfBoltGroup, neccDist;

    // Find number of nodes:
    GH_Path_Plate = new GH_Path(0, 0);

    while (PlatePropsDataTree.PathExists(GH_Path_Plate))
    {
      nNodes++;
      GH_Path_Plate = new GH_Path(nNodes, 0);
    }

    // Loop over all nodes:
    for (int m = 0; m < nNodes; m++)
    {
      // Find number of plates:
      nPlates = 0;
      GH_Path_Plate = new GH_Path(m, 0);
      GH_Path_Node = new GH_Path(m);

      while (PlatePropsDataTree.PathExists(GH_Path_Plate))
      {
        nPlates++;
        GH_Path_Plate = new GH_Path(m, nPlates);
      }

      // Loop over all plates
      for (int n = 0; n < nPlates; n++)
      {
        GH_Path_Plate = new GH_Path(m, n);

        a3 = PlatePropsDataTree.Branch(GH_Path_Plate)[13] / 1000;
        a1 = PlatePropsDataTree.Branch(GH_Path_Plate)[11] / 1000;
        DistanceToTimber = PlatePropsDataTree.Branch(GH_Path_Plate)[5];
        nx = PlatePropsDataTree.Branch(GH_Path_Plate)[6];

        LengthOfBoltGroup = (nx - 1) * a1;
        neccDist = LengthOfBoltGroup / 2 + a3 + DistanceToTimber;

        PlatePropsDataTree.Add(neccDist, GH_Path_Plate); // From center of the node
      }
    }

    PlatePropsDataTreeInc3 = PlatePropsDataTree;

  }
```

B.6 Bolt Holes Modelling

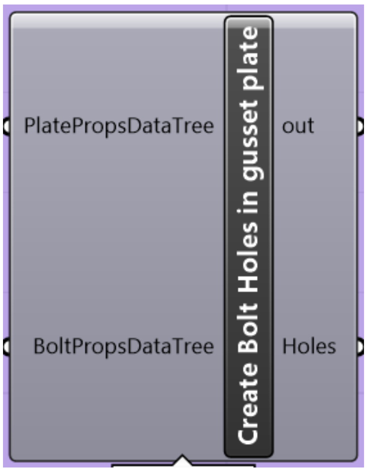


Figure B.6: The component generating the geometry for the bolt holes in the gusset plate.

```
private void RunScript(DataTree<double> PlatePropsDataTree, DataTree<double>
     BoltPropsDataTree, ref object Holes)
 {
   double tolerance = doc.ModelAbsoluteTolerance;
   GH_Path GH_Path_Plate = new GH_Path();
   GH_Path GH_Path_Bolt = new GH_Path();
   DataTree<Brep> CylinderDataTree = new DataTree<Brep>();
   Point3d P;
   Vector3d vecY;
   Plane pl;
   Circle circ;
   Cylinder cyl;
   Brep cyl2;
   int nNodes = 0;
   int nPlates = 0;
   int nBolts = 0;
   double x, y, z, d, t, distanceToBoltGroupCenter;

   // Find number of nodes:
   GH_Path_Plate = new GH_Path(0, 0);

   while (PlatePropsDataTree.PathExists(GH_Path_Plate))
   {
     nNodes++;
     GH_Path_Plate = new GH_Path(nNodes, 0);
   }
```

```

    // Loop over all nodes:
    for (int m = 0; m < nNodes; m++)
    {
      // Find number of plates:
      nPlates = 0;
      GH_Path_Plate = new GH_Path(m, 0);

      while (PlatePropsDataTree.PathExists(GH_Path_Plate))
      {
        nPlates++;
        GH_Path_Plate = new GH_Path(m, nPlates);
      }

      // Loop over all plates
      for (int n = 0; n < nPlates; n++)
      {
        GH_Path_Plate = new GH_Path(m, n);
        distanceToBoltGroupCenter = PlatePropsDataTree.Branch(GH_Path_Plate)[19];

        // Find number of bolts:
        nBolts = Convert.ToInt32(PlatePropsDataTree.Branch(GH_Path_Plate)[6] *
             PlatePropsDataTree.Branch(GH_Path_Plate)[7]);

        for (int k = 0; k < nBolts; k++)
        {
          GH_Path_Bolt = new GH_Path(m, n, k);

          x = BoltPropsDataTree.Branch(GH_Path_Bolt)[0] / 1000 +
               distanceToBoltGroupCenter;
          z = BoltPropsDataTree.Branch(GH_Path_Bolt)[1] / 1000;
          y = -PlatePropsDataTree.Branch(GH_Path_Plate)[8] / 2 / 1000;
          d = (PlatePropsDataTree.Branch(GH_Path_Plate)[9] + 2) / 1000;
          t = PlatePropsDataTree.Branch(GH_Path_Plate)[8] / 1000;

          P = new Rhino.Geometry.Point3d(x, y, z);
          vecY = new Rhino.Geometry.Vector3d(0, 1, 0);
          pl = new Rhino.Geometry.Plane(P, vecY);
          circ = new Rhino.Geometry.Circle(pl, P, (d + 2d / 1000) / 2);
          cyl = new Rhino.Geometry.Cylinder(circ, t);
          cyl2 = Rhino.Geometry.Brep.CreateFromCylinder(cyl, true, true);
          CylinderDataTree.Add(cyl2, GH_Path_Plate);
        }
      }
    }
    Holes = CylinderDataTree;
  }
```

B.7 Gusset Plate Modelling

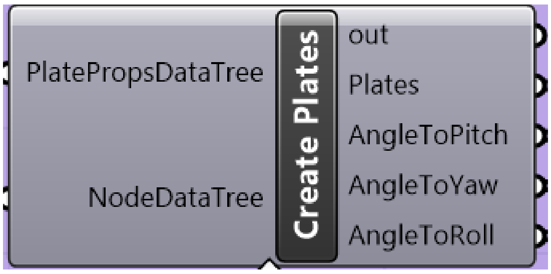


Figure B.7: The component generating the geometry for the gusset plates.

```
private void RunScript(DataTree<double> PlatePropsDataTree, DataTree<double>
    NodeDataTree, ref object Plates, ref object AngleToPitch, ref object
    AngleToYaw, ref object AngleToRoll)
  {
    GH_Path GH_Path_Plate = new GH_Path(0, 0);
    GH_Path GH_Path_Flat = new GH_Path();
    GH_Path GH_Path_Node = new GH_Path();

    DataTree<Rhino.Geometry.BoundingBox> Plates_ = new
        DataTree<Rhino.Geometry.BoundingBox>();

    DataTree<double> AngleToYaw_ = new DataTree<double>();
    DataTree<double> AngleToRoll_ = new DataTree<double>();
    DataTree<double> AngleToPitch_ = new DataTree<double>();

    Rhino.Geometry.BoundingBox box = new Rhino.Geometry.BoundingBox();

    int nNodes = 0, nPlates = 0;
    double a1, a3, e1, h, t, distLong, lengthBoltGroupX, neccDistToTimber,
        angleRoll, angleYaw, anglePitch, outerRadius, nBoltsX, ringThickness;

    while (PlatePropsDataTree.PathExists(GH_Path_Plate))
      GH_Path_Plate = new GH_Path(++nNodes, 0);

    for (int m = 0; m < nNodes; m++)
    {
      nPlates = 0;
      GH_Path_Node = new GH_Path(m);

      outerRadius = NodeDataTree.Branch(GH_Path_Node)[6] / 2;
      ringThickness = NodeDataTree.Branch(GH_Path_Node)[7] / 1000;

      GH_Path_Plate = new GH_Path(m, 0);
      while (PlatePropsDataTree.PathExists(GH_Path_Plate))
```

```
        GH_Path_Plate = new GH_Path(m, ++nPlates);

      for (int n = 0; n < nPlates; n++)
      {
        GH_Path_Plate = new GH_Path(m, n);

        t = PlatePropsDataTree.Branch(GH_Path_Plate)[8] / 1000;
        h = PlatePropsDataTree.Branch(GH_Path_Plate)[1] / 1000;

        // Add the contribution of angled beams to the contribution of the angled
             ring:
        angleYaw = PlatePropsDataTree.Branch(GH_Path_Plate)[3];
        anglePitch = PlatePropsDataTree.Branch(GH_Path_Plate)[4];
        angleRoll = PlatePropsDataTree.Branch(GH_Path_Plate)[2];

        a1 = PlatePropsDataTree.Branch(GH_Path_Plate)[11] / 1000;
        a3 = PlatePropsDataTree.Branch(GH_Path_Plate)[13] / 1000;
        e1 = PlatePropsDataTree.Branch(GH_Path_Plate)[15] / 1000;
        nBoltsX = PlatePropsDataTree.Branch(GH_Path_Plate)[6];
        lengthBoltGroupX = (nBoltsX - 1) * a1;

        neccDistToTimber = PlatePropsDataTree.Branch(GH_Path_Plate)[5];

        distLong = neccDistToTimber + a3 + lengthBoltGroupX + e1;

        Point3d Pt1 = new Rhino.Geometry.Point3d(0.001, -t / 2, -h / 2);
        Point3d Pt2 = new Rhino.Geometry.Point3d(distLong, t / 2, h / 2);

        box = new Rhino.Geometry.BoundingBox(Pt1, Pt2);

        Plates_.Add(box, GH_Path_Plate);

        AngleToYaw_.Add(angleYaw, GH_Path_Plate);
        AngleToPitch_.Add(anglePitch, GH_Path_Plate);
        AngleToRoll_.Add(angleRoll, GH_Path_Plate);
      }
    }
    Plates = Plates_;
    AngleToPitch = AngleToPitch_;
    AngleToYaw = AngleToYaw_;
    AngleToRoll = AngleToRoll_;
  }
```

B.8 Connectivity Data Tree

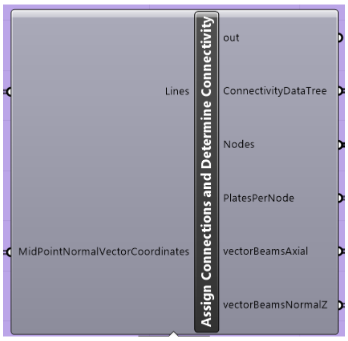


Figure B.8: The component generating the connectivity data tree and more.

```
private void RunScript(List<Line> Lines, DataTree<double>
     MidPointNormalVectorCoordinates, ref object ConnectivityDataTree, ref object
     Nodes, ref object PlatesPerNode, ref object vectorBeamsAxial, ref object
     vectorBeamsNormalZ)
{
  GH_Path GH_Path_Plate = new GH_Path();
  GH_Path GH_Path_Curve = new GH_Path();
  GH_Path GH_Path_Node = new GH_Path();

  DataTree<int> ConnectToPlate = new DataTree<int>(); // Contains two types of
       information: (1) Line id and (2) If it is a startpoint (=1) or not (=0).
  DataTree<double> Coordinates_ = new DataTree<double>();
  DataTree<Vector3d> vectorBeams_ = new DataTree<Vector3d>();
  DataTree<Vector3d> plateVectorTransformZ_ = new DataTree<Vector3d>();

  Point3d StartPoint = new Point3d();
  Point3d EndPoint = new Point3d();

  List<int> PlatesPerNode_ = new List<int>();
  List<Point3d> Nodes_ = new List<Point3d>();
  Point3d intCon;

  int DuplicateIndex = 0;

  List<Point3d> PointsToCheckForMultipleConnections = new List<Point3d>();
  bool StartPointIsInPointsToCheckForMultipleConnections,
       EndPointIsInPointsToCheckForMultipleConnections, alreadyInNodes;

  double tempAngleXY, tempAngleToHorizon, vecX, vecY, vecZ, dist;

  // Loop over all curves to check which nodes are connected to multiple beams:
  for (int c = 0; c < Lines.Count(); c++)
```

```
    {
      StartPoint = Lines[c].From;
      EndPoint = Lines[c].To;

      StartPointIsInPointsToCheckForMultipleConnections = false;
      EndPointIsInPointsToCheckForMultipleConnections = false;

      if (StartPoint != EndPoint)
      {
        // Check the start point:
        for (int p = 0; p < PointsToCheckForMultipleConnections.Count(); p++)
        {
          intCon = PointsToCheckForMultipleConnections[p];
          dist = Math.Pow(intCon.X - StartPoint.X, 2) + Math.Pow(intCon.Y -
               StartPoint.Y, 2) + Math.Pow(intCon.Z - StartPoint.Z, 2);
          if (dist < 0.15)
          {
            alreadyInNodes = false;
            for (int j = 0; j < Nodes_.Count(); j++)
            {
              intCon = Nodes_[j];
              dist = Math.Pow(intCon.X - StartPoint.X, 2) + Math.Pow(intCon.Y -
                   StartPoint.Y, 2) + Math.Pow(intCon.Z - StartPoint.Z, 2);
              if (dist < 0.15)
              {
                alreadyInNodes = true;
              }
            }
            if (alreadyInNodes == false)
            {
              Nodes_.Add(StartPoint);
              PlatesPerNode_.Add(0);
              StartPointIsInPointsToCheckForMultipleConnections = true;
              DuplicateIndex = p;
            }
          }
        }

        if (StartPointIsInPointsToCheckForMultipleConnections == false)
          PointsToCheckForMultipleConnections.Add(StartPoint);
        else
          PointsToCheckForMultipleConnections.RemoveAt(DuplicateIndex);

        // Check the end point:
        for (int p = 0; p < PointsToCheckForMultipleConnections.Count(); p++)
        {
          intCon = PointsToCheckForMultipleConnections[p];
          dist = Math.Pow(intCon.X - EndPoint.X, 2) + Math.Pow(intCon.Y -
               EndPoint.Y, 2) + Math.Pow(intCon.Z - EndPoint.Z, 2);
          if (dist < 0.15)
          {
            alreadyInNodes = false;
            for (int j = 0; j < Nodes_.Count(); j++)
            {
              intCon = Nodes_[j];
              dist = Math.Pow(intCon.X - EndPoint.X, 2) + Math.Pow(intCon.Y -
                   EndPoint.Y, 2) + Math.Pow(intCon.Z - EndPoint.Z, 2);
              if (dist < 0.15)
              {
                alreadyInNodes = true;
              }
            }
            if (alreadyInNodes == false)
            {
              Nodes_.Add(EndPoint);
              PlatesPerNode_.Add(0);
              EndPointIsInPointsToCheckForMultipleConnections = true;
              DuplicateIndex = p;
            }
          }
        }
        if (EndPointIsInPointsToCheckForMultipleConnections == false)
          PointsToCheckForMultipleConnections.Add(EndPoint);
        else
          PointsToCheckForMultipleConnections.RemoveAt(DuplicateIndex);
      }
    }

    Nodes = Nodes_;

    // Create the ConnectToPlate DataTree which tells us which lines are connected
         to which node and in which order they should appear in the DataTree:
    for (int c = 0; c < Lines.Count(); c++)
    {
      GH_Path_Curve = new GH_Path(c);

      StartPoint = Lines[c].From;
      EndPoint = Lines[c].To;

      tempAngleXY = Math.Atan2(EndPoint.Y - StartPoint.Y, EndPoint.X -
           StartPoint.X);
      tempAngleToHorizon = Math.Atan2(EndPoint.Z - StartPoint.Z,
           Math.Sqrt(Math.Pow(EndPoint.X - StartPoint.X, 2) + Math.Pow(EndPoint.Y -
           StartPoint.Y, 2)));

      // Calculate the twist based on pyramid geometry and the surface normal on
           the mdpoint of the lines:
      vecX = MidPointNormalVectorCoordinates.Branch(GH_Path_Curve)[0];
      vecY = MidPointNormalVectorCoordinates.Branch(GH_Path_Curve)[1];
      vecZ = MidPointNormalVectorCoordinates.Branch(GH_Path_Curve)[2];
```

```

        if (StartPoint != EndPoint)
        {
          for (int ic = 0; ic < Nodes_.Count(); ic++)
          {
            GH_Path_Node = new GH_Path(ic);

            intCon = Nodes_[ic];
            dist = Math.Pow(intCon.X - StartPoint.X, 2) + Math.Pow(intCon.Y -
                 StartPoint.Y, 2) + Math.Pow(intCon.Z - StartPoint.Z, 2);

            if (dist <= 0.15)
            {
              GH_Path_Plate = new GH_Path(ic, PlatesPerNode_[ic]++);

              ConnectToPlate.Add(c, GH_Path_Plate);
              ConnectToPlate.Add(1, GH_Path_Plate);

              plateVectorTransformZ_.Add(new Vector3d(vecX, vecY, vecZ), GH_Path_Node);

              vectorBeams_.Add(new Vector3d(EndPoint.X - StartPoint.X, EndPoint.Y -
                   StartPoint.Y, EndPoint.Z - StartPoint.Z), GH_Path_Node);
            }

            dist = Math.Pow(intCon.X - EndPoint.X, 2) + Math.Pow(intCon.Y -
                 EndPoint.Y, 2) + Math.Pow(intCon.Z - EndPoint.Z, 2);
            if (dist <= 0.15)
            {
              GH_Path_Plate = new GH_Path(ic, PlatesPerNode_[ic]++);

              ConnectToPlate.Add(c, GH_Path_Plate);
              ConnectToPlate.Add(0, GH_Path_Plate);//

              plateVectorTransformZ_.Add(new Vector3d(vecX, vecY, vecZ), GH_Path_Node);


              vectorBeams_.Add(new Vector3d(StartPoint.X - EndPoint.X, StartPoint.Y -
                   EndPoint.Y, StartPoint.Z - EndPoint.Z), GH_Path_Node);
            }
          }
        }
      }

      for (int ic = 0; ic < Nodes_.Count(); ic++)
      {
        GH_Path_Node = new GH_Path(ic);
        Coordinates_.Add(Nodes_[ic].X, GH_Path_Node);
        Coordinates_.Add(Nodes_[ic].Y, GH_Path_Node);
        Coordinates_.Add(Nodes_[ic].Z, GH_Path_Node);
      }
```

```

    ConnectivityDataTree = ConnectToPlate;
    PlatesPerNode = PlatesPerNode_;

    vectorBeamsAxial = vectorBeams_;

    vectorBeamsNormalZ = plateVectorTransformZ_; //

  }
```

Appendix C

Deriving Formulas for the Parametric Model

C.1 Formula for Spacing of Beams

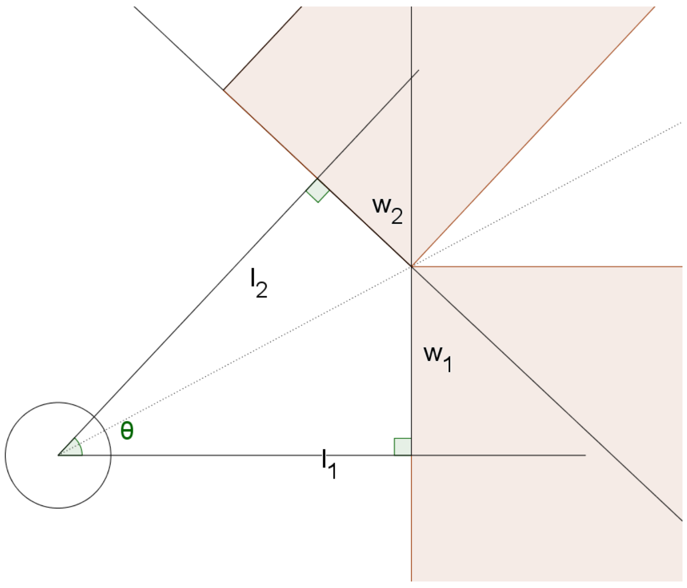


Figure C.1: Two rectangular objects packed as close together as possible without interfering with each other.

Figure C.1 shows that θ can be divided into to angles, creating two right triangles

$$\theta = \theta_1 + \theta_2 \tag{C.1}$$

where θ is restricted by

$$0 \le \theta \le \pi \tag{C.2}$$

Using Rottmann [47, p. 40], the following equation is true due to the shared hypotenus.

$$\frac{\sin(\theta_1)}{w_1} = \frac{\sin(\theta_2)}{w_2} = \frac{\sin(\theta - \theta_1)}{w_2} \tag{C.3}$$

Using Rottmann [47, p. 42],

$$\sin(\theta - \theta_1) = \sin(\theta) \cdot \cos(\theta_1) - \cos(\theta) \cdot \sin(\theta_1)$$

which by insertion in C.3 implies

$$w_2 \cdot \sin(\theta_1) = w_1 \cdot \sin(\theta) \cdot \cos(\theta_1) - w_1 \cdot \cos(\theta) \cdot \sin(\theta_1)$$

$$w_2 \cdot \tan(\theta_1) = w_1 \cdot \sin(\theta) - w_1 \cdot \cos(\theta) \cdot \tan(\theta_1)$$

$$w_2 \cdot \tan(\theta_1) + w_1 \cdot \cos(\theta) \cdot \tan(\theta_1) = w_1 \cdot \sin(\theta)$$

$$\tan(\theta_1) \cdot \Big(w_2 + w_1 \cdot \cos(\theta)\Big) = w_1 \cdot \sin(\theta)$$

$$\tan(\theta_1) = \frac{w_1 \cdot \sin(\theta)}{w_2 + w_1 \cdot \cos(\theta)} \tag{C.4}$$

Figure C.1 shows that $\tan(\theta_1)$ also can be written as

$$\tan(\theta_1) = \frac{w_1}{l_1}$$

implying that l_1 can be written as

$$l_1 = \frac{w_2 + w_1 \cdot \cos(\theta)}{\sin(\theta)} \tag{C.5}$$

It is easy to see that the formula can be used to calculate l_2 as well by changing the indicies

$$l_2 = \frac{w_1 + w_2 \cdot \cos(\theta)}{\sin(\theta)} \tag{C.6}$$

C.1.1 Avoiding Spurious Solutions

Since the formulas should work for any angle between 0 and π, it is important to verify that the resulting distances are valid practical solutions for all angles.

One such problem occurs when the angle becomes obtuse ($\pi/2 < \theta < \pi$). Somewhere in this interval, the equation for the most slender bar will give a distance from the center less than the minimum R. We will solve this by locking the length of the bar to R if Equation C.5 or C.6 give spurious results.

Let $l_2 = R$ and use Equation C.6 to find

$$R = \frac{w_1 + w_2 \cdot \cos(\theta)}{\sin(\theta)}$$

$$w_1 = R \cdot \sin(\theta) - w_2 \cdot \cos(\theta)$$

Inserted into C.5, this implies further that

$$l_1 = \frac{w_2 + \left(R \cdot \sin(\theta) - w_2 \cdot \cos(\theta)\right) \cdot \cos(\theta)}{\sin(\theta)}$$

$$l_1 = \frac{w_2 + R \cdot \sin(\theta) \cdot \cos(\theta) - w_2 \cdot \cos^2(\theta)}{\sin(\theta)}$$

$$l_1 = \frac{w_2 \cdot \sin^2(\theta) + R \cdot \sin(\theta) \cdot \cos(\theta)}{\sin(\theta)}$$

$$l_1 = w_2 \cdot \sin(\theta) + R \cdot \cos(\theta) \tag{C.7}$$

or in case $l_1 = R$

$$l_2 = w_1 \cdot \sin(\theta) + R \cdot \cos(\theta) \tag{C.8}$$

Note that equations C.7 does not depend on w_1. In practice, it can be shown that this criterion is sufficient since θ and w_2 completely describe the corner point on bar 2 which bar 1 has to interset with. Changes in w_1 does not effect how close bar 1 is to bar 2 – only changes in l_1 has an effect. This is illustrated in Figure C.2 where bar 2 can be as wide as required, although it interferes with the "cake slice" of bar 1.

The second problem area occurs when both $l_1 \leq R$ and $l_2 \leq R$ according to Equation C.7 or C.8. We can although solve this problem by setting both $l_1 = R$ and $l_2 = R$, since the angle completely secures against overlapping.

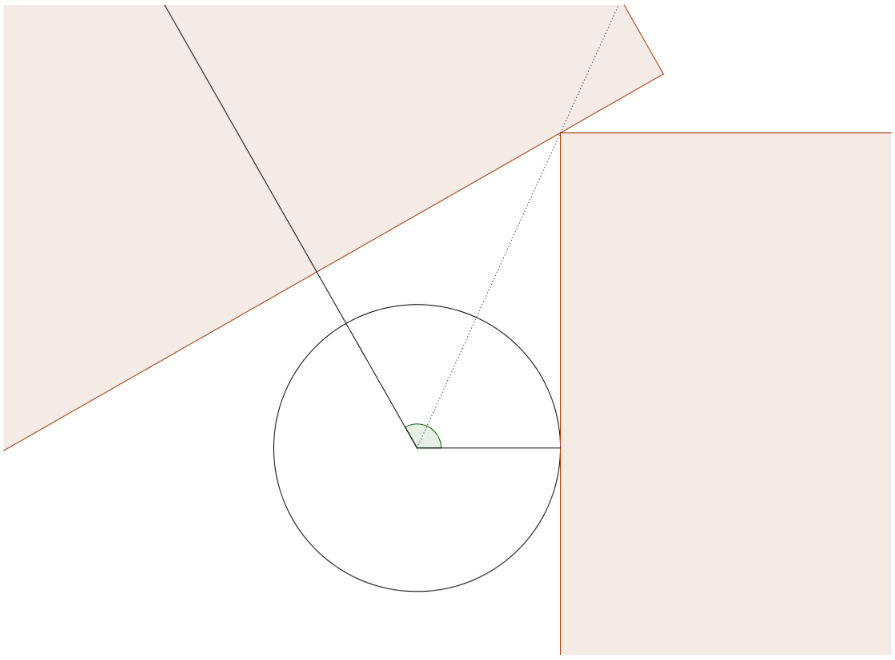


Figure C.2: Two bars spaced around a node. Bar 1 to the bottom right is limited by the radius of the circle. Bar 2 to the top left is limited by bar 1.

Hence using the set of rules established, the following example was calculated using $w_1 = 2.5$, $w_2 = 1.5$ and $R = 1$. The results are gathered in Figure C.3 for the domain $0 \leq \theta \leq \pi$. It is clear that the rules catches spurious results by forcing l_1 and l_2 to be larger or equal to R. It is also shown that the extreme cases of $\theta = 0$ gives ∞ while $\theta = \pi$ gives R, which is reasonable.

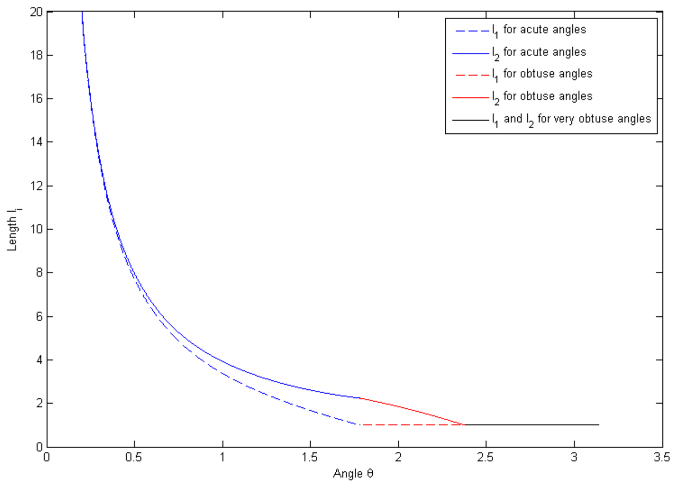


Figure C.3: The distance calculated from angles between 0 and π radians. Using $w_1 = 2.5$, $w_2 = 1.5$ and $R = 1$.

C.2 Formula for Calculating Minimum Ring Radius

The formula is based on the same theory as in Chapter C.1, only with some minor tweaks to the formulas. The ring radius is only depending on the plates and their foot print on the ring, making the spacing theory from C.1 valid for our situation.

We can therefore still use Equation C.4 to write .

$$\tan\theta_1 = \frac{w_1 \cdot \sin\theta}{w_2 + w_1 \cdot \cos\theta}$$

This time, using l_i as radius will not be sufficient. It is therefore necessary to calculate the diagonal r of the quadrilateral in Figure C.1

$$\tan\theta_1 = \frac{w_1}{l_1} = \frac{w_1}{\sqrt{r^2 - w_1^2}}$$

which inserted in C.4 implies that

$$\frac{1}{\sqrt{r^2 - w_1^2}} = \frac{\sin\theta}{w_2 + w_1 \cdot \cos\theta}$$

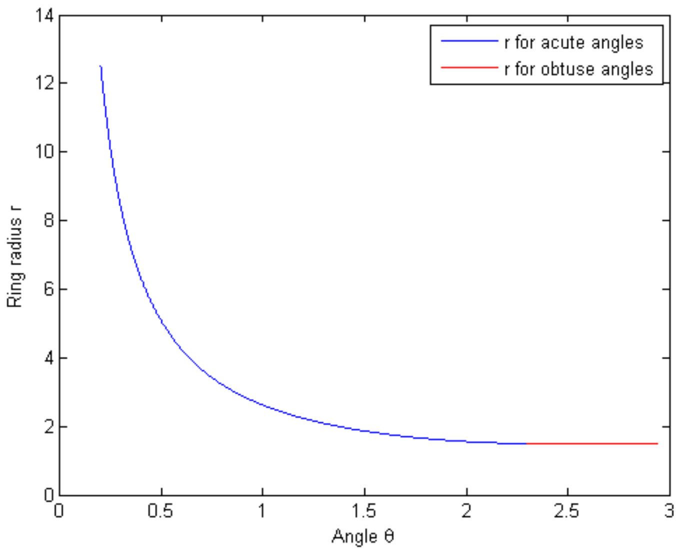


Figure C.4: Example of necessary ring radius when $w_1 = 1$ and $w_2 = 2$ for angles between 0 and π radians.

$$r = \sqrt{w_1^2 + \frac{\left(w_2 + w_1 \cdot \cos\theta\right)^2}{\sin^2\theta}}$$

which can be simplified to

$$r = \csc\theta \sqrt{w_1^2 + w_2^2 + 2w_1 w_2 \cos\theta} \tag{C.9}$$

This set of equations works until a certain angle θ where r starts increasing. It is clear that r has to be locked to the maximum of w_1 and w_2 when

$$\sin\left(\theta - \pi/2\right) = \min\left(w_1/w_2, w_2/w_1\right) = -\cos\theta$$

which implies that

$$r = \max\left(w1, w2\right) \tag{C.10}$$

when

$$\theta \geq \cos^{-1}\left(-\psi\right) \tag{C.11}$$

where

$$\psi = \min\left(\frac{w_1}{w_2}, \frac{w_2}{w_1}\right)$$

C.3 Formula for Plate Footprint on the Ring

Because of the pitching of the beams relative to the nodal coordinate system, the footprint of the plate on the ring is shifted vertically and prolonged. Figure C.5 shows this effect

$$z_t = \frac{d}{2}\tan\phi + \frac{h_{\text{eff}}}{2}\sec\phi \tag{C.12}$$

and

$$z_b = \frac{d}{2}\tan\phi - \frac{h_{\text{eff}}}{2}\sec\phi \tag{C.13}$$

using that

$$h_{\text{eff}} = h\cdot|\cos\omega| + t\cdot|\sin\omega|$$

where ω is the twist of the beam around the normal beam axis.

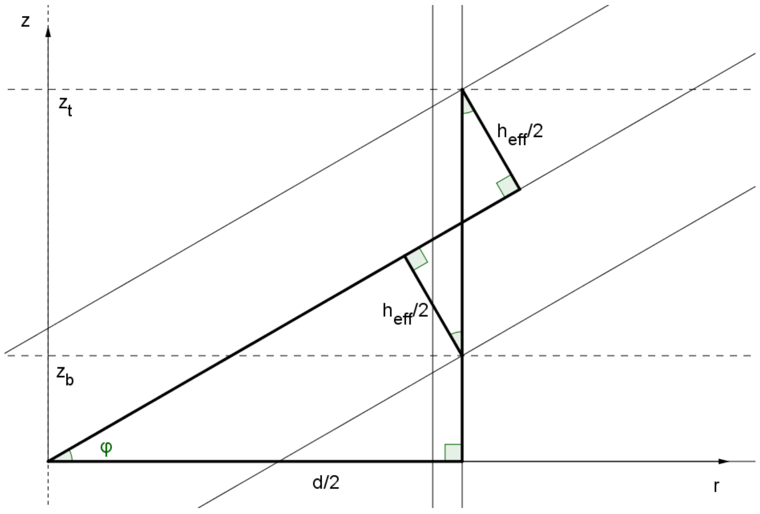


Figure C.5: The effect of pitching the plate an angle ϕ

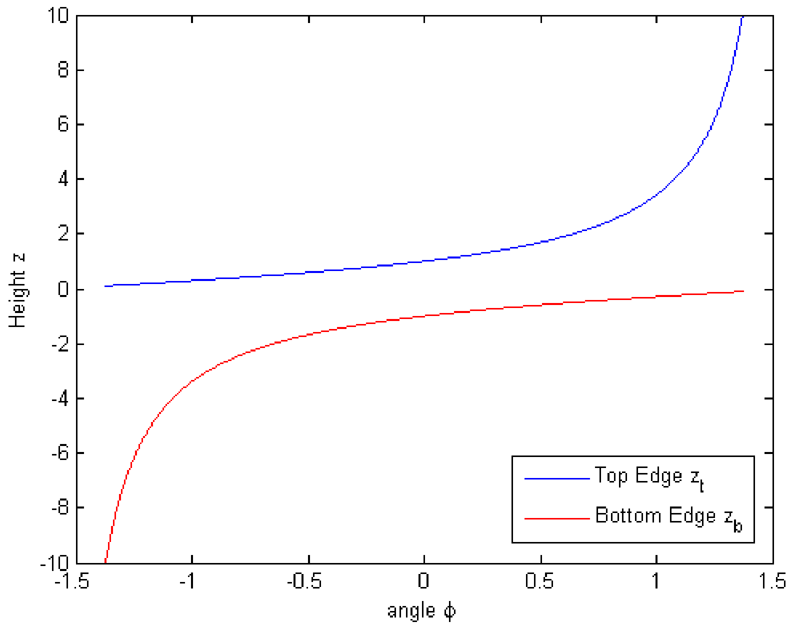


Figure C.6: Height of top edge and bottom edge from the plate footprint on the ring when $r = 1$ and $h = 2$.

C.4 Formula for Finding Necessary Length to Timber Edge before Pitching the Plates

Since all the plates are generated in the global coordinate system before they are pitched into a correct angle to the node, it is crucial that the effect of the shortened distance to the timber edge is considered. Figure C.7 shows that

$$l_{1,\text{edit}} \cdot \cos\phi - \frac{h}{2} \cdot \sin\phi = l_1 \tag{C.14}$$

$$l_{1,\text{edit}} = \frac{l_1 + h/2 \cdot \sin\phi}{\cos\phi} \tag{C.15}$$

$$l_{1,\text{edit}} = l_1 \sec\phi + \frac{h}{2} \tan\phi \tag{C.16}$$

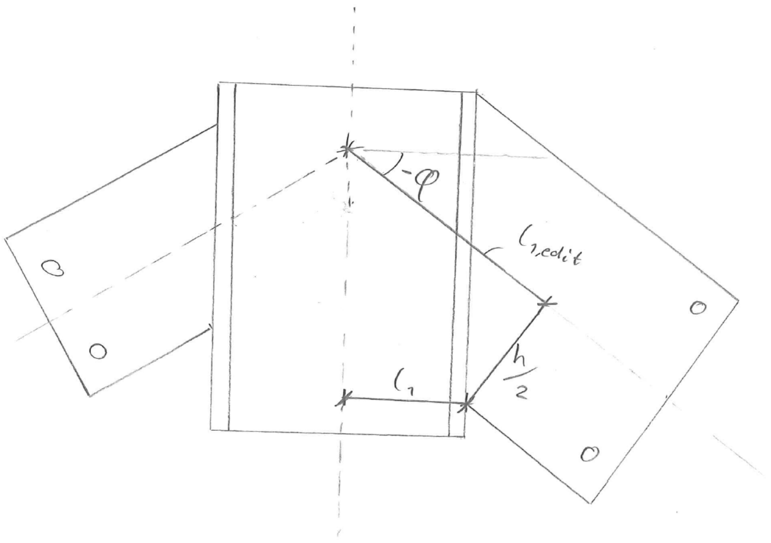


Figure C.7: Pitching the plates forces the plate to be lengthened.

Appendix D

Material Properties Aluminum C95500

✓ **STANDARD-STOCKED PRODUCT**

C95500

Cast • GreenAlloy™

Product Description:	NIckel Aluminum Bronze
Solids:	½" to 9" OD
Tubes:	1⅛" to 9" OD
Rectangles:	Up to 15"
Standard Lengths:	144"
Shape/Form:	semi-finished, mill stock or near-net shapes, anode, bar stock, billet/bloom, squares, hex, plate, profile or structural shape, flats/rectangular bar
Compliance:	C95500 is compliant with key legislation including (1) Federal Safe Drinking Water Act 1974 – SDWA, (2) Federal Reduction of Lead in Drinking Water Act 2011 and (3) California AF1953

Typical Uses

Builders Hardware	window hardware
Consumer	musical instruments, piano keys
Electrical	electrical hardware
Fasteners	stuffing box nuts
Industrial	machine parts, glass molds, welding jaws, wear plates, aircraft components, pickling equipment, valve guides/seats/bodies, piston guides, pump fluid ends, glands, worms, worm gears, hot mill guides, sewage treatment applications, valve components, bearings, gears, bushings, landing gear parts, handgun recoil mechanisms
Marine	ship building, covers for marine hardware, marine applications, marine hardware
Ordnance	government fittings

Note: Also available in heat-treated condition.

Similar or Equivalent Specification

CDA	ASTM	ASARCON	SAE	AMS	FEDERAL	MILITARY	OTHER
C95500	B505 B505M				QQ-C-390, G3	MIL-B-16033, CLASS 4	Aluminum Bronze 9D

Chemical Composition

Cu%	Fe%	Ni%[1]	Al%	Mn%
78.00 min	3.00-5.00	3.00-5.50	10.00-11.50	3.50

Chemical Composition according to ASTM B505/B505M-14

[1]Ni value includes Co.
Note: Cu + Sum of Named Elements, 99.5% min. Single values, unless otherwise noted, represent maximums.

Machinability

Copper Alloy UNS No.	Machinability Rating	Density (lb/cu in at 68˚ F)
C95500	50	0.272

Mechanical Properties

Tensile Strength, min		Yield Strength, at .5% extension under load min		Elongation, in 2 in. or 50 mm min	Brinell Hardness	Remarks
ksi	MPa	ksi	MPa	%	typical BHN	
95	655	42	290	10	208 (3000 kg)	

Mechanical Properties according to ASTM B505/B505M-14

Physical Properties

	US Customary	Metric
Melting Point – Liquidus	1930˚ F	1054˚ C
Melting Point – Solidus	1900˚ F	1038˚ C
Density	0.272 lb/in^3 at 68˚ F	7.53 gm/cm^3 at 20˚ C
Specific Gravity	7.53	7.53
Electrical Conductivity	8% IACS at 68˚ F	0.049 MegaSiemens/cm at 20˚ C
Thermal Conductivity	24.2 Btu · ft/(hr · ft^2 · ˚F) at 68˚ F	41.9 W/m at 20˚ C
Coefficient of Thermal Expansion	9 · 10^{-6} per ˚F (68˚-572˚ F)	15.5 · 10^{-6} per ˚C (20˚-300˚ C)
Specific Heat Capacity	0.10 Btu/lb/˚F at 68˚ F	419 J/kg at 293˚ C
Modulas of Elasticity in Tension	16000 ksi	110000 MPa
Magnetic Permeability*	1.32	1.32
Magnetic Permeability**	1.2	1.2
Poisson's Ratio	0.32	0.32

Physical Properties provided by CDA
*As Cast, Field Strength 16 kA/m **TQ 50 Temper, Field Strength 16 kA/m

Fabrication Properties

Joining Technique	Suitability
Soldering	Good
Brazing	Fair
Oxyacetylene Welding	Not Recommended
Gas Shielded Arc Welding	Good
Coated Metal Arc Welding	Good

Fabrication Properties provided by CDA

Thermal Properties

Treatment	Temp./Time - US	Temp./Time - SI
Stress Temperature	600	316
Solution Minimum	1600	872
Solution Maximum	1675	914
Solution Time	1.0	
Solution Medium	Water	
Precipitation Value		
Precipitation Time		
Precipitation Medium		
Annealing Minimum	1150	622
Annealing Maximum	1225	663
Annealing Time	1.0	
Hot Treatment Minimum		
Hot Treatment Maximum		

Thermal Properties provided by CDA

Appendix E

Parametric Model

See attached file:

Parametric_Model_Gridshell_Connections_2018-06.gh

Remark: All code in the orange group is from Huseby and Eliassen [44]. The model is used with their permssion.

Appendix F

FEA of Node 83 in Case Study

See attached file

Node_83_AbaqusAnalysis.CAE