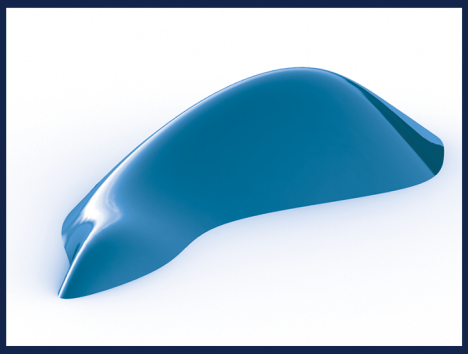
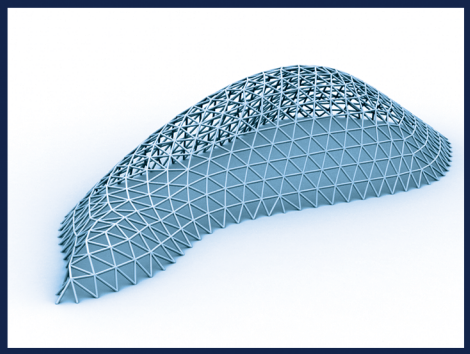
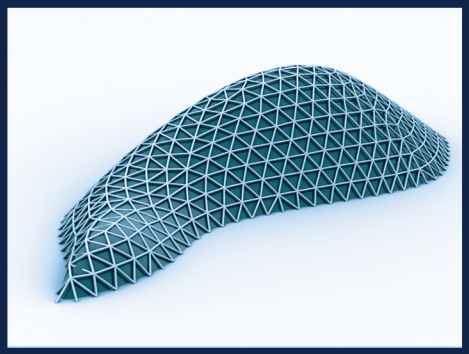
Master Thesis in Architectural Engineering

A Generative Gridshell Form Finding Tool

Development of a generative parametric gridshell form finding tool

Peter Vejrum

Faculty of Civil Engineering
Aarhus University
Section of Architectural Engineering

June 2013

AARHUS UNIVERSITY

A Generative Gridshell Form Finding Tool

Development of a generative parametric gridshell form finding tool

Master Thesis in Architectural Engineering at
Aarhus University

by

Peter Vejrum

AARHUS
UNIVERSITY

Preface

The following report was written as a result of my thesis project that concludes my Master studies in Architectural Engineering at Aarhus University. In the thesis I have developed and case tested a gridshell form finding tool aimed to aid architects and engineers in the conceptual design phases of a gridshell structure.

I would like to extend my sincerest thanks and appreciation to my two committed and understanding supervisors Morten R. Knudsen and Andreas Bak for their support and guidance throughout this project. I would also like to thank Andreas Bak for clearing a desk for me at his office in Bristol and taking his time to guide me through the development phase of this project.
Next, a special thanks goes out to Stephen Willacy (City Architect of Aarhus), Klaus Petersen (Schmidt Hammer Lassen) and Jeppe Flummer (Local Sculptor) for taking their time to sit and discuss the case of my thesis.
Lastly I would like to thank my girlfriend, friends and family for their support during this process.

Peter Vejrum
Aarhus, 2013

Table of Contents

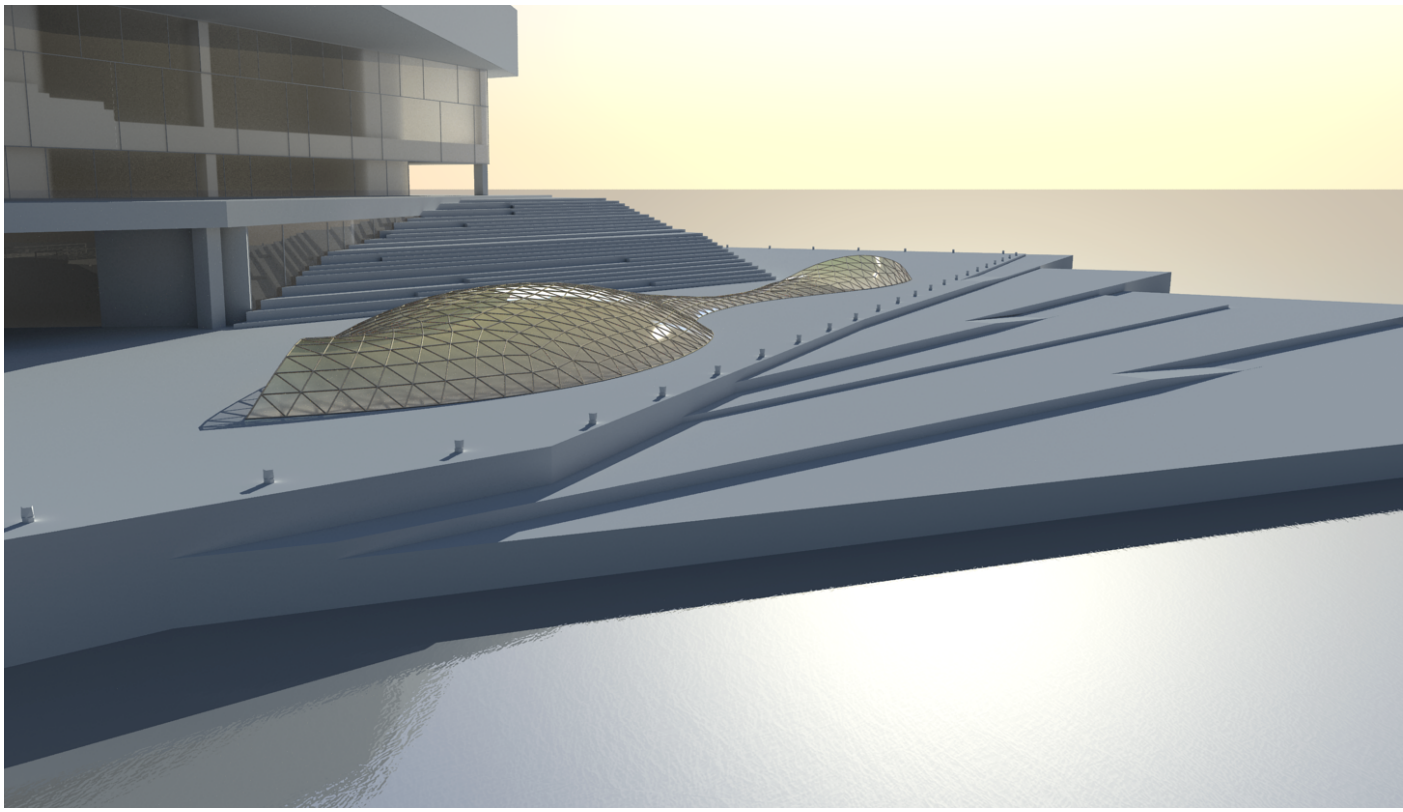
1. Abstract

A striking trend in contemporary architecture is the creation of complex free form gridshell structures. The trend is pioneered by architects and engineers working with cutting edge digital parametric software.
In thesis an actual gridshell form finding software tool is developed to help facilitate this new trend. The tool is targeted to help a team of architects and engineer to quickly generate and optimize the complex grid line geometry of a gridshell. The form finding tool should aid in the design process so that it will take less time, be less expensive and create longer spanning lightweight gridshells.
The geometric optimization of the form finding tool is the main focus of the thesis. The optimization component simulates the physical principle of hanging chain models made famous by great designers like Frei Otto, Heinz Isler and Antoniou Gaudi. Optimization is achieved with a self-developed dynamic relaxation algorithm programmed in the C# programming language. The geometric optimization will result in a severe reduction in member bending stress due to an improved membrane behaviour of the gridshell shape. Applied loads will mainly be resisted through axial compression resulting in smaller required cross-sections, longer spans, smaller deformations and higher natural frequencies. The dynamic relaxation component is coded in C# so that it can be directly implemented in the popular parametric modelling software Grasshopper.

The utilization of the form finding tool can be described in 5 steps:

1. The designer creates an initial free form surface acting as the input shape of the tool. The surface is linked to Grasshopper.
2. The form is tessellated in to a grid of discrete line elements. The tessellation is performed through generative algorithms made in Grasshopper that divides the surface into points. The points are connected to create triangular, quadrilateral, hexagonal or Voronoi grids. The designer selects the desired type of grid.
3. The designer defines a list of points where the grid is supported.
4. The grid of lines and list of support points is fed to the dynamic relaxation algorithm. The algorithm transforms the grid of lines into a particle-spring-system. The differential equations of the system is solved iteratively with a numerical integration scheme called Verlet integration. The relaxation of grid-lines is out-put with real-time updates to Grasshopper that allows the designer to follow the optimization process. The grid of lines is fully relaxed once the numerical integration converges towards an equilibrium state.
5. The relaxed lines is dynamically linked to a commercial finite element analysis with a component called Karamba. Karamba structurally analyzes and verifies the members of the grid for user chosen applied loadings and cross-sections.

The components developed are evaluated in small examples and a case. The case has the purpose of showcasing all the developed components. The case treat the creation of a free form gridshell playground structure.
The result of the thesis is that the developed tool is able to quickly generate and optimize the geometry of very complex grid geometries. The optimization results in large structural benefits allowing for the design of larger, lighter and more impressive free form gridshells.



2. Introduction

> "I would distinguish the difference between the engineer and the architect by saying the architect's response is primarily creative, whereas the engineer's is essentially inventive." - Peter Rice

The best structural designs are born in processes where aesthetical, functional and technical requirements are all being included in the conceptual phase. This inclusion can be achieved if architects and engineers work together in the earliest stages of a project.
Using digital tools in the conceptual design phase of an project is recently becoming more popular among architects and engineers. It used to be that the digital tools used by architects, would not directly link to the analysis and simulation tools of the engineer. This caused a delay when evaluating architectural concepts with engineering software. It also resulted in a split work process where architects would design a concept and then have it evaluated structurally by engineers. This process would go back and forth until a design was selected.
A new trend sees designing teams of architects and engineers use parametric software that combines digital architectural modelling with direct structural analysis and simulations. This means that structural evaluation can be done instantaneously. The trend calls for close collaboration between the architects and engineers which should promise for great designs. This trend has also resulted in a change in contemporary architecture. The new software tools available allow the generation of new and more complex structures such as free form gridshell structures.

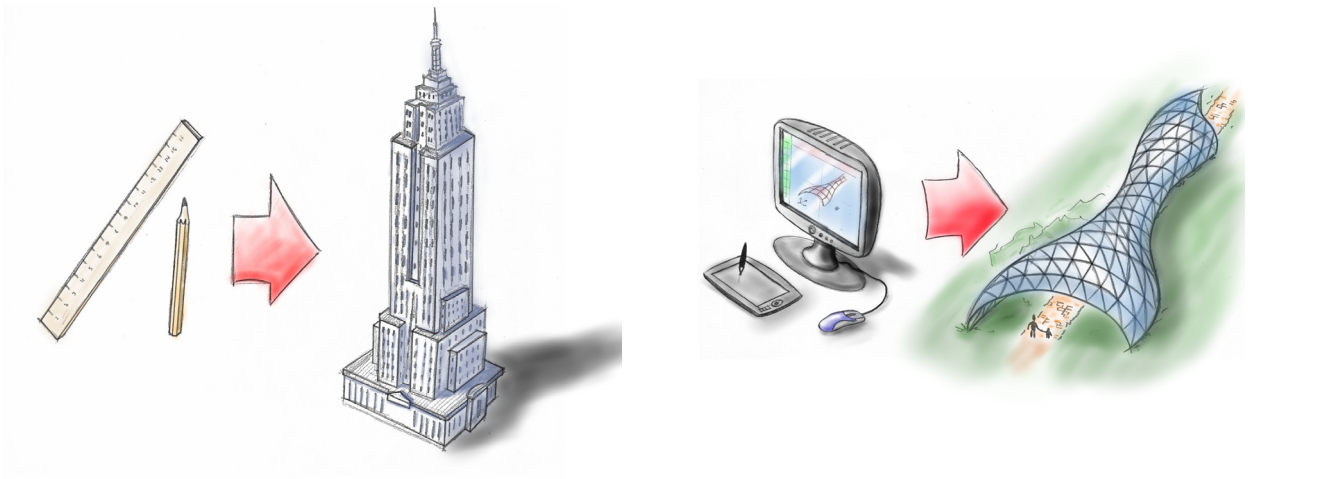


Figure 1: The tools available to the designing team can be seen in the architecture they create. New powerful computational tools allow designers to create complex free form geometries.

Computer programming allows designing teams to self-develop software tools, that can be implemented in the parametric software, to solve specific and well defined design problems. A common design problem among architects and engineers is finding the right form for a structure. The choice of form directly dictates the amount of material required to resist applied loads, so a smart choice of form will reduce costs and allow for lighter structures. Structural form finding of complex geometries has a long tradition of creating physical models to approximate structural behaviour. Physical form finding was used to generate and optimize the shape of complex structures such as shells or membrane structures. As computers become more and more powerful, form finding can be performed with detailed computational models. The form finding process becomes much faster and accurate and allows the designing

team to explore a wide variety of possible solutions.

The motivation for this thesis, is to develop a software form finding tool. The tool should aid a designing team of architects and engineers to quickly generate, optimize and analyze the geometry of complex gridshell structures with a minimal amount of time spent calculating and modelling.

The main focus of the thesis is, through self-developed code, compose a form finding algorithm that can perform the geometric optimization of a gridshell. The backbone of the form finding tool will be a custom scripted dynamic relaxation algorithm developed in the C# programming language. Dynamic relaxation can be thought of as the numerical evolution of the hanging chain model. The dynamic relaxation algorithm will be coupled with generative algorithms that create grid-line geometry from free form surfaces to create gridshells.

A gridshell structure resembles a shell structure and their shapes are almost identical. Both shells and gridshells are lightweight structures that allow for very large spans. The difference between the two is that shell structures consists of a continuous surface while gridshells is a combination of discrete members connected in structural nodes. Both structure types mainly resist applied loads through axial stress, but the cross-sections of the gridshell allow it to resist bending forces that will be present if the form is not sufficiently optimized. Grid shells are commonly used as lightweight roof structures.

Figure 2: The Blop. A gridshell structure in Eindhoven, Netherlands. Image source: freeform-structures.com

The structure of this thesis is divided into two major parts and one minor parts.

- The first major part, found in chapter 3, concerns the functionality that a gridshell form finding tool must possess to ensure its successful implementation. The goal of the first part is to derive a list of requirements for the developed software. This is achieved through a study of different relevant subjects and theory the form finding tool will include.
- The second major part, found in chapter 4, describes the developed form finding software tool in detail as well as showcasing its functionality. The tool is build around the parametric modelling platform Grasshopper. The benefits of parametric modelling and Grasshopper are explained in chapter 4.1. The tool includes generative algorithms that create grid-line geometry from free form surfaces described in chapter 4.2. A close-inspection of the dynamic relaxation algorithm that optimizes the grid-line geometry as a particle-spring-system is found in chapter 4.3 .
- The last, and minor part, found in chapter 5, is a case study that utilizes the developed form finding tool. The goal of the case is to showcase some of the possibilities and functionality of the tool when working with conceptual design. The case selected is a real world project that will be put out for competition sometime in the future. It is not the intention to create a fully polished design, but to use the tool to create a number of suggestions. The case concerns creating a free form playground structure near the new Urban Media Space in Aarhus (UMS). The tool will be used to create a free-form gridshell structure that will act as a controversial playground.

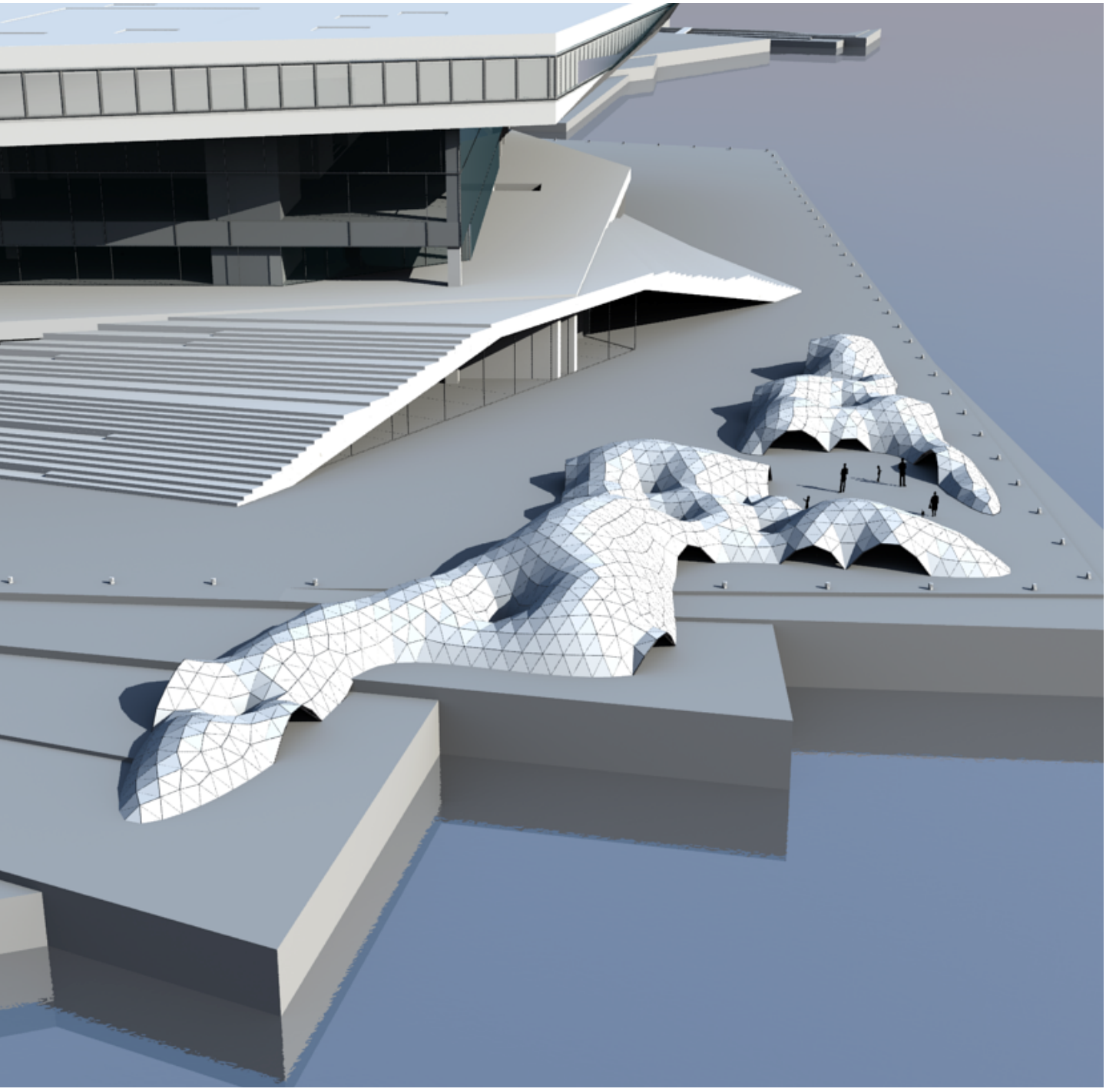


Figure 3: Geometry created with the developed form finding tool that will be explained in this thesis.

3. Requirements of a gridshell form finding tool

The following pages consists of passages aimed to research the required utilities that the gridshell form finding tool must possess in order to be beneficial in a collaborative form finding process. A short summary of findings will be presented after each passage. The summary presents the utilities and parameters that should be included in the developed tool.

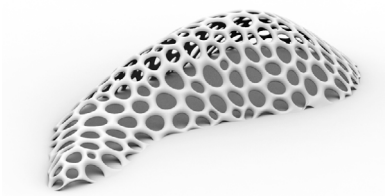


Figure 4: An organic looking gridshell structure created using the design tool. See much more in chapter 4.3.

3.1 Breaking boundaries of traditional engineering

The following passage will try to briefly highlight the difference in how engineers and architects approach structural form finding by explaining their approaches of solving problems. The goal is to use this knowledge when developing the design tool to target its functionality so it will be beneficial for both parties.
The collaborative nature of the form finding tool is inspired by the legacy of pioneering engineer Peter Rice (1935–1992). Rice believed that the modern engineer has to break free from the boundaries of traditional engineering in order to achieve innovative utilization of structural materials. In a speech about engineering Rice said the following:

> "... By using only pragmatism, by using only rational thought, by always demanding and wanting to know whether what we want to do is right, we destroy the very basis upon which the good or noble things in our life exist. " - Peter Rice

Rice believed that in order to be creative and imaginative, engineers must not be inhibited by only solving problems rationally. In contrast, architects work subjectively rather than objectively when solving problems [Larsen, 2003]. Architects are trained in taking an emotional and empathic approach to design. This distinction in problem solving is clearly an issue when engineers and architects work on the same design but at different stages of the process. Their different approaches will lead to different results. Rice had the philosophy that innovative structural design can be achieved with a symbiotic approach between the engineer and the architect. The architect will aid the engineer in breaking free of pragmatism where the engineer will extend the possibilities of structural materials to achieve the ideas of the architect.

3.1.1 The modern symbiotic master builder

Throughout medieval history, during the construction of the great cathedrals, artisans known as master builders single-handedly functioned as both architect, craftsman and engineer. The master builder was skilled in both artistic and technical design, utilizing knowledge brought down from generation to generation. It was not until the industrial revolution and the development in fields of structural- and material science that a segregation of trades lead to the birth of the two professions architect and engineer.

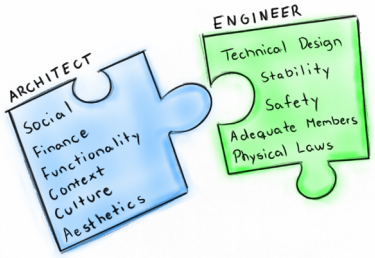


Figure 5: Structural design is a puzzle where both architects and engineers bring pieces to the solution.

The Swiss-born architect and designer Charles-Édouard Jeanneret-Gris, known as Le Corbusier, explained the roles of the modern architect and engineer:

> "These are the engineer's responsibilities; the respect of physical laws, the strength of materials, supply, economy considerations, safety, etc. And these the architect's; humanism, creative imagination, love and beauty, and freedom of choice." - Le Corbusier

Traditionally when engineers and architects collaborate to design a structural concept they divide their tasks between them. The architect creates the project concept while including contextual, social, functional and physical issues. The engineer works with technical issues such as structural calculations, member selection and stability while trying to realize the architectural vision without compromising the original architectural concept. Not changing the initial design is never the case because the initial structural designs are not made in collaboration, and a lot of things will later have to be changed because of it. This is not without cost. Major decisions and alterations are less expensive to make early in the design process and increase in cost as the detailing progresses.

The different responsibilities mentioned by Le Corbusier leads to a clear cognitive distinction in how architects and engineers approach structural design. Architects will tend to work subjectively by envisioning the structural appearance and functionality of the building while incorporating cultural, social and aesthetical considerations. Engineers will try to approach a structural solution objectively by rationally breaking down problems into something they can calculate. This results in a systematical process where the engineers solves a structural puzzle of finding a combination of adequate structural members. This is where the problem of the traditional structural design process lies. The structural design concept should not be designed by the architect or the engineer alone. It should be, as Rice imagined, a collaborative undertaking where the two parts have mutual understanding and respect for each other. The designing team should function as a symbiotic modern master builder.

Some of the best examples of modern building design are realizations of the combined efforts of engineers and architects creating a structural design readable in the architecture. A design where aesthetics, functionality and structural efficiency are merged into a tactile solution.

Figure 6: The Sidney Opera house is a brilliant example of magnificent structural design from a collaborative effort by the architect and engineer. Peter Rice is credited for solving the problematic geometry of the roof shells thus realizing the visions of the architect Jørn Utzon. Image source: http://en.wikipedia.org/wiki/File:Sydney_Opera_House_Sails.jpg

3.2 Free-form design - A look to nature

"The form of an object is a diagram of the forces." - D'Archy

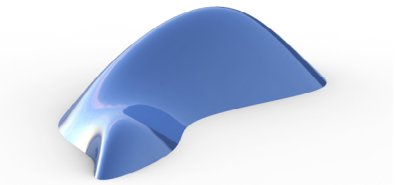


Figure 8: A free form blop like surface that will considered in further detail in chapter 4.

The tool that is to be developed is focused around creating a free form gridshell structure. Before getting into the specific structural capabilities of the gridshell, a rundown of the free form concept is required.

The very basic function of buildings is to shelter people from external forces such as the sun or bad weather. In the majority of building structures there is a need for level floors. Normally structural elements are fabricated to be straight for this reason and simply because it logistically easier and cheaper to build with repetitive straight elements and orthogonal connections.

The governing force upon buildings is gravitational vertical live and dead loads. The structural goal is to carry gravitational forces to the ground. The structural system is chosen from interior functionality and externally applied loads.

Longitudinal straight elements are best utilized when the gravitational force acts upon them parallel and in-plane to the element's longitudinal axis. This is the case with vertical elements such as columns and walls which are highly efficient structurally. Horizontal elements, such as beams or slaps, resist loads through out of plane bending and deformations due to inherent stiffness. These horizontal elements are heavy and inefficient. There is not much room for structural optimization in straight element buildings, except for choosing the lightest adequate beams, slaps and columns. Also the appearance is bound to be a combination of straight elements, limiting the freedom of the aesthetical design.

A shift towards free form structures can be witnessed in contemporary architecture. Innovation and imagination are required when creating these free form structures, seeing how they often are highly complex and unique. In order for the architect to be creative and the engineer to be inventive both must often study nature for inspiration. Nature has no need for planar surfaces when creating structures. This is due to the fact that everything nature creates is designed to aid its own survival and not the comfort of its inhabitants. Everything, nature creates, consists of irregular organic free form shapes. There are seldom completely right angles or straight lines in nature. This is due to highly complex and irregular forces found in nature such as wind, gravity, water, etc. Nature builds its structures to best cope with these forces and can do so without needing to duplicate structural elements for the sake of production cost. Every single element, that together forms a natural existing structure such as the branch of a tree or the bone of an animal, is unique. Elements are unique, because nature fabricates elements by growing them into the shape they need based on the principle of minimizing energy and material needed to cope with the forces that act upon them.

Modern technology has lead to a paradigm shift in digital element fabrication that allows modern engineers to adapt this rule, because the production cost of a thousand identical elements are no cheaper to produce than the same number of different elements [Kolarevic, 2005].

Since the goal of this thesis is to create a tool that finds the form a free-form gridshell structure, it makes a lot of sense to look to nature. If nature was to solve the problem of enclosing an area, it would create a free form membrane structure, simply because it is the most material- and energy-efficient way to do so.

The principles of minimizing energy and material usage to create a free form structure are examined further in chapter 3.4 about structural form finding. These principles will be applied in the form finding tool to create free form structures with a minimum amount of material.

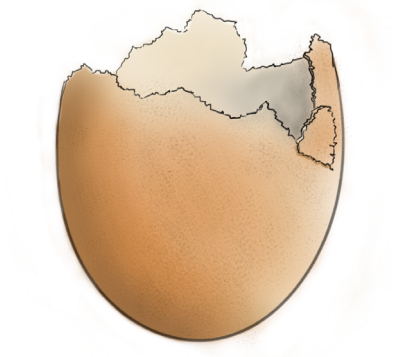


Figure 7: The membrane behaviour of a eggshell allows the shell to be extremely thin in respect to the amount of space it encloses while still being very strong. This behaviour ensures that only the absolutely required material is used.

Figure 9: The internal structure of a leaf is highly complex and irregular. The leaf is optimized through evolution to best serve the needs of the tree while being able to resist external loads. Image source: http://en.wikipedia.org/wiki/File:Leaf_1_web.jpg.

Design goals derived from the two previous passages:

1. The tool should be made easy to use for both architects and engineers. This means that the tool must be able to communicate with leading industry CAD platforms.
2. It should provide utilities to aid and optimize technical, functional and aesthetical design.
3. The form finding tool should draw inspiration from nature by optimizing and finding form based on external loading, thus reducing internal energy and reducing material.
4. The tool should not be restricted to straight elements. The structures should be free form, just as they would be in nature so structural complexity should not possess a barrier.

3.3 Membrane action structures

A structural understanding of membrane behaviour found in nature is required to make use of if developing the gridshell form finding tool. Continuous membrane structures are normally referred to as shells in architecture. Both shells and grid-shells are often used as roof structures due to their long spans and light weight and appearance.

3.3.1 Continuous shells

Continuous shells are three dimensional surfaces that posses double curvature. As structural elements, shells are unique because of their long span to thickness ratio, often smaller than 500 to 1 [Chilton, 2000]. This capability stems from double curvature of the shell that leads to membrane action which efficiently diverts gravitational force to the shell's supports via in-plane stresses. This allows shell structures to appear elegant and lightweight and cover large areas without the need for columns. This structural efficiency is why membrane structures are found throughout nature.

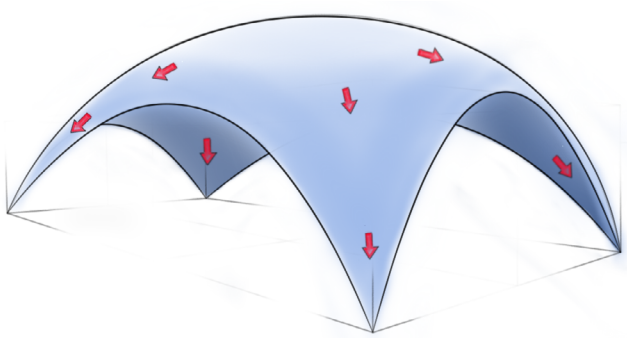


Figure 10: Shell structures carry loads through in-plane stresses.

Continuous shell structures are not separated into vertical and horizontal elements like grid-shells. Shells designed properly possess membrane action that allows only for uniformly distributed in-plane stress fields. These in-plane stress fields consist of normal stresses and shear stresses. Out-of-plane bending stresses are near negligible if the shape of the shells is optimized correctly. The membrane action of the initial curvature enables the shell to resist in-plane forces and out-of-plane loads.

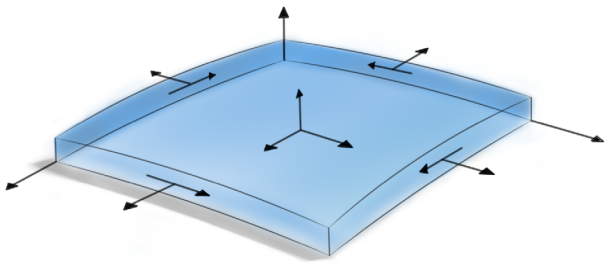


Figure 11: Visualization of the in-plane shear and normal stress on a infinitesimal shell element.

In order to optimize the structural resistance of a shell the shape needs to reduce bending actions as much as possible [Hoefakker, 2010]. This optimization can be achieved through form finding.
The Swiss engineer Heinz Isler conducted experiments with flat and curved structures. One of his experiments showed that a curved plastic element could resist more than 30 times the loading of the same flat plastic element. The reason is that when shells are designed properly there is no need for a cross section with high moment of inertia, seeing that there is very little bending. This allows shells to have the small thickness that makes them so unique.

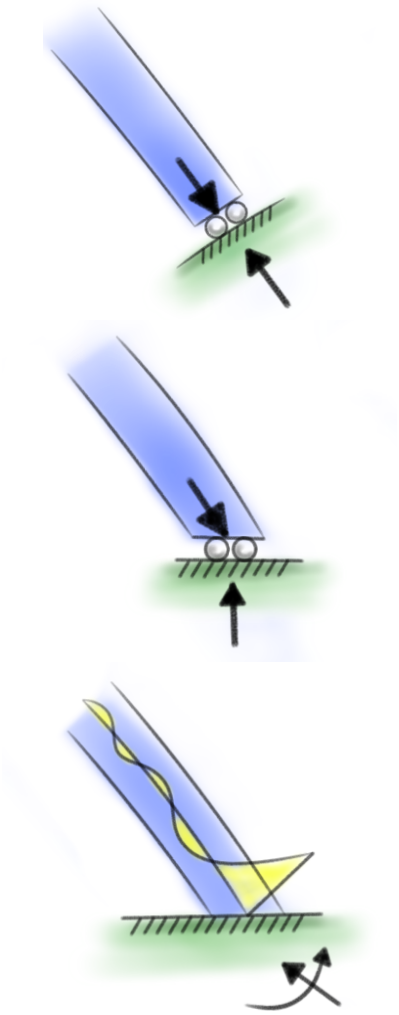


Figure 12: Three different support conditions for a membrane is shown. The first is membrane compatible and can be described using traditional membrane theory. The second and third support options are membrane incompatible.

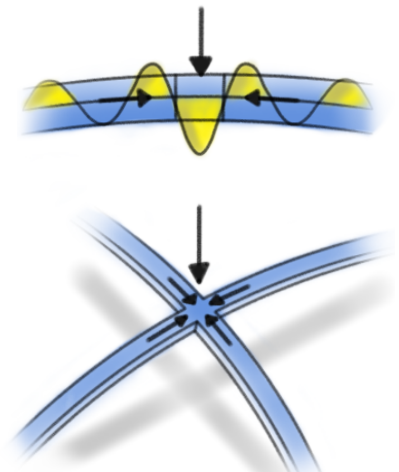


Figure 13: Concentrated loads on the membrane or changes in geometry due to large deflection will disturb the membrane behaviour.

As a result of the elaborate, spatial shape shells behave differently from normal beam or plate structures. The mathematical description of the shell properties is thus more complicated than that of beams or flat plates. Optimizing shell structures is a complex process and used to be very time consuming, because there was a need for physical modelling, to find the desired optimal shape. This is no longer the case with powerful computational form finding as described in chapter 3.4.2. The form of the shells was originally found experimentally to ensure that no out of plane bending would occur. Engineers such as Heinz Isler and Frei Otto performed physical modelling to find the shapes that lived up to these requirements. This meant that shapes were limited and production costs were large. [Dimčić, 2011]

According to membrane theory [Hoefakker, 2010], there is a set of equilibrium requirements to achieve true membrane behaviour. These requirements are important when designing the support conditions for the shell structure shown in figure 12.
If these boundary requirements are not met, and the membrane behaviour is disturbed by bending actions then additional structural elements such as ribs or edge beams are required. True membrane behaviour shells are rarely seen in architecture. Most shells will have some sort of bending reinforcement. As seen in figure 12, many of the membrane disturbances are related to the supports of the shell or loads that lead to large deformations. Consequently edge reinforcements are usually efficient to achieve undisturbed membrane behaviour in the rest of the shell.
Buckling can become an issue with thin shells. Double curvature makes so, that the shell cannot buckle in-plane but only out of plane.
Shells perform best when loads are distributed uniformly over the surface. Point loads will deform the structure and result in disturbed membrane behaviour, figure 13.

Downwards loading results in large horizontal reactions in shells. The in-plane forces developed in a shell have outwards facing components. These outwards facing reaction must be absorbed in the support. Traditional means of doing so is creating closed buttresses or a tension ring.

Figure 14: The most simple continuous shell structure is the semispherical dome. Throughout history these domes were erected using materials strong in compression such as bricks or un-reinforced concrete. Pantheon, build 126 AD, is made out of unreinforced concrete. Concrete, as a material was lost for ages. To this day, Pantheon is the largest un-reinforced concrete dome in the world. It spans 43.3 meters.

Image source: http://commons.wikimedia.org/wiki/File:Internal_Pantheon_Light.JPG

Figure 15: Heinz Isler (1926-2009) used physical modelling to find the shape of his magnificent concrete shells. This shell is a roof to a highway service area.

Image source: http://commons.wikimedia.org/wiki/File:Deitingen_Sued_Raststaette,_Schalendach_01_09.jpg

Figure 16: L'Oceanogràfic is a great example of structure that makes concrete apear lightweight when used as a shell. The design is a combined effort of the architect Félix Candela and engineers Alberto Domingo and Carlos Lázaro.

Image source: http://commons.wikimedia.org/wiki/File:L'Oceanografic_(Valencia,_Spain)_01.jpg

3.3.2 Gridshells

Gridshells posses the same overall free form double curved shape as continuous shells. They are however structurally different. In gridshells, the material is not continuous, but transformed into discrete members forming the shell structure often resulting in a large amount of structural elements. This results in a high structural complexity which is one of the main reasons why few have been built. Also the fact that gridshells are best utilized as roofing structures limit their area of use.

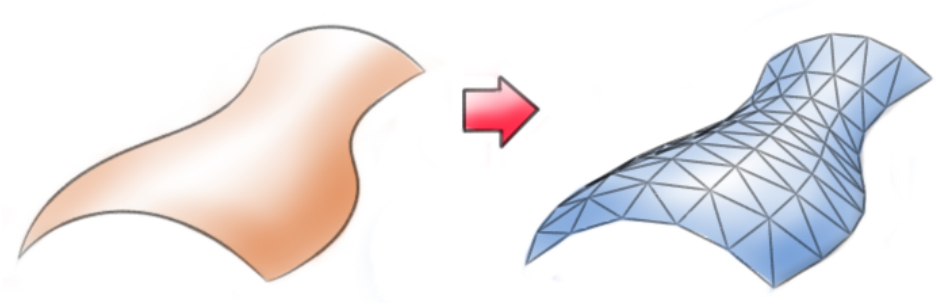


Figure 17: . A gridshell is a free form surface transformed into discrete structural elements connected in joints.

3.3.3 Structural behaviour:

In continuous shells loads are distributed as in plane normal and shear stress distributed equally over the continuous material that make up the shell. This membrane behaviour activates all the material in the shell and locks everything in its place. [Toussaint, 2007]
The gridshell is a system of structural elements connected in nodes/joints, that transmit forces in the direction of the members but also resist out of plane bending due to the profile nature of the members. The inherited bending stiffness of the gridshell members will make up for any disturbance in the membrane behaviour. Diagonal stiffness is needed to make sure that the members of the gridshell is locked in place and allow shear forces to be transmitted from one element to another. This can be achieved with rigid member connections, a continuous layer/ skin over the members, cross cables between nodal joints, or rigid cross bracing between nodal joints.

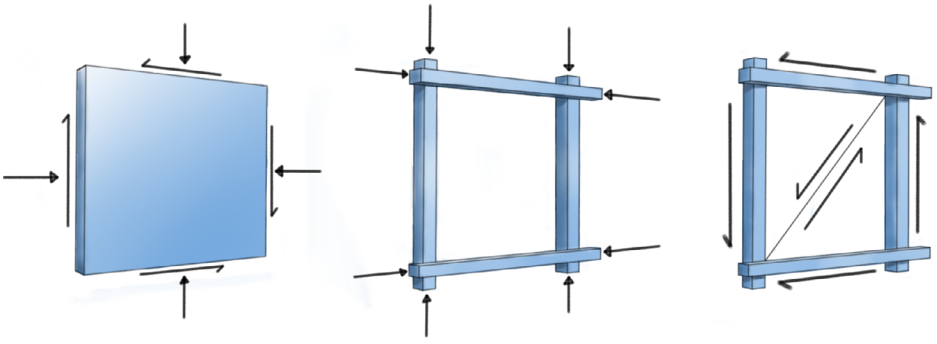


Figure 18: *Left:* The Continuous shell element carries loads through in-plane stressed equally distributed over its thickness. *Middle:* A gridshell element without diagonal reinforcement. The elements resist loads through axial forces and out-of-plane bending which results in larger deformations than that of the continuous shell element. *Right:* A diagonally reinforced gridshell element that allows shear forces to be transferred between the edge of the elements.

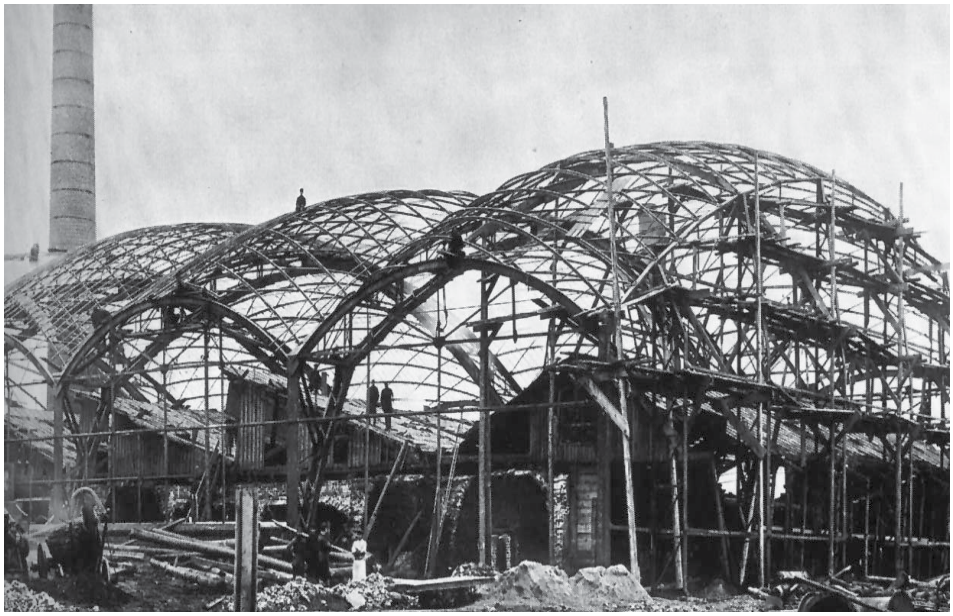


Figure 19: The first double-curved gridshell structure ever build is credited to the famous Russian architect and engineer Vladimir Shukhov. This picture is taken during its construction in 1897. To create such structures Shukhov had to invent mathematical analysis of free form geometry. Shukhov is famous for his hyperploid structures and is considered a pioneer of structural engineering that later led to breakthroughs in curved structures such as shells and tensile structures.

Image source: http://en.wikipedia.org/wiki/Vladimir_Shukhov

Figure 20: The Multihalle in Mannheim is considered a pioneering piece of architecture in terms of form finding. It was designed by Frei Otto in collaboration with Ove Arup & Partners structural engineers. The grid is build of a double-layered pinewood laths. The grid was build flat on the ground and then bent into its final shape by raising the grid and locking it in its supports. The shape of the grid was found using physical modelling. The physical model was a net of hanging chains to determine the structural geometry.

Image source: http://shells.princeton.edu/Mann2.html

Figure 21: This curved free form gridshell structure is located in Warsaw, Poland and stood finished in 2007. It is designed by architects The Jerde Partnership International and engineered by Arup. The gridshell functions as roof covering a shopping area. The gridshell is composed of steel elements and triangular glass panels. The shape of the gridshell and the triangulaztion of the mesh was optimized using form finding which will be discussed in detail in chapter 3.4

Image source: freeformstructures.com

3.3.4 Structural verification of gridshell members using FEM

Gridshells carry loads through normal axial- and bending stresses in the structural members, where axial stresses are typically dominating. Structural material is discretized into elements, so that applied loads cannot be transported to the supports in straight lines like in continuous shells. This can be the cause of large deflections if there is no diagonal stiffening of the gridshell cells when compared to regular continuous shells [Hoefakker, 2010].

Figure 22: Member stress distribution is a combination of normal stresses due to axial loads and bending stresses due to out of plane bending. In order to best utilize the member cross-section, bending stresses should be minimized.

Gridshells make good cases for finite element analysis (FEM) because of their structural complexity. FEM-analysis can be used to calculate displacement, stresses and member stability. Gridshell structures often have constrained joints, unlike truss-structures with pin-joints. This causes every node in the structure to have six degrees of freedom when reviewed with a linear FEM analysis [Dimčić, 2011]. The six degrees of freedom translates into three axial forces and three moments for the x-, y- and z-direction. The gridshell elements resist these forces through their moment of inertia for bending and cross-sectional material for axial forces.

The bending- and normal-stresses in the gridshell members have to be reduced in order to design a structure that is good at resisting external loads and internal forces. Reducing stresses in the structure will result in a more economical solution because less material is required. Since slender structural elements are better at resisting axial forces than bending forces, the geometric arrangement of structural elements should be made so bending stresses are minimized as much as possible. This geometric optimization can be done using physical form finding as described in chapter 3.4. The out-of-plane bending capacity can be greatly increased, by improving the moment of inertia of member cross-section, if bending stresses are unavoidable. Improving moment of inertia is achieved by increasing member height or applying an extra layer of members to create a double-layered gridshell structure under the expense of more material and a more complex structure.

Gridshells are built successfully with a large variety of materials. Equal for all of them is that they are traditionally analyzed using FEM. The stresses found with FEM-analysis, can be evaluated using different yield criteria based on material modelling such as Trescas or von Mises. They can also be compared to the standards found in Eurocodes.

The nature of the gridshell structure will cause loads to be concentrated at structural joints This causes axial- and bending extremes to be found at the end of the structural members. The ends need to be structurally verified to ensure that member failure does not occur for selected load conditions.

Traditionally von Mises's yield criteria is applied to calculate member utilization of ductile materials such as steel structures. Von Mises's yield criteria is based on the theory that a material will maximize its resistance against any deformation [Hagsten & Nielsen, 2010]. The elastic energy in a material is the potential strain and the material starts to yield once it reaches its elastic limit [Dimčić, 2011]. The FEM software used in this thesis will be Karamba3D, which is to be discussed in chapter 4.4. Karamba3D uses the structural verification method applied in Eurocode 3 DS/EN-1993 section 6.3.3 to calculate the utilization of beam elements.

3.3.5 Appearance of gridshell structures.

Gridshells are built up of many structural members, joints and panels. If the mesh layout of the grid is not following a consistent proportional pattern, it can result in an aesthetically unpleasing structure [Dimčić, 2011]. People are remarkably good at spotting and disliking deviations in a pattern. This becomes an important fact when designing the mesh layout for the gridshell structure. Any irregularities should be avoided to make the visuals of the structure more pleasing to the eye and any established grid patterns should be consistent. This does not mean that there is no room for irregularity. Consistent irregularity can also be a pattern. Something that is demonstrated many times in nature. A recurring consistent irregular, yet for the eye pleasing, pattern found in nature is the voronoi structure see figure 24.
The search for consistent patterns can be translated into a set of gridshell development rules that should be followed in order to create pleasing structural appearances.

1. The grid pattern should be followed consistently without irregular deviations.
2. The structural members should try not to have large deviations in length and cross-section. Variation in cross-section should be implemented in continuous paths to generate a sense of smoothness.
3. Structural panels, or cells, should share the same polygon type such as triangles, quadrilateral, pentagons or hexagons. If creating an irregular grid, it should vary with a consistent pattern as seen in the voronoi structure.
4. Panels should be small enough to allow a proper tessellation of the shell surface. Large panels will make kinks in the structural guide lines and cause optical disturbance.
5. Some variety in panel size is to be expected. The degree of variance should be minimized so that panels share the same level of scale.
6. The amount of members connected in joints should not differentiate. There should be a consistency in the angle between members in joints.

These rules will be applied in chapter 4.3. where the grid generation algorithms will be developed. The dynamic relaxation technique explained in chapter 3.5 can be applied to smooth grid line paths making for a visually better grid see figure 41.

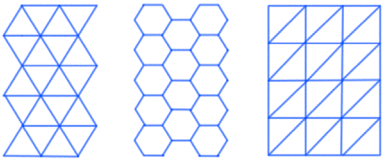
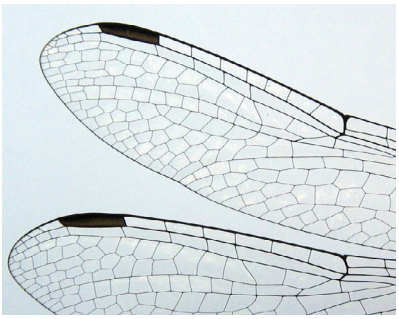


Figure 23: Typical gridshell patterns. *Left:* Triangular grid. *Middle:* Hexagon grid. *Right:* Quadrilateral grid with diagonal bracing.



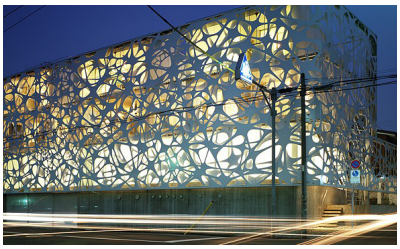


Figure 24: Voronoi patterns. *Top:* Cracks in clay. *Middle:* The internal structure of a dragonfly. *Bottom:* Voronoi pattern applied to a facade in Kitamagome, Japan. The project dubbed Airspace Tokyo was made by a collaborativ effort of architect Thom Faulders and digital technologist Sean Ahlquist

Figure 25: Structural guide-lines maintain a sense of smoothness in the grid pattern. Here showed in an example from MyZeil shopping mall in Frankfurt. Guide lines must not be broken in order to achieve a pleasing aesthetic design. Image source: http://www.freeformstructures.com/

3.3.6 Surface curvature

The free form curved surfaces that will be discretized into gridshells can be described by cross-sectional curves. Each point on the surface can be described by an infinite number of intersecting curves with different radius of curvature. The two curves, in any given point of the surface, that posses the largest and smallest radius of curvature are named the principle curvatures κ_1 and κ_2. The Gaussian curvature is the product of the two principals of curvature $\kappa_g = \kappa_1 \cdot \kappa_2$ [Hoefakker, 2010].

The generation of surface geometry used in this thesis is explained in detail in chapter 4.1

$\kappa_g < 0$: When the principle curvatures are opposite the resulting surface is called anticlastic.

$\kappa_g > 0$: When the principle curvatures are both positive the resulting surface is called clastic.

$\kappa_g = 0$: When one of the principle curvatures is zero the resulting surface is called cylindrical

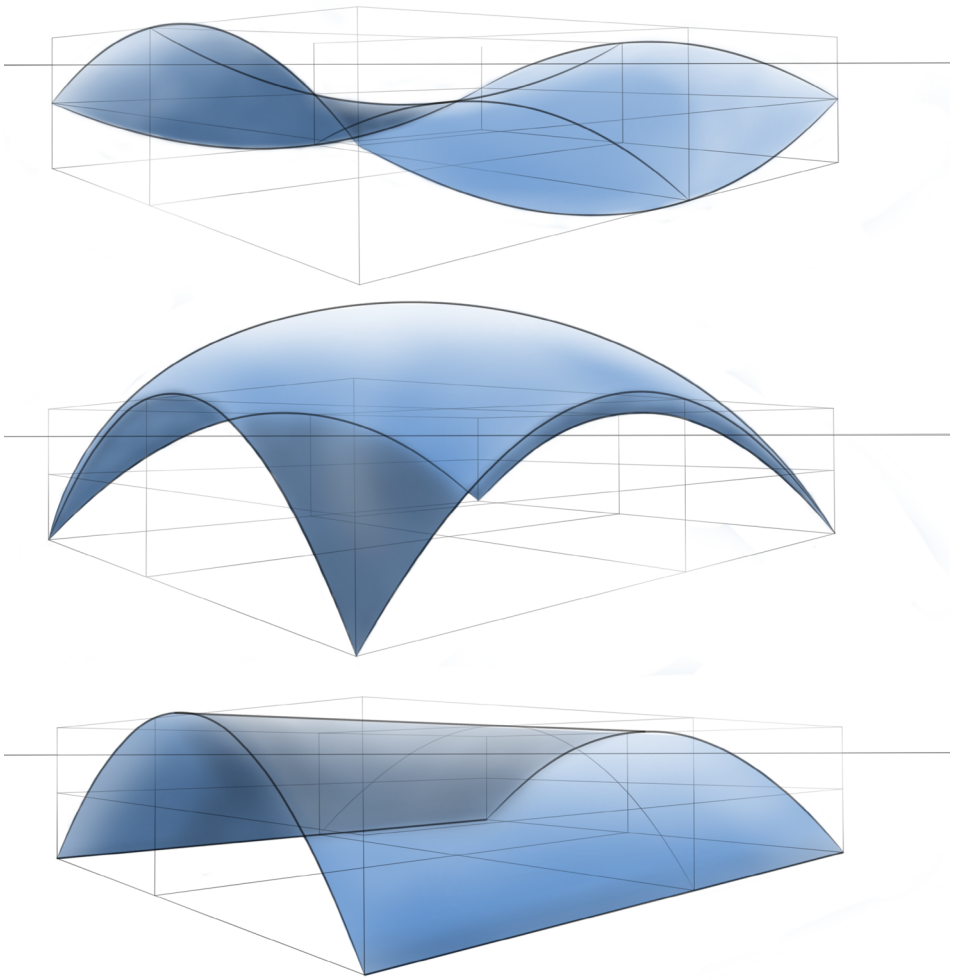


Figure 26: Three types of Gaussian curvature is shown above.

Design goals derived from the previous passage:

1. The tool should tackle the complex nature of the gridshell geometry and make it more accessible.
2. The tool should be able to find a shape that possesses membrane action because of its structural capability.
3. The supports of the structure should be made membrane compatible to achieve the most efficient structure.
4. The tool should be able to generate a vide variety of grid-geometry on a surface.
5. The grid should be customizable to allow freedom of choice to the designing team.
6. Some sort of diagonal bracing should be able to be introduced to increase the structural efficiency of the gridshell.
7. The tool should be able to dynamically use FEM to calculate stresses and displacements in the gridshell structure.
8. The grid generation should aim to make consistent patterns that are visually pleasing.

3.4 Structural Form Finding

> "…At its best [structural engineering design] is an art... it is primarily about the choice of form. The forces on that form and the analysis of its behaviour follow. " - Ted Happold

The desire for an optimal structural shape is always present when designing a structure. This search for an optimum in shape is normally called form finding. When dealing with free form architecture there are fewer restrictions seeing that virtually any form could be applicable as long as it fits the design requirements.
In architecture, form finding is the search for a shape that fits the desired appearance, functionality and climate of the interior and exterior spaces. Architectural form finding is a somewhat subjective affair as mentioned in the introduction to the chapter 3.1. Good architects design their structures so that functionality and appearance will fit the cultural and social needs of the people that are to use the structure.
For engineers, form finding, is about analytically optimizing the shape of the structure to obtain optimal structural behaviour. The engineers perform structural form finding and optimization with a pragmatic approach to the solution. As mentioned earlier, form finding for engineers, becomes a case of finding absolute adequate structural elements that fits the overall concept made by the architects.
Obviously a conflict in shape will arise from structural and architectural form finding because they have different optimums and thus different forms. Architects will work after the principle of "form follows function" where engineers will use "form follows force".
It is important to mention that optimal is not the same as perfect. Structural form finding is about finding a structural solution that is better than the initial input.
As mentioned in chapter 3.1, the ultimate goal of structural and architectural design, and thus form finding, must be to unify the requirements of both desired shapes and generate a collaborative solution that fulfils the goals of both. This must be done finding a balance between the two approaches to form finding and respect the values both fields contribute. The goal of this thesis is not to provide a specific proposal of this symbiotic design process but to develop a tool that will aid the collaborative form finding design phase.
The goals of the form finding process have to be discretized into specific criteria that can be rationally and mathematically optimized. This immediately leads to a problem. Because, how can one discretize important subjective terms such as aesthetics and functionality. This is something philosophers have been trying to answer for centuries.

> "Beauty is no quality in things themselves: It exists merely in the mind which contemplates them; and each mind perceives a different beauty. One person may even perceive deformity, where another is sensible of beauty; and every individual ought to acquiesce in his own sentiment, without pretending to regulate those of others." - (Hume 1757, 136) [Sartwell, 2012]

It turns out that discretizing aesthetics cannot be done collectively but only individually. One cannot objectively value something that is entirely subjective and get the same result as another person. Structure functionality can to some extent be broken down into geometric criteria, such as required floor area, required free height, lighting, indoor climate, acoustic performance etc.
The development goal of the form finding tool must to create something with high informa-

tive value, so that the things that can be rationalized, such as a span length, amount of material, member selection are mathematically decided while still visually presenting the subjective aesthetic values of the design. It will then be up to the designing team of engineers and architects to create a design using the form finding that fulfils both structural, functional and aesthetical requirements.
The above assertions lead to the conclusion that the form optimization of a structure is closely related to the optimization of the structural design seeing how it can be objectively discretized. A common focus of structural optimization is to achieve a minimal stresses and greatest span with as little material as possible leading to lower costs. Especially when dealing with free form structures this can be achieved by mimicking principles found in nature as described in chapter 3.2. Previously this structural optimization was done using physical modelling, which will be the subject of the next chapter.

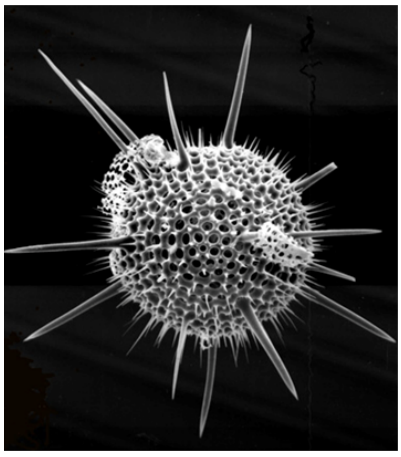


Figure 27: A mineral skeleton structure produced by the amoeba Radiolarian. The structure is highly optimized to provide the necessary shell functionalites with a minimal amount of material.

A building with an efficiently performing indoor climate could also be the desired goal for the form finding process. Indoor climate optimization can also be approached with form finding. Orientation of the building, window orientation, sun shading and natural ventilation can be included in the form finding process to produce structures that perform great climatically without the need of artificial climate control. This optimization of indoor climate is not within the scope of this thesis.

3.4.1 Physical modelling

Physical modelling is at the core trying to mimic the force transferring principles of nature to find optimal shapes. Nature is unparalleled in creating efficient structures with a minimal amount of material and energy.

As mentioned in chapter 3.2, in most cases, the dominating force to man-made structures is vertical gravitational forces. Therefore it is often the main concern of form finding, to find an optimized form, so that the resulting structure will resist gravitational loads in a more efficient way. That is why structures found with form finding are excellent roof structures.
Tension structures such as cable- fabric membrane structures are easily modelled physically. When flexible membrane structures are loaded, the membrane will take on a new form unknown in advance (Lewis 2003). This new form will resist gravitational loads in the most energy efficient way given the boundary conditions. This behaviour can be modelled in advance with an elastic piece of fabric or with Soap film that will provide a minimal energy surface between rigid supports. The German engineer Frei Otto is famous for having used these principles when form-finding lightweight structures such as the Olympic stadium in Munich.
It is also possible to model compression structures physically. The general principle is to create the form using a model in tension under gravitational loads. Compression shells can be seen as the inverse of pneumatic membranes and soap bubbles. Soap bubbles are real-life examples of the mathematical problem of minimal surfaces.

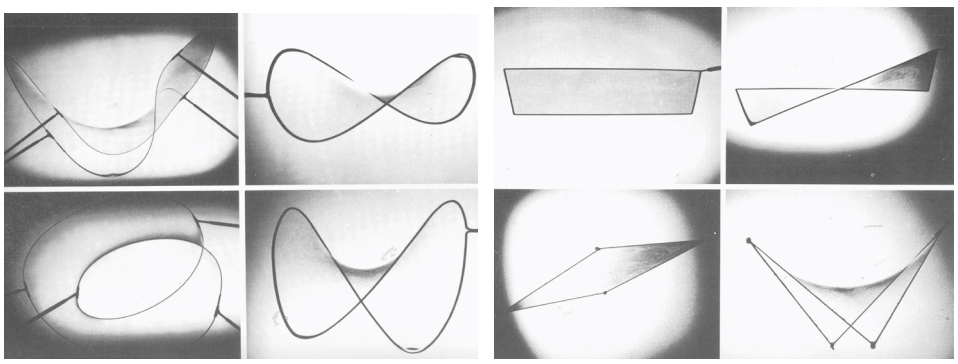


Figure 28: Frei Otto did extensive research of the potential of soap film form finding. Soap film creates minimal surfaces when extended between restrictive edges. Membrane structures posses the same ability and can thus be modelled with soap film. Today this can be done computational with computers to get precise results. Image source: [Otto, 1965]

Rigid arches can be modelled using weighted thread to find the catenary (Latin for chain) or funicular shape. This is more commonly referred to as a hanging chain model. Upon the chain acts a uniform gravitational load. The chain is suspended at its ends. The resulting shape of the chain will be a result of the strain in the chain caused by the tension, as the chain lacks moment capacity. When the shape of the hanging chain is inverted, a shape is found where only compression forces exists.
The resulting inverted form is optimized for the applied load under the form finding. An arch form found with an uniformly distributed load will excel at resisting uniformly distributed loads once inverted. The English natural philosopher and architect Robert Hooke (1635 – 1703) was one of the first to explain how the form compression arch structures could be modelled using a hanging chain. This was done after he discovered the law of elasticity in 1660, which describes the linear proportionality of tension and extension in an elastic spring. He is famous for having published an anagram stating; "As hangs the flexible line, so but inverted will stand the rigid arch.".

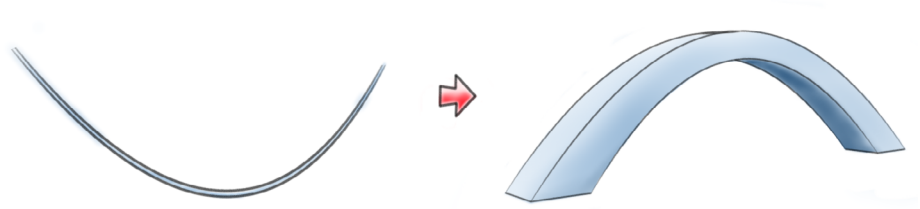


Figure 29: The catenary shape can be inverted to find the form of rigid arches.

Architects such as Antonio Gaudi and Frei Otto understood how to apply the hanging chain inversion principle to help them find the geometry of some of their most famous structures.

> The straight line belongs to man, the curved line belongs to God. - Antonio Gaudi

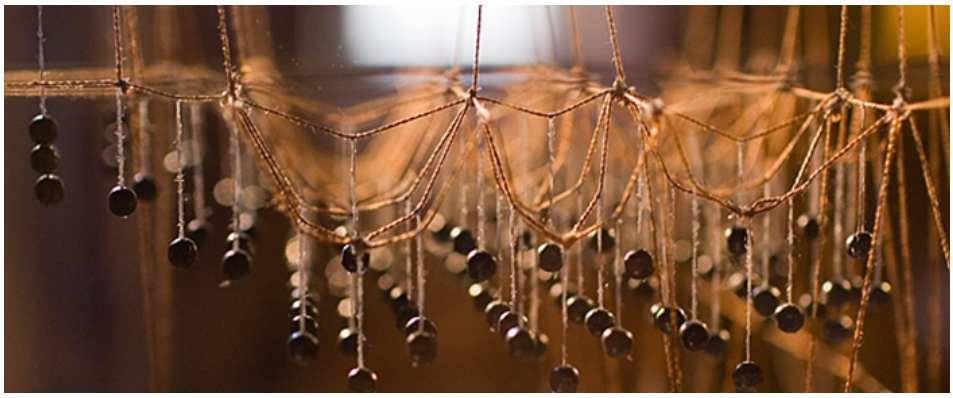


Figure 30: A hanging chain model from the Gaudi museum in Barcelona. The hanging chain is used to physically model the geometry of a catenary arch. When the geometry is inverted a rigid arch where the internal forces only consist of axial compression is found. Image source: Unknown.

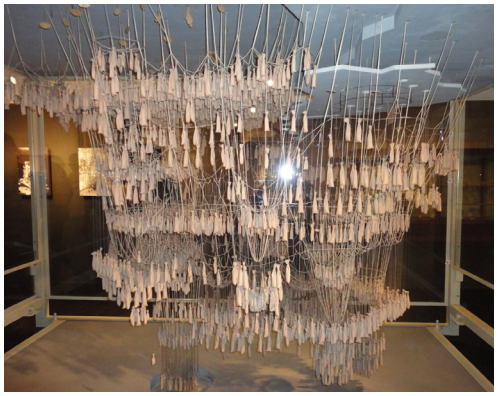


Figure 31: Gaudi used string models with applied weights to generate the form of some of his works. *Left:* A replica of the model used to find the form of the famous church La Sagrada Familia in Barcelona. Image source: http://schetzelsintheuk.wordpress.com,

Figure 33: Right: The structural system replicates the form found with the model and in doing so benefits from mainly axial forces.
Image source: http://commons.wikimedia.org/

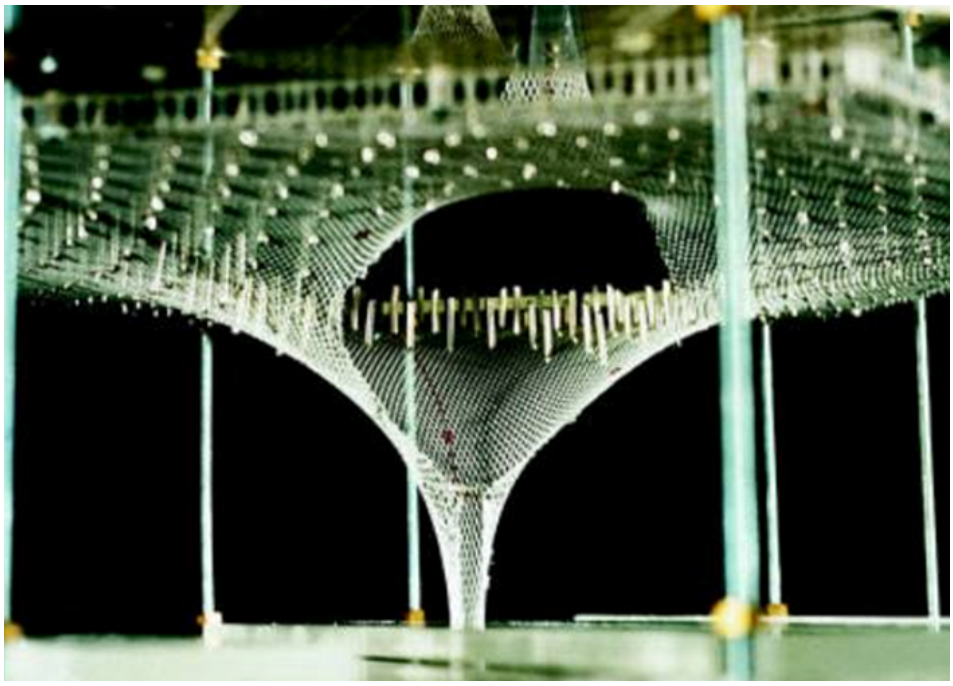


Figure 32: Frei Otto used physical models to aid in the form finding of the new train station in Stuttgart. The project is still in development after the death of Frei Otto.
Image source: http://architecturehabitat.blogspot.dk

3.4.2 Computational form finding

Form finding can be made more efficient when done numerically. Computational form finding is at the very basis an optimum problem. The solution to the problem can be found numerically with iterative calculations. In chapter 3.4 it was described how form finding is often related to structural optimization, so the optimum problem be solved is about finding a structural optimum. The starting point of the optimization process is often an initial shape that fits the functional requirements but statically does not fulfil the optimization criteria such as amount of material or size of deflections. The iterative form finding process is based on numerical algorithms aimed to optimize structural behaviour. The algorithms optimize the geometry of the structure in the shape of geometric updates until the overall structural form is in static equilibrium. The outcome of the form finding process is a transformed shape that is statically more efficient than the initial shape. The outcome is presented visually during the form finding process and after reaching an equilibrium state it can be exported to other platforms.

Computational form finding is a powerful tool when dealing with the optimization of membrane structures because of their geometric nonlinear behaviour. Membrane structures will experience non-linear deflections when loadings are large enough and are therefore hard to solve with traditional mechanical calculations. The goal of membrane structure optimization is to reduce bending stresses in the structure. Doing so will greatly reduce material required and allow for greater spans with less deflection.

The most common numerical form finding methods are dynamic relaxation and force density [Douthe & Baverel, 2006]. Dynamic Relaxation will be used in this thesis seeing how it can be developed without complex matrix calculations making real-time optimization fast on regular computers. It also allows optimization of highly complex, and thus highly nonlinear, structural systems making it perfect for gridshell form finding.

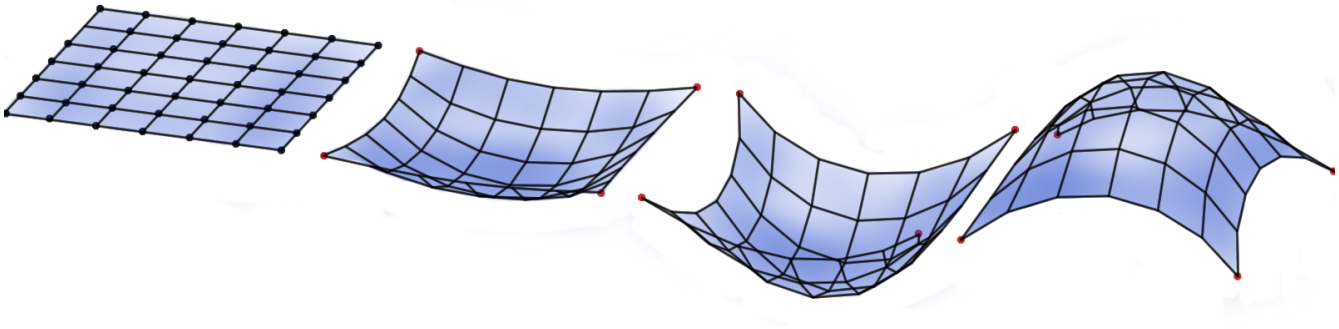


Figure 34: The process of dynamic relaxation is used to imitate the hanging chain model used in physical form finding. The geometry is discretized into structural elements and joints. Force is applied in the joints and the structure deforms like a hanging chain would. When the structure is inverted a compression-only structure is found.

Dynamic relaxation has been widely applied in structural form finding of gridshells in both past and present. Because dynamic relaxation is a geometrical optimization of the shape of the structure, it does not matter if the structure is made of steel or a composite material like wood or fibreglass seeing how the goal of the process is to reduce bending stresses to a minimum and let applied loads be resisted almost entirely through axial stresses.

Dynamic relaxation can be thought of as the numerical process of simulating the hanging chain models made by Isler, Otto and Gaudi. The process finds the funicular shape of a complex geometry based on the loads applied during the form finding process. A structure almost entirely rid of bending stresses can be found, once the inverted shape resulting from the dynamic relaxation process is utilized. See more in chapter 3.5.

The benefits of computational form finding can be summarized as the following:

1. Much less time consuming overall.
2. Models can be altered quickly in contrast to physical models.
3. Geometry is exact and contain a precise level of detail allowing for exact calculation and fabrication.
4. Geometry found in computational form finding can be applied in other platforms such as FEM-software and drawing-software.

Traditional modelling should however not be left out of the design phase. A computational model will not provide the same sense of consistency as a physical model. Especially when dealing with a feeling of scale and context of a design. With the introduction of less expensive 3D printers, physical models of the computed form can be printed out for further inspection.

Design goals derived from the previous passage:

1. Since aesthetics cannot be discretized, the tool should accurately visually represent the form found and be capable of inserting the structure into a contextual model.
2. The tool should respond fast, so many structural models can be created in a short amount of time.
3. The tool should imitate physical form finding by applying the method of dynamic relaxation so that the result is a structure in compression and bending stresses are minimized.
4. The tool should be able to communicate with industry leading software platforms.

3.5 Dynamic Relaxation

Understanding the underlying theory of dynamic relaxation is required in order to program a form finding tool that is capable of performing real-time dynamic relaxation. By personally developing the dynamic relaxation script, a full understanding of the underlying capabilities is ensured.
Alistair Day derived the dynamic relaxation method in 1965. The theory that will be used in the development of the form finding tool in this thesis is inspired by the computational steps suggested in [Wakefield, 1999].

> "This programming system method, known as dynamic relaxation, as originally invented by Alistair Day, one of the team members, is a very powerful tool for examining the behaviour of nonlinear, geometrically variable systems." - [Rice, 1996]

The basis methodology of dynamic relaxation is a numerical discretization of the structural form into a particle-spring-system. The system is built by a grid of translational spring elements connected by nodes. The mass of the spring elements is lumped at the nodes. Loading is applied to the nodes which causes the system to oscillate around its equilibrium position. The movement of the system is damped with viscous or kinetic damping. The spring element grid will compose a form of only axial forces where all the spring elements are in equilibrium, once nodal movements have died out due to damping. At this point it is important to mention that the equilibrium state found is related to the applied loads, so often loads resembling the governing load case will be chosen.
The mass of the nodes are fictitious and only used to find the appropriate form. By keeping node mass fictitious the mass can be selected to improve the speed of which the dynamic relaxation process converges. The physical basis of the dynamic relaxation method is based on a iterative step-by-step solution, for a small time increment, of Newton's second law of motion. In each stage of the iterative calculation the resultant force in every node is calculated and moved using numerical integration of the equations of motion.

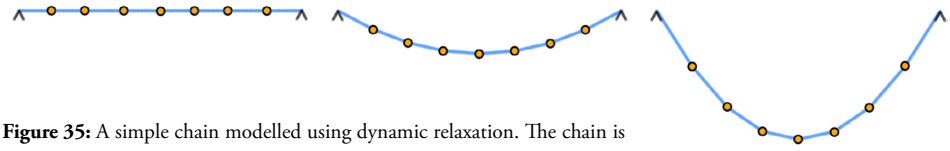


Figure 35: A simple chain modelled using dynamic relaxation. The chain is discretized into spring elements connected by nodes. Loading is applied to the nodes that deform the geometry to a catenary shape.

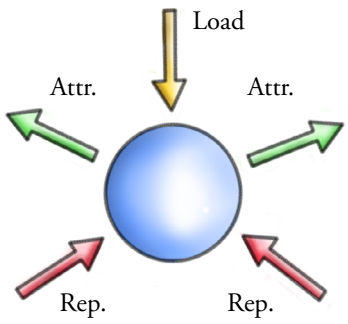


Figure 36: Forces of attraction and repulsion acting on a node in the particle-spring system as result of an applied load.

3.5.1 Basic Equations

In order to setup the basic equations required to perform dynamic relaxation the equilibrium of a node in the particle system is considered.
Equilibrium occurs once there is balance in applied loads and link force of repulsion/attraction between the spring elements.
Newton's second law of motion states that the acceleration of the node is directly proportional, and has the same direction, as the sum of the forces acting on the node. The mass is inversely proportional to the acceleration.

$$Force = Mass \cdot Acceleration \tag{1}$$

If only forces and movement in the x-direction is considered, then the second law for a given time t can be written as

$$F_x(t) = M\, a_x(t) \tag{2}$$

The derivation of the basic equations used in dynamic relaxation are the same for y- and z-direction.
Hooke's law stated that the force needed to extend or compress a spring is directly proportional to the deformation of the spring. The constant ratio between the force and deformation is defined as the stiffness of the spring.

> "Ut tensio, sic vis" / "As the extension, so the force." - Robert Hooke

$$F_{sx} = K_{ix}\delta_{ix} \tag{3}$$

The equations are derived from a spring-mass-damper system with displacement in the x-direction as the only degree of freedom. This is visualised in figure 37.

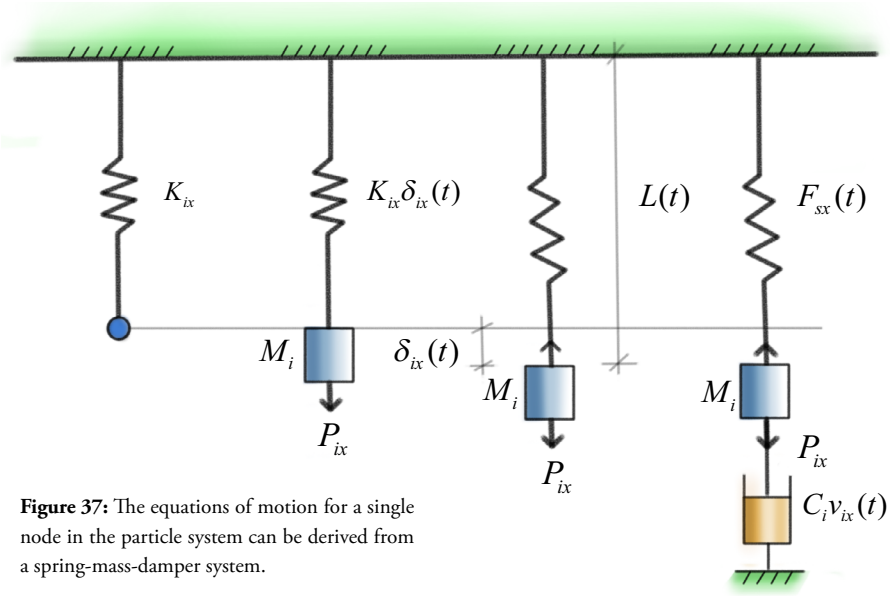


Figure 37: The equations of motion for a single node in the particle system can be derived from a spring-mass-damper system.

In the spring-mass-damper system the forces acting in the x-direction on the node with index i, visualized as the blue mass in figure 37, are the following.

- P_{ix}: Constant applied load at the node i in the x-direction.
- $C_i v_{ix}(t)$: Viscous damping of the system in the x direction at the time t.
- $F_{sx}(t)$: Spring force acting on the node in the x-direction at time t.

The system can be described using Newton's second law and Hooke's law. The resulting equation is then

$$P_{ix} - K_{ix}\delta_{ix}(t) - C_i v_{ix}(t) = M_i a_{ix}(t) \tag{4}$$

Where the terms represent the following:

- M_i: Lumped fictitious mass at node i in the x-direction.
- $\delta_{ix}(t)$: Current displacement in the x-direction of the node i at time t.
- K_{ix}: Stiffness of the spring.
- C: Viscous damping constant of node i.
- $v_{ix}(t)$: Velocity in the x-direction of node i at time t.
- $a_{ix}(t)$: Acceleration in the x-direction of node i at time t.

Equation (4) can be rewritten into

$$R_{ix}(t) = M_i a_{ix}(t) + C_i v_{ix}(t) \tag{5}$$

Where $R_{ix}(t)$ represents the resultant force acting upon node i in the x-direction at the time t. This resultant force is a combination of applied load and the spring element forces from surrounding springs.

Equation (5) can expressed in central finite difference form for an diminutive time step Δt by performing a double numerical integration of the acceleration. The central difference method is often used in computational dynamics to compute the numerical approximation of the derivatives in Newton's equations of motion. The result of the double numerical integration is

(6)

$$R_{ix}(t) = M_i \left(\frac{v_{ix}(t + \Delta t/2) - v_{ix}(t - \Delta t/2)}{\Delta t} \right) + C_i \left(\frac{v_{ix}(t + \Delta t/2) - v_{ix}(t - \Delta t/2)}{2} \right)$$

If the damping constant C is replaced with damping proportional to the mass the equation can be written as a recurrence equation for nodal velocity at time $t + \Delta t/2$

(7)
$$v_{ix}\left(t + \frac{\Delta t}{2_i} \right) = A \frac{\Delta t}{M_i} R_{ix}(t) + B \; v_{ix}\left(t - \frac{\Delta t}{2} \right)$$

Where the constants A and B are given by

(8)
$$A = \frac{1}{1 + C\frac{\Delta t}{2}}$$

(9)
$$B = \frac{1 - C \; \Delta t/2}{1 + C\,\Delta t/2}$$

Here C is the damping factor pr. unit mass. The geometry can be updated projected to the time $t + \Delta t/2$

(10)
$$x_i(t + \Delta t) = x_i(t) + v_{ix}(t + \Delta t/2)\,\Delta t$$

Equation 7 and 10 can be applied to update the node geometry of all three directions.
After the node geometry has been updated the new resultant spring link forces can be calculated.

(11)
$$R_{ix}(t + \Delta t) = P_{ix} + \sum \left(\frac{F}{L} \right)_m (t + \Delta t)(x_j - x_i)(t + \Delta t)$$

The summations is the updated link forces from spring links m connected from nodes j connected to node i. The force in the links is denoted F_m and the length of the links denoted L_m.

3.5.2 Damping and convergence

Damping is required to bring the nodal oscillations of the particle system to rest at the equilibrium position. This is normally described as the system converging to its equilibrium state. The rate of convergence is dependant on selection of fictitious nodal mass, damping coef-

ficient and selected time step for the numerical integration. If the time step is too large and damping insufficient, the numerical integration will overshoot the nodal movement and cause geometric instability.

There are two types of damping available to include in the Dynamic Relaxation method, viscous damping and kinetic damping. Viscous damping is mathematically modelled such that it is proportional to the nodal velocity and acting in the opposite direction of the velocity thus slowing the movement. Viscous damping is easy to incorporate in the basic equations that update the nodal geometry, but can cause instability if the initial nodal geometry starts from an inaccurate starting position. This is due to the fact that very large geometric displacements occur if the damping is insufficient during the first iterations of the dynamic relaxation process. The most rapid convergence is achieved when the lowest mode of vibration of the system is critically damped.

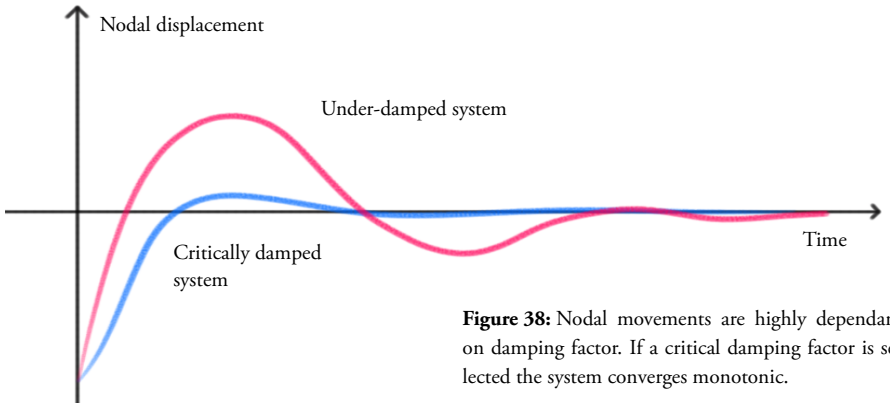


Figure 38: Nodal movements are highly dependant on damping factor. If a critical damping factor is selected the system converges monotonic.

The other form of damping is kinetic damping. In a simple harmonic motion the velocity, and thus kinetic energy, is maximum around the equilibrium position. Kinetic damping works by tracking kinetic energy in nodes based on nodal velocities. When a local peak in nodal kinetic energy is detected by the algorithm the velocity of the node is set to zero and the dynamic relaxation process is initialised again. This continues until the system converges on the final equilibrium position. This form of damping results in fast convergence seeing how it is not necessary to determine a damping coefficient close to critical damping to achieve fast convergence [Douthe & Baverel, 2006].

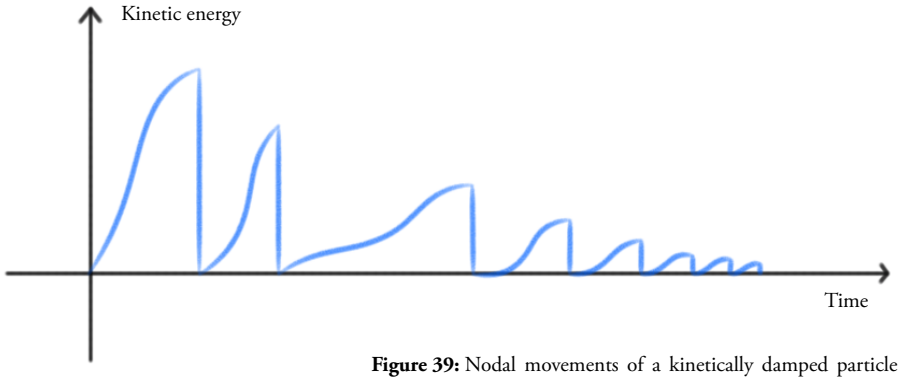


Figure 39: Nodal movements of a kinetically damped particle system. Each peak is a temporary local equilibrium. The system reaches global equilibrium when the energy has dissipated.

Viscous damping will be selected for the dynamic relaxation algorithm. Viscous is very simple to implement in the code. Modern computers are so powerful that the dynamic relaxation algorithm, if written correctly, will converge rapidly even if using a damping coefficient that causes under damping. Using viscous damping also allows the user to change damping in real-time while the dynamic relaxation process is running. Seeing how the dynamic relaxation

algorithm gives a visual feedback of the geometry being updated, the user can alter the damping coefficient to speed up the rate of convergence.

3.5.3 Computation procedure

The following steps make up the iterative computational dynamic relaxation procedure [Wakefield, 1999]:

1. Set residual forces, velocity and kinetic energy to zero.

$$R_{ix}(t=0)=0$$

$$v_{ix}(t=0)=0$$

$$E_{kix}(t=0)=0$$

2. Set resultant forces of each node initially equal to the applied load at the node.

$$R_{ix}(t=0)=P_{ix}$$

3. Calculate forces of repulsion and attraction from springs linked to other nodes and add the net link force to the resultant force .

$$R_{ix}(t+\Delta t)=P_{ix}+\sum\left(\frac{F}{L}\right)_m (t+\Delta t)(x_j-x_i)(t+\Delta t)$$

4. Fixed nodes that represent structural supports have their resulting forces set to zero so they do not move.

$$R_{fixed,ix}(t+\Delta t)=0$$

5. Calculate nodal velocities and update the geometry of each node using eq. (7) and (10).

$$v_{ix}\left(t+\frac{\Delta t}{2_i}\right)=A\frac{\Delta t}{M_i}R_{ix}(t)+B\ v_{ix}\left(t-\frac{\Delta t}{2}\right)$$

$$x_i(t+\Delta t)=x_i(t)+v_{ix}(t+\Delta t/2)\,\Delta t$$

6. Repeat procedure at step 3 until convergence of the particle system is obtained.

The procedure described has to be scripted into programming language. The resulting algorithm will then be able to perform the dynamic relaxation process. The algorithm is described in much further detail in chapter 4.2. In Chapter 4.2 it will also be described why a different numerical integration scheme was chosen over the one derived in equation (10). The chosen integration scheme, named Velocity Verlet, results in much more stable convergence and is given as

$$\vec{x}(t+\Delta t)=\vec{x}(t)+\vec{v}(t)\Delta t+\frac{1}{2}\vec{a}(t)\Delta t^2 \tag{12}$$

3.5.4 The relaxed form

The shape of the geometric output of the dynamic relaxation process is mostly effected by spring stiffness. The resulting shape of the grid will be more flat if a high spring stiffness is selected and more curved if a low stiffness is selected. There is an infinite amount of possible forms that can be derived for the form finding, but one thing goes for all the structures; they will all posses and overall shape where the nodes are in equilibrium for the applied load. It is then a matter of designing supports correctly that allow for membrane action mentioned in chapter 3.3.1 to create an effective structure.

Dynamic relaxation will also contribute to the appearance of the mesh layout of the gridshell structure. The relaxation process relaxes the grid and while doing so makes the appearance more consistent and smooth. This smoothing of the mesh pattern will contribute to a more pleasing visual structure as described in chapter 3.3.5.

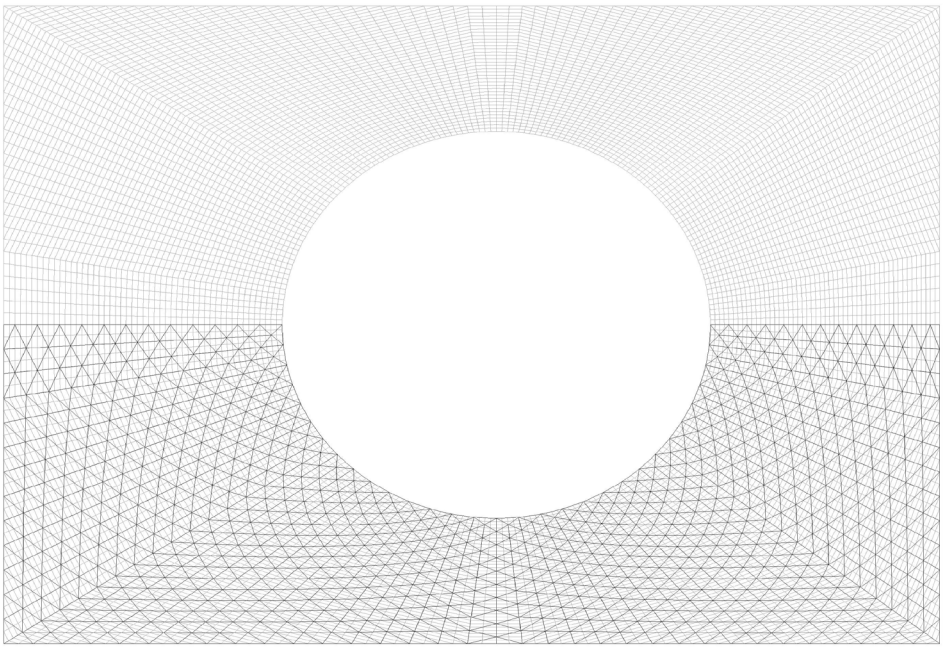


Figure 40: The un-relaxed gridshell mesh layout for the courtyard roof of British Museum. Before relaxation there are kinks in the structural lines making the grid less efficient. Image source: [Williams, 2001]

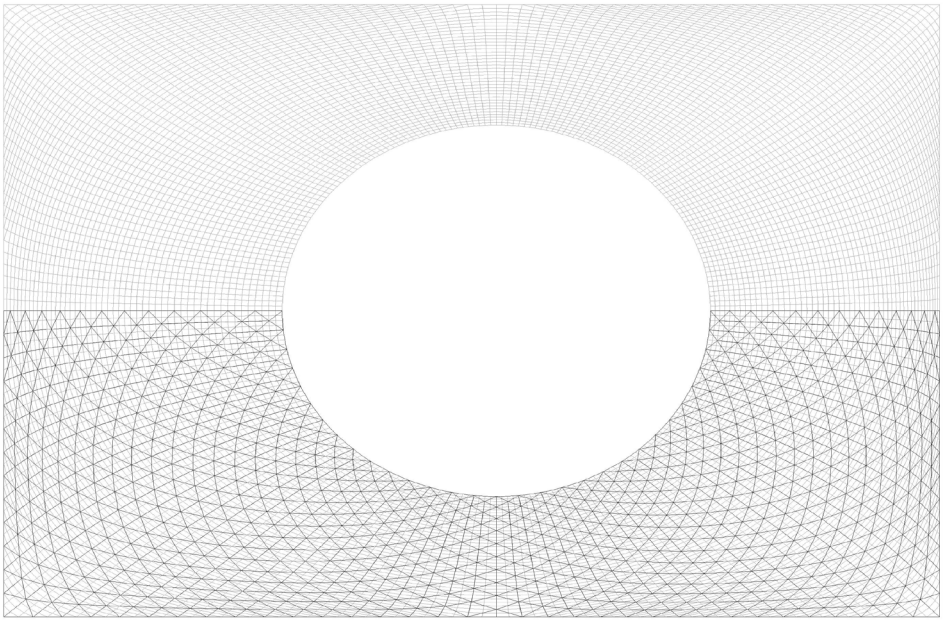


Figure 41: The mesh after dynamic relaxation shows smooth continues guide lines in the structure making it more continuous. Bottom: Image source: [Williams, 2001]

Figure 42: The court yard glass and steel gridshell after completion. The project was made in collaboration with Foster And Partners architects and Buro Happold engineers. Image source: [Williams, 2001]

Design goals derived from the previous passage:

1. The dynamic relaxation script must posses customize-able real-time updating parameters for damping, spring stiffness and nodal mass to achieve faster convergence. The parameters will also be used to change the form.
2. The input of the dynamic relaxation component should be a grid of lines connected in nodes. Nodes that act as supports should be specified in the input.
3. The shape of the input grid should not matter. This is to ensure freedom of choice for the designing team.
4. The tool must follow the iterative computational procedure presented by Wakefield.

Figure 43: The platform used to develop the tool will be Grasshopper. This choice is discussed in much further detail in chapter 4.1.1.

4. Development of a gridshell form finding tool

The following chapter is focused on taking the key requirements found in the previous chapter and convert them into a working computational form finding tool.
The dynamic relaxation algorithm is the core of the tool. It will be programmed using the C# programming language.
In the previous chapter there was a high emphasis on allowing the designing team to model gridshell structures based on user specified input parameters. Grasshopper, a computational platform that is build around parametric modelling, is selected for the platform of the program which will be described in this chapter. Grasshopper is also compatible with the C# language and can visualize the results of the dynamic relaxation component.

4.1 Parametric modelling as a basis for geometric form finding

"Think. Do the rules. Code. Hit enter." - Miloš Dimčić

Parametric design as a principle is a very broad term and can be applied in many different fields of work. A simple example of computational parametric design is a common spread sheet. Once the spread sheet has been properly set up the output will change given the input parameters and make for a quick and effective tool.
In this thesis, parametric design, will refer to the case of algorithmically generating geometry using a set of varying user specified input parameters. Before describing parametric design and generative geometry in detail a short explanation of the advantages of parametric design will be given.
As discussed in chapter 3.2; modern contemporary architecture should not be, and is not, constrained to straight lines. As a result of this, free form surfaces are more and more often used in modern architecture. The free forms are build of surfaces with double curvature. Double curvature surfaces lead to high structural complexity. In order to construct these free form surfaces they must be discretized into structural elements such as members, joints and facade panels. The structural complexity is a result of the fact, that in truly free form structures every structural element is unique. In order to fabricate and assemble all these structural elements, exact and unique specifications for every element are needed. Specifications in the shape of plans, drawings and instructions for the machines that produce the elements. This level of detail is needed for everything, down to the tiniest level, every screw, every weld must be specified seeing how it is unique.
Even though it is possible, it would be economically unwise and very time consuming to hire an army of technical drawers to produce plans for everything. Not to mention the horror of facing a change in overall form, something that happens numerous times in a unrestricted design process.
In order to efficiently realize a complex free form structure a high amount of automation is required. Automation that follows a set of rules set up by the designing team. That is what Dimčić means when he expresses; "Think. Do the rules. Code. Hit enter". The automation should be programmed with geometric algorithms that automatically produce plans for production and assembly for every element. Everything should be controlled by parameters so that changes made in the design process will automatically change the output plans and drawings. The quote by Miloš Dimčić can be translated into four guidelines that can be followed when creating the parametric form finding tool.

1. **Think**: Engineers and architects should work together and derive the parameters necessary to create a strong design. Parameters that when included in a set of rules control all aspects of the geometry of the structure.
2. **Do the rules:** Derive how the parameters interact with each other to generate geometry with a parametric model. Create rules that optimize the structure based on the parameters. Optimization may include structural design, aesthetics or indoor climate.
3. **Code:** Translate the rules and parameters into programming language the computer can interpret. The resulting code is a set of generative algorithms that when fed with parameters produce geometry that follow the rules made by the designers.
4. **Hit enter**: The output is computed instantaneously and changes made in input immediately reflects in the output design.

In generative design, parametric design can be thought of as a sort of geometric inheritance. A concept where all the geometry forming a structure is connected such that if one aspect of the design is changed it will have a ripple effect through the rest of the geometry. This allow for a great amount of freedom in the design phase seeing how it is now longer a problem to make changes to the form as everything changes automatically.

4.1.1 Grasshopper

Even though Grasshopper initially can seem intimidating and overly complex it is one of the most powerful and intuitive parametric design tools available. Grasshopper is a graphical generative algorithm editor that allows the user to parametrically create geometry using the 3-D engine from the three dimensional modelling CAD-software Rhinoceros. The benefits of using Rhinoceros as the underlying engine, is its ability to generate surfaces with double curvature that are infinitely accurate through mathematical descriptions explained in 4.2.1. It also allows for multi-platform communication with other types of CAD- and structural calculation-software. Grasshopper has a striving community that helps improve and develop the possibilities of the software.

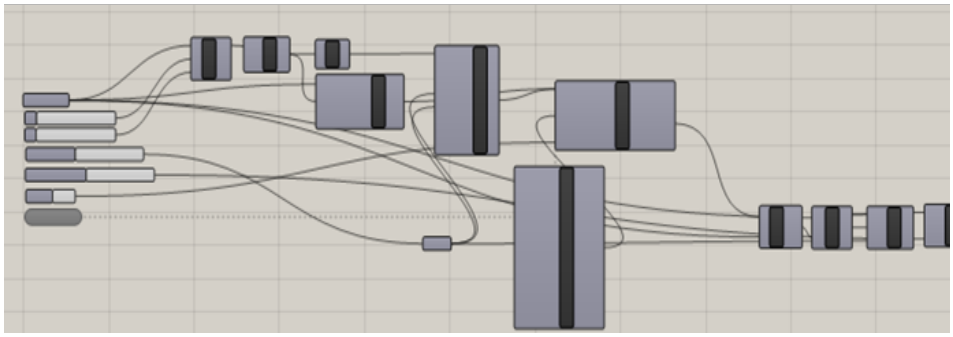


Figure 44: The graphical user interface (GUI) of Grasshopper. Geometry in grasshopper is generated with small algorithmic components. The components are wired together to create the overall design. The geometry components are fed with input parameters that can be changed with a rippling effect through the geometry. This geometric inheritance allows the user to create designs that are not possible through conventional software.

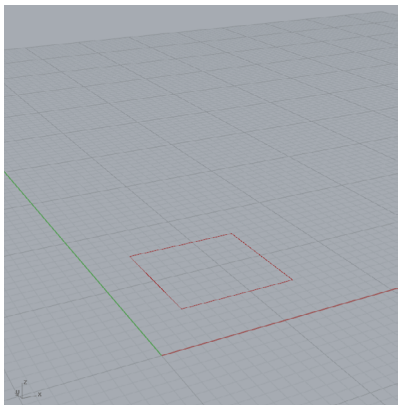
The components of Grasshopper are similar to rivalling 3D-software. Geometry is build up of points, vectors, lines, curves, surfaces, meshes and tools that alter these components. But the way geometry is generated in Grasshopper is entirely different to other CAD-software. Geometry in Grasshopper is created with predefined algorithms based on geometric inheritance. This means that every piece of geometry in Grasshopper can be linked together. The end result of the geometric model is thus chained to user specified input parameters such as point coordinates or the shape of a surface. Changes made to the input parameters will change the output model which allows for fast alterations in the design. It also allows the designer to create incredibly complex geometries with a dazzling amount of geometric components while keeping complete control of everything.

Grasshopper is extremely adaptable seeing how it is possible for users to personally program algorithmic components with scripting in known programming languages such as Visual Basic, Python and C#. The user made components can introduce new utilities not found

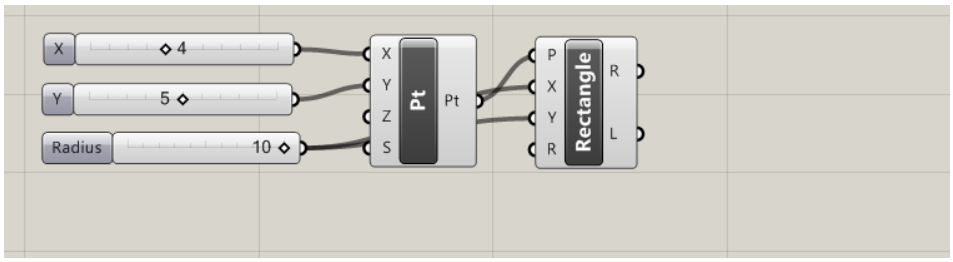
within the original Rhinoceros engine or the Grasshopper platform. This is the case with the developed dynamic relaxation component. The only limit to the use of Grasshopper is the imagination of the user.

The underlying 3D engine from Rhinoceros excels in dealing with curved geometry such as curves and double curved surfaces which is extremely important in relation to this thesis.
The capabilities above is why Grasshopper has been chosen as the underlying generative platform for this thesis. It is also currently free of charge.
A short example of generating geometry of a simple parametric building in Grasshopper will be explained, in order to quickly provide an example on the structure and functionality of Grasshopper. The pictures are taken from the graphical user interface of Grasshopper to the right and Rhinoceros to the left.

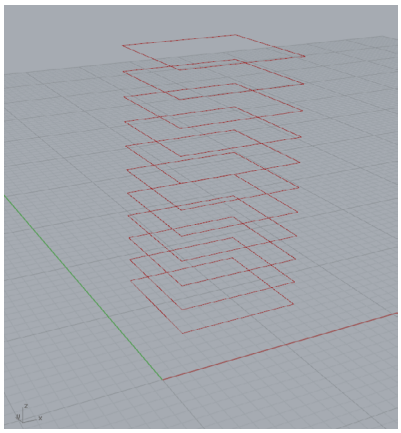
Figure 45: Shows screenshots from the GUI of Grasshopper and Rhinoceros as the parametric model is made through steps 1 to 3.



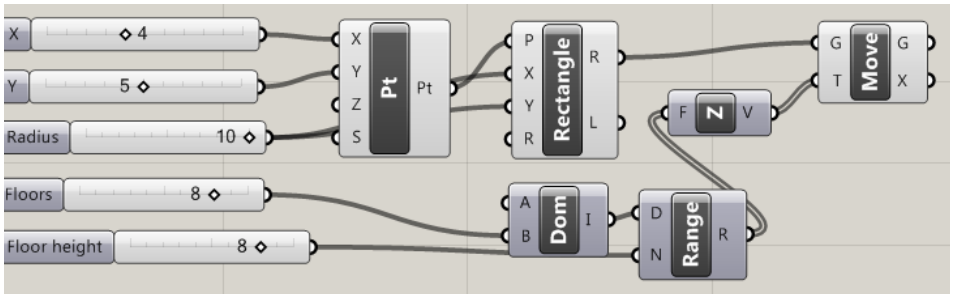
Step 1:



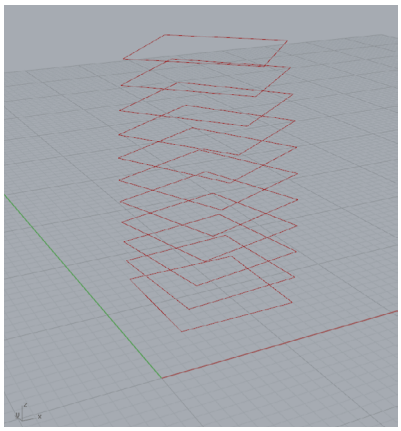
A simple point is created. The x- and y-coordinates can be altered using parametric numerical sliders. At the point a rectangle is drawn. The size can be altered using a second slider. This geometry can be considered the base floor plan of the parametric building.



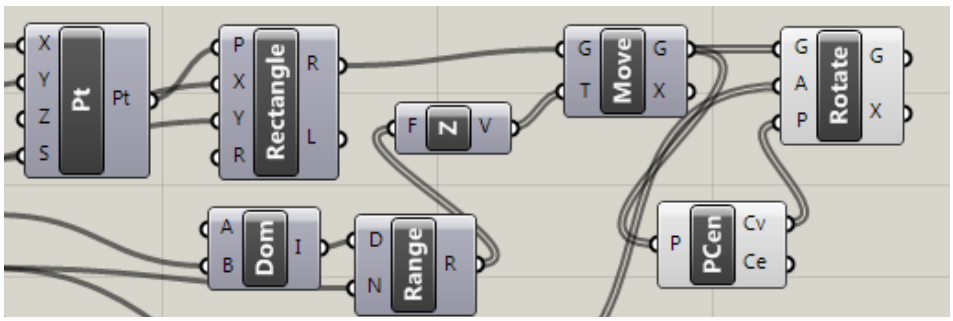
Step 2:



The rectangle is copied vertically upwards. The amount of times and distance for each copy is controlled using sliders allowing the user to create more floors and change the height of the floors. The sliders allow the user to adjust the design even after it is finished.



Step 3:



The floors are rotated with varying angles as the ascend upwards. This will give the structure a hyperbolic look. The finished building can be seen on the next page.

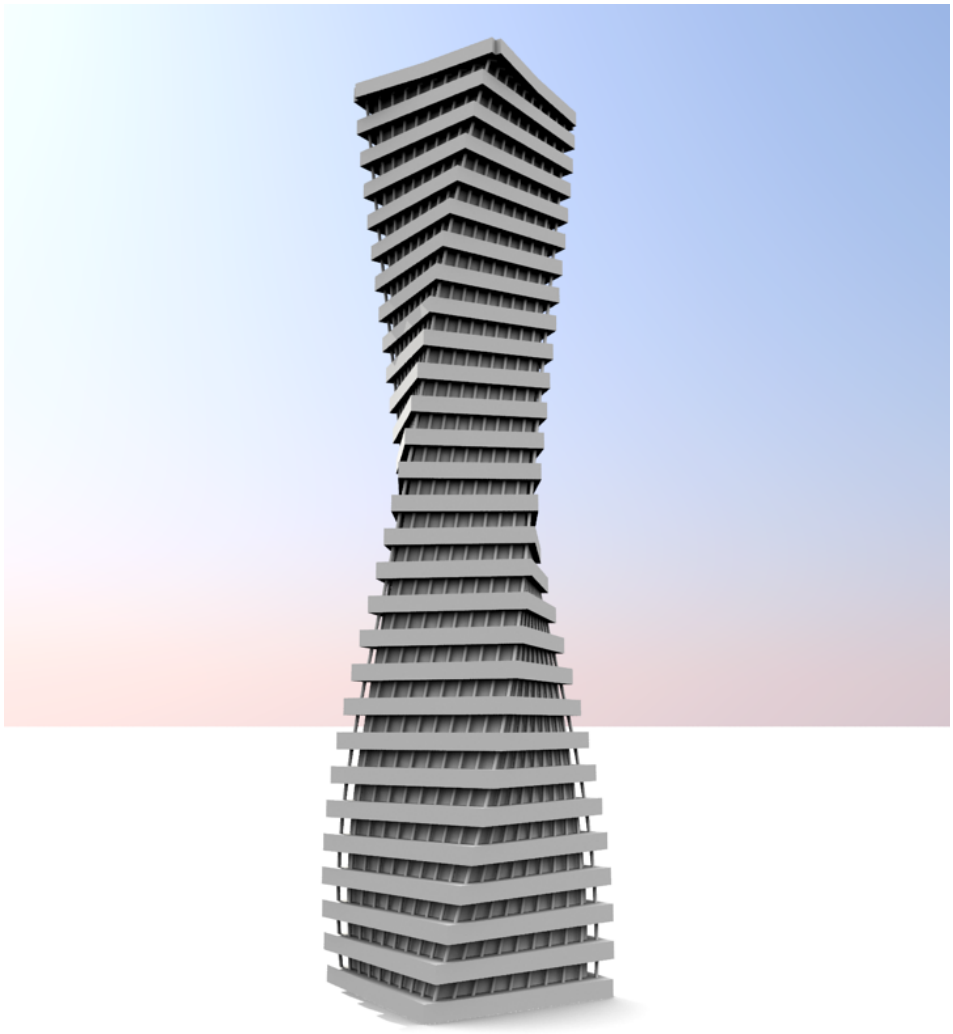


Figure 46: After a few more steps a parametric building is made. Everything seen on the figure can be altered using the input sliders, so the amount and size of the floors, amount of columns, rotation of the floors and so on can be altered in no time.

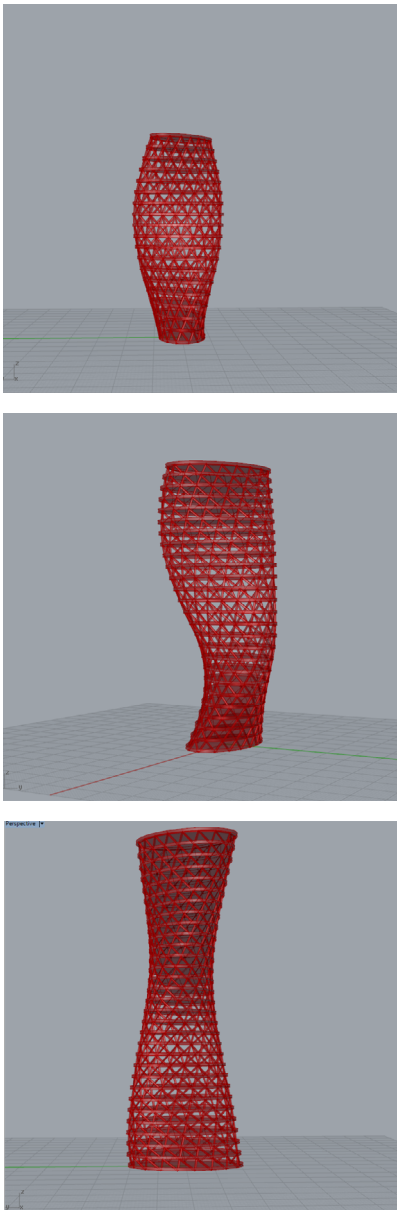


Figure 47: The towers showed in the pictures above are the results of of another parametric model made in Grasshopper. Input parameters are made to define the curvature of the tower, amount of floors, amount of structural members, etc. The model is build using traditional means such as ellipsoid, curves and surfaces. The parametric model allows the designer to quickly generate and visualize a large amount of models. Each model has a high informative value with precise details of each geometric elements such as member orientation and surface area. Once approaching detailing of the design alterations are not a problem as they often would be seeing how easily geometry can be altered and produced if a change in shape is desired.

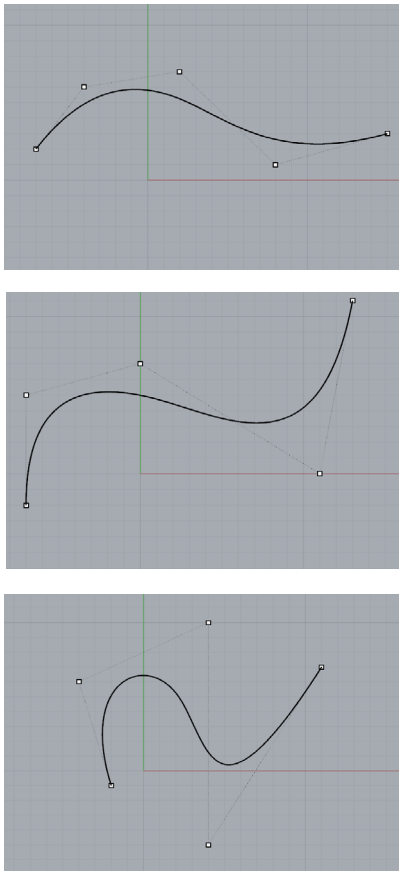


Figure 48: NURBS curved are created using control points. On the figure the control points are shown as small white squares.

4.2 Grid generation

4.2.1 The input geometry is a free form surface

When using parametric modelling, it is very important to keep in mind the goal of the design process. Tools such as grasshopper allow a new level of geometric complexity and variation, but it is crucial that the geometry created should best serve the functional requirements of the given project and not be created for the sake of complexity. Therefore when the designing team "Thinks" and "Do the rules" they must make sure, that the input parameters of the parametric model allows for total freedom in form so they are not restrained by the model.
As a consequence of this, the main input of the form finding tool is a free form surface modelled by the designing team. This is in contrast to physical form finding methods where the input is a flat grid that gets relaxed into its final shape as shown in chapter 3.4.1.
Allowing a free form surface to be the input will result in a large variety of shape deformations during the dynamic relaxation process. This shape deformation is largely dependent on how close the initial input was to equilibrium before the process. The designing team can also decide to stop the dynamic relaxation process at any time. This means, that a shape that is not fully converged towards equilibrium can be selected because of pleasing aesthetics or functional form over the fully relaxed form. The form will still be closer to equilibrium than the initial input form and require less material to build. These reasons make the tool very capable at conceptual design seeing how the optimization process will alter the input geometry. If the tool is utilized in detailing, then the models should be made parametrically as described in 4.1.

4.2.2 Parametrically represented curves and surfaces

The input surface should be generated parametrically in the Grasshopper environment so it can be altered during the design phase. In engineering and architectural field of computer-aided design (CAD) the industry standard representing free form surfaces is with non-uniform rational b-splines (NURBS). NURBS is a infinitely precise mathematical representation and generation of free form curves and surfaces that was originally developed to be used in the aeronautical and car industry.
The shape of NURBS curves and surfaces are dependant of input parameters in form of control points lying off the geometry itself. The huge advantage of NURBS surfaces is the ability to generate any free form geometry given the right amount and position of the control points. This allows the users to define and change the surface using these control points using the parametric Grasshopper environment.

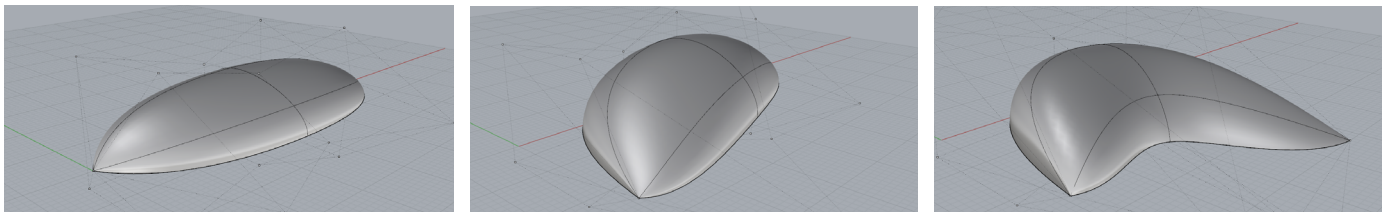


Figure 49: NURBS surfaces are created in the same way using control points. Once a surface has been created in the grasshopper environment it can be changed by altering these control points to get the exact shape that is wanted. The alteration can be made with parametric inputs or simply by moving the control points. This means that people with little CAD experience could jump right in an start shaping the input surface to their wishes.

The surfaces are generated using two parametric values u and v, so that any point on the surface can be represented with a combination of u and v which allows for a transformation from Cartesian coordinates to u- and v- values. In grasshopper there is a large amount of possibilities how to utilize this parametric NURBS representation of surfaces to create parametric algorithms that create surfaces that can be altered easily throughout the design process. This parametric representation of the free form allows the user to divide or tessellate the surface

into a discrete mesh representing a grid-net. The generation of grid-nets will be explained in the next passage.

4.2.3 Surface tessellation

The dynamic relaxation form finding component requires an geometric input in the form of a structural grid. This structural grid must be made of straight interconnecting lines with coordinate information of where the grid is supported. A number of methods of converting any given free form surface to a grid-net is presented below.

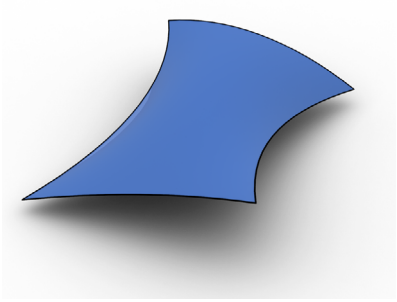


Figure 50: This arbitrary curved smooth surface will be used to present the different types of grid generation created in the form finding tool.

Grid-generation is the tessellation of a free form surface to a discrete mesh of connected lines. The mesh will represent the overall shape of the free form surface.
The surface tessellation, also known as panelling, is usually performed in two different ways with benefits and disadvantages. The easiest method is projection where a predefined planar grid is orthogonally projected onto a free form surface. This is a quick and easy solution to create an even grid if the surface does not possess a high degree of curvature. If the curvature of the surface is high the grid will become uneven and the appearance of the structure will suffer, see chapter 3.3. Joints and structural elements will also have a high degree of variance.
A better way create meshes is to use surface division. The method of surface division uses the internal parametric u and v representation of the free form surface to divide the surface into a discrete grid of points with Cartesian coordinates. The size of the divisions can be defined by the user to generate a grid with an appropriate size. This method of tessellation can be applied to a wide range of surfaces with different degrees of curvature.

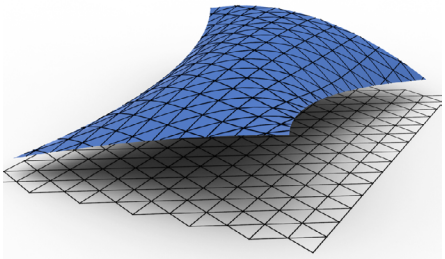


Figure 55: A grid created using surface projection method. The grid is somewhat uneven even though the surface is quite flat.

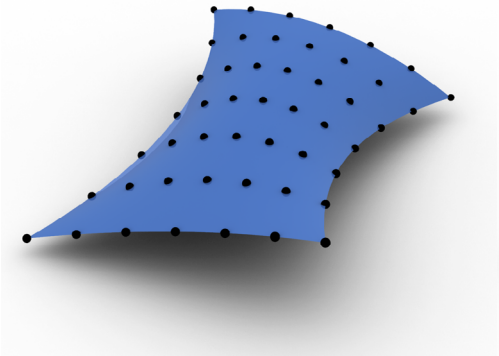


Figure 52: Division of surface into a rectangular point grid.

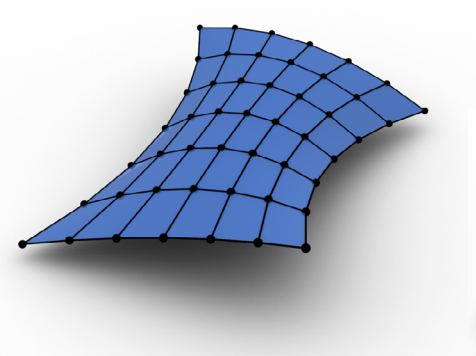


Figure 53: Connecting the grid points creates mesh lines that form the grid.

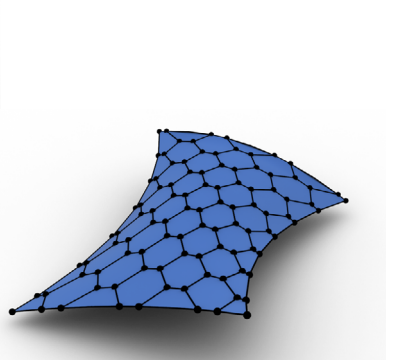


Figure 54: Hexagon mesh created using surface division.

The surface grid is usually rectangular with evenly distributed points. The grid is then populated with connecting lines to create a polygon mesh.
Meshes are geometrical structures used in computer graphics to generate visuals of complex, and often curved, geometries. Meshes are build using a combination of three components. These components can be translated into structural elements.

- *Mesh Vertices*: The grid points, or nodal coordinates, where the mesh lines intersect with each other. These vertices represent structural joints in the gridshell.
- *Mesh Edges*: The lines that show the structure of the mesh. The mesh edges will later be modelled as the structural members of the gridshell.
- *Mesh Faces*: Cells, or panels, between mesh edges. Every mesh face has a unique orientation and a normal vector specifying its exact orientation in space. The mesh faces can be thought as the panels between the structural members.

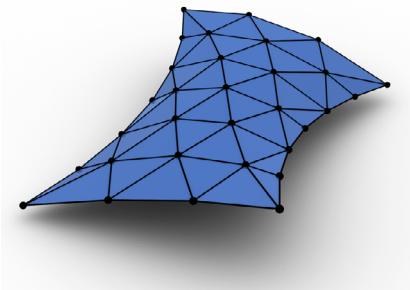


Figure 51: A triangular mesh created using the surface division method.

The Gaussian curvature of the smooth input surface is transformed into angle defects in

the polyhedron made by the generated mesh, meaning that smooth surface curvature are now concentrated at the mesh vertices. When the smooth surface is divided into smaller and smaller segments the smoothness of the mesh increases.
The appearance of the mesh can be changed by the way points are connected to create triangular, quadrilateral, diamond or hexagonal shapes.

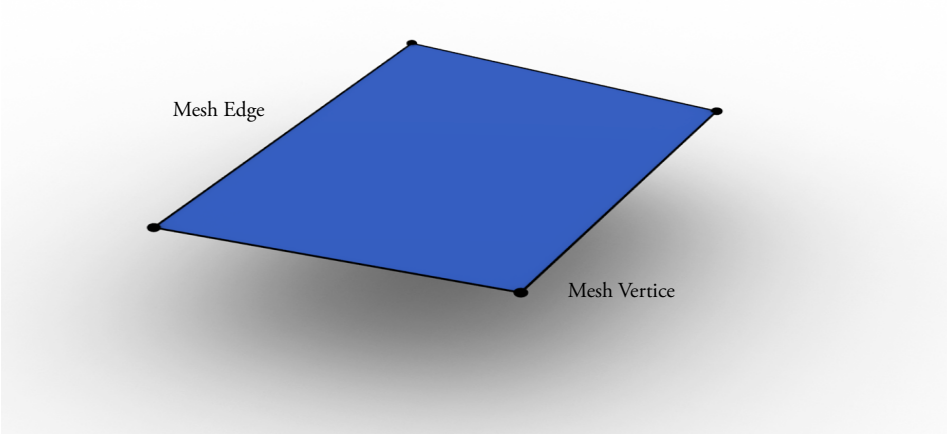


Figure 56: Mesh edges and vertices are the two components that will be used to generate the grid-shell structure. It is also the input to the dynamic relaxation script.

4.2.4 Equilateral triangles using physic based repulsion and attraction

A method of improving upon the surface division principle can be applied to create something very close to an equilateral grid using triangles. An equilateral grid is a mesh where all the mesh edges are of the same length. It is mathematically impossible to transform a smooth double curved surface into equilateral triangles that are exactly the same size without making compromises. But it is possible to get quite close. If the length of the mesh edges are allowed to vary a little or the mesh is allowed to extend the limits of the surface borders it is possible to make an equilateral grid.
An algorithm was developed in Grasshopper that populates the surface with a grid of points. Every point is connected to each other with a virtual spring that allows for repulsion and attraction between points much like the principle in dynamic relaxation. These points are restrained to the surface and relaxed into an evenly distributed arrangement on the surface by letting the interconnecting spring forces move the grid points.. The resulting grid will be composed of a mesh with small variance in triangle size which should reduce production cost of members, joints and panels.

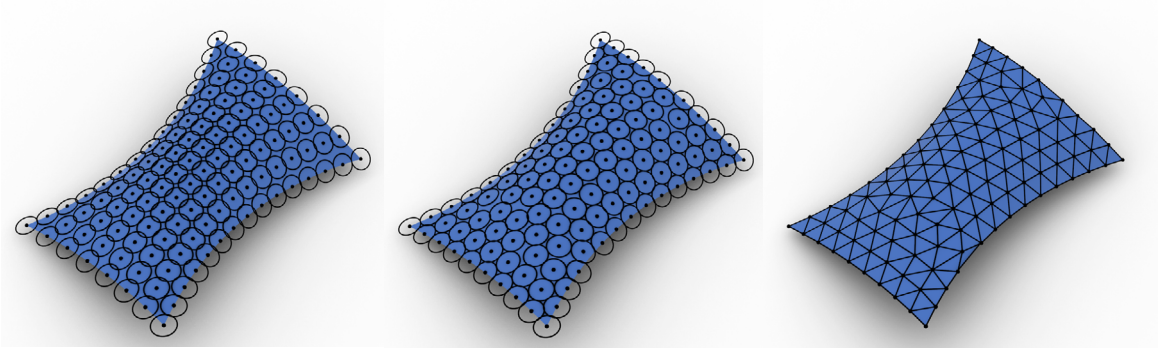


Figure 57: The process of creating the equilateral triangle grid using forces of repulsion and attraction.

The meshes generated have several roles in the form finding tool. The mesh edges will visually represent the structural elements of the gridshell and thus visualize the entire gridshell structure.

- The lines in the mesh are used as input for the form finding tool. The tool returns a relaxed mesh of lines.
- The edges and vertices can be imported into FEM software as structural members and joints to perform structural analysis of the gridshell structure.
- The mesh components can be used to generate plans for fabrication and construction of joints, supports, structural members and panels.

4.2.5 Types of grid generating algorithms

A number of different grid generating algorithms were composed in grasshopper to allow the designing team to quickly explore different tessellation possibilities. Each grid type has its own distinct visual style and will result in a unique gridshell. A number of examples are shown below to represent these algorithms in action. The input is an arbitrary free form blob-like surface to show how versatile the algorithms are.

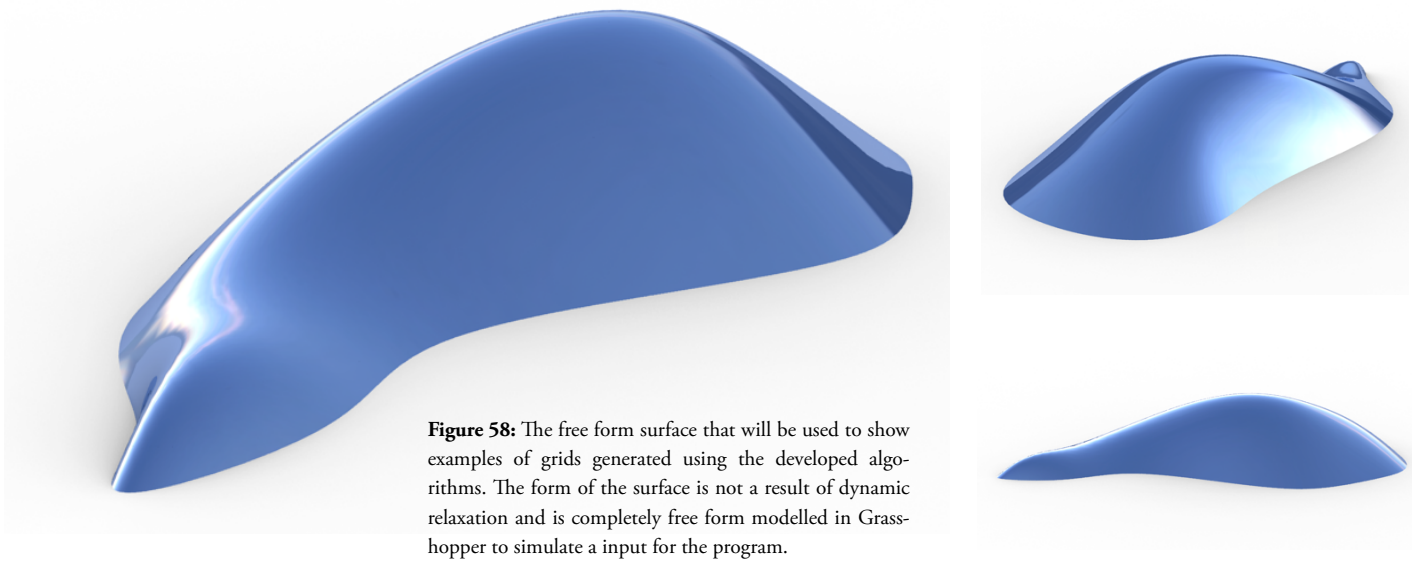


Figure 58: The free form surface that will be used to show examples of grids generated using the developed algorithms. The form of the surface is not a result of dynamic relaxation and is completely free form modelled in Grasshopper to simulate a input for the program.

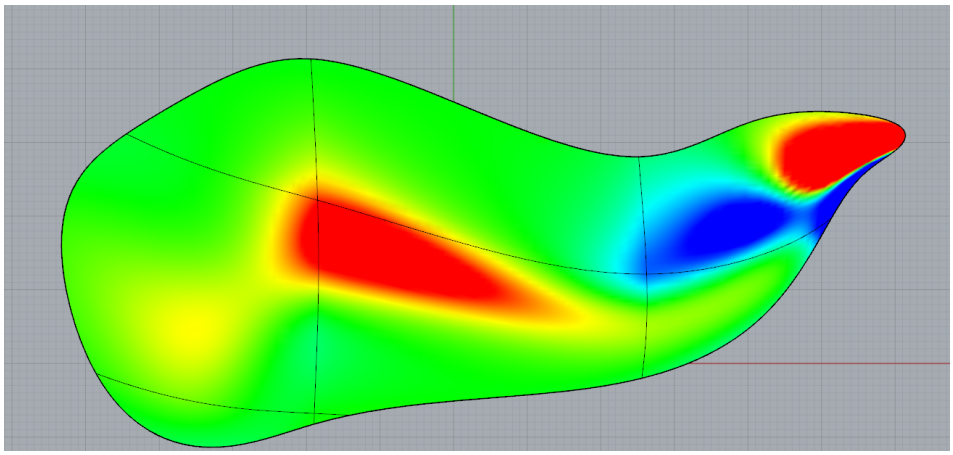


Figure 59: Surface curvature can be shown in Rhinoceros. It shows where grid complications might arise. Flat surfaces are easier to tessellate. Areas with a local maximums or minimums in Gaussian curvature result in kinks in the grid if member length is not sufficiently small. Visualization of curvature can be used to quickly evaluate the surface. Criteria can be set so that only a certain degree of curvature is acceptable due the restrains of structural nodes.

Triangular tessellation

Triangular meshes have many advantages to gridshell structures. They are geometrically locked which makes for very rigid structures. Panels covering triangles are always planar. The joints of triangular grids are not as simple as the following grid types. The triangular grid will always have six elements joining at the mesh vertices making complex connections. See chapter 4.6 for further discussion of connections and fabrication.

The grid is generated by evenly distributing points on the free form surface and connecting the points with lines forming a triangular grid.

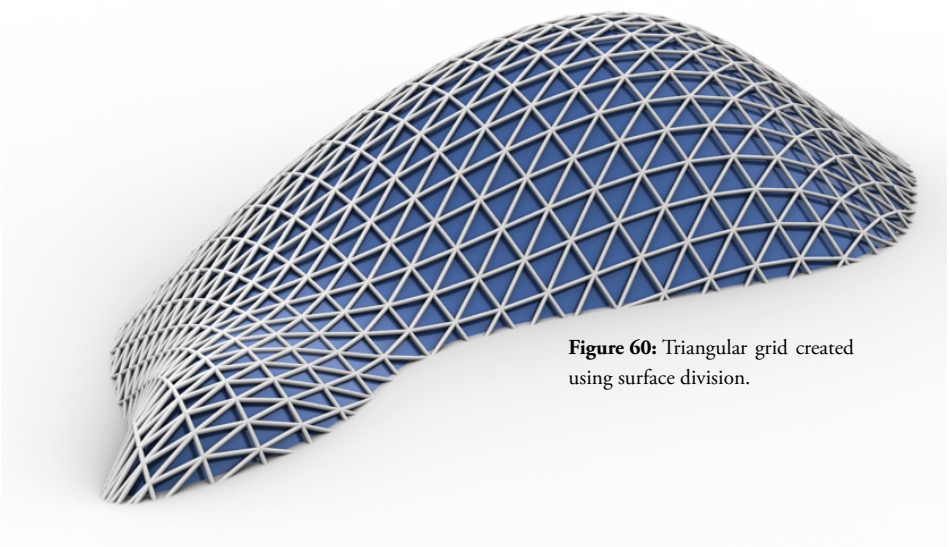


Figure 60: Triangular grid created using surface division.

Quadrilateral tessellation

Quadrilateral meshes are common in gridshell structures. They are simpler to produce and construct compared to the triangular meshes. This is due to the fact that every structural joint connects only four members compared to six of the triangular mesh. This simplicity is at the expense of stability. The quad cell is not geometrical stable and can easily buckle if diagonal bracing is not existent. Bracing is usually implemented with cables in selected cells.

The grid is formed in a similar fashion to the triangular grid but without the diagonal bracing.

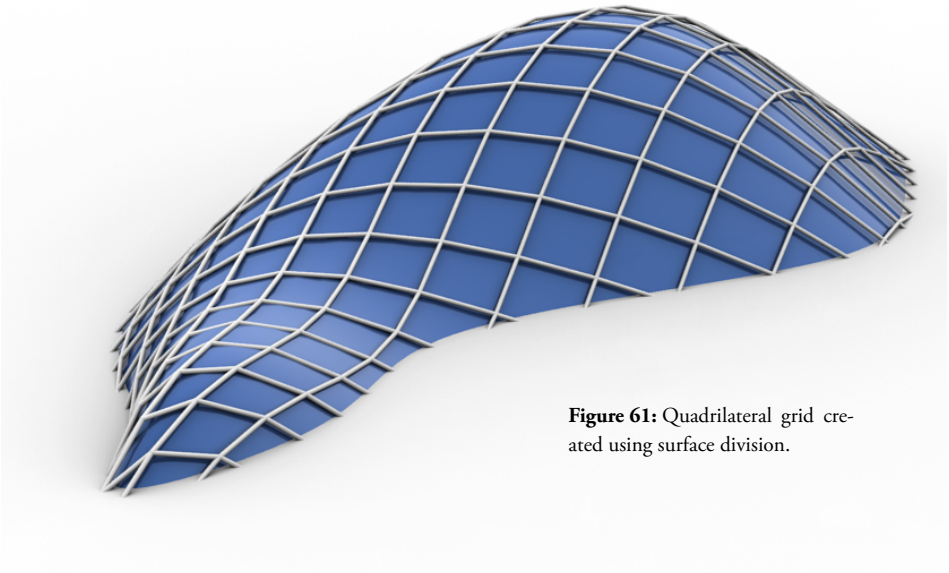


Figure 61: Quadrilateral grid created using surface division.

The structural cells can be shifted to represent a parallelogram by scrambling the vertices of the mesh.

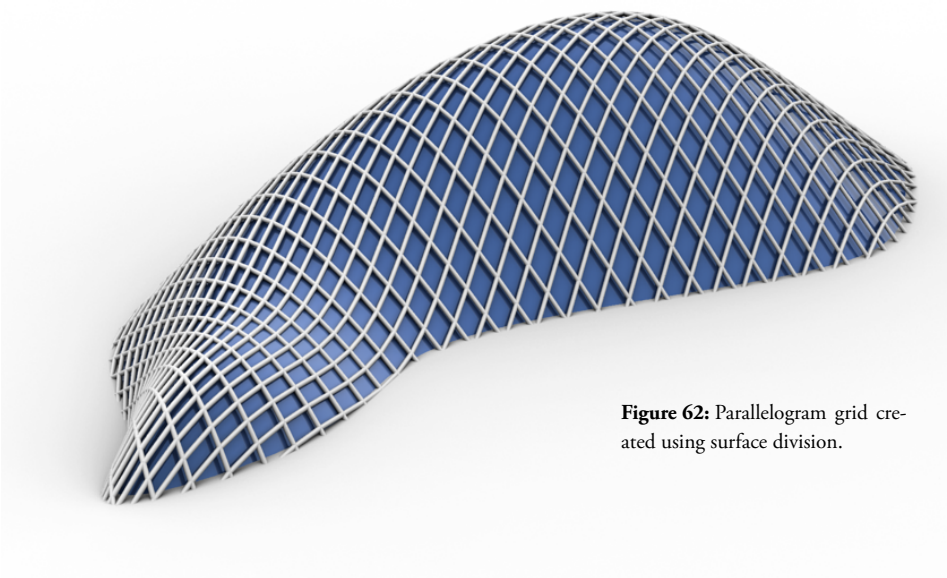


Figure 62: Parallelogram grid created using surface division.

Hexagonal tessellation

Hexagonal be implemented to achieve a honeycomb-like structure . The advantages of this structure is that the structural joints only connects three structural elements making for simpler connections.

The grid is generated similar to the triangular grid. Hexagons are created without connecting all the lines to form triangles.

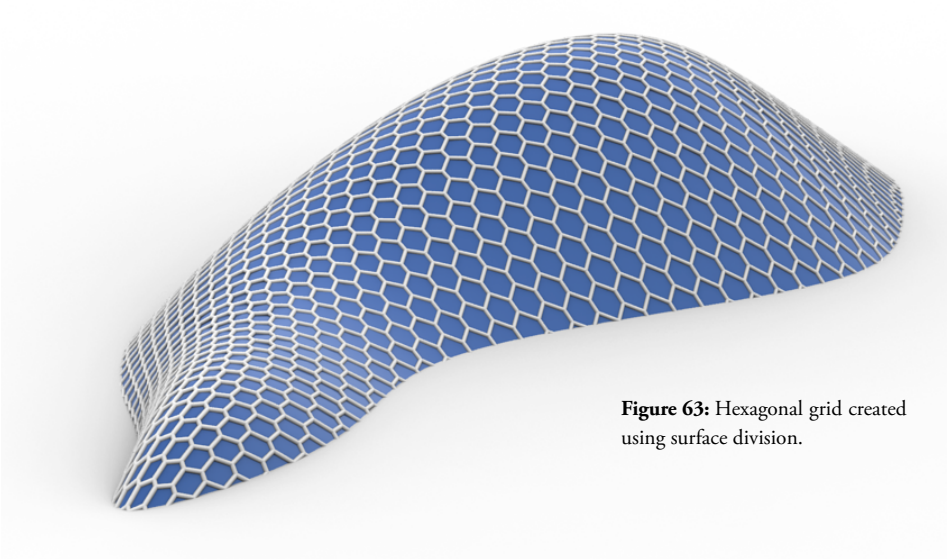


Figure 63: Hexagonal grid created using surface division.

Voronoi tessellation

The Voronoi grid is formed by randomly populating the free form surface with points. The space between the points is divided into regions that form the cells of the grid. The random nature of the voronoi distribution gives the resulting grid a natural and organic appearance.

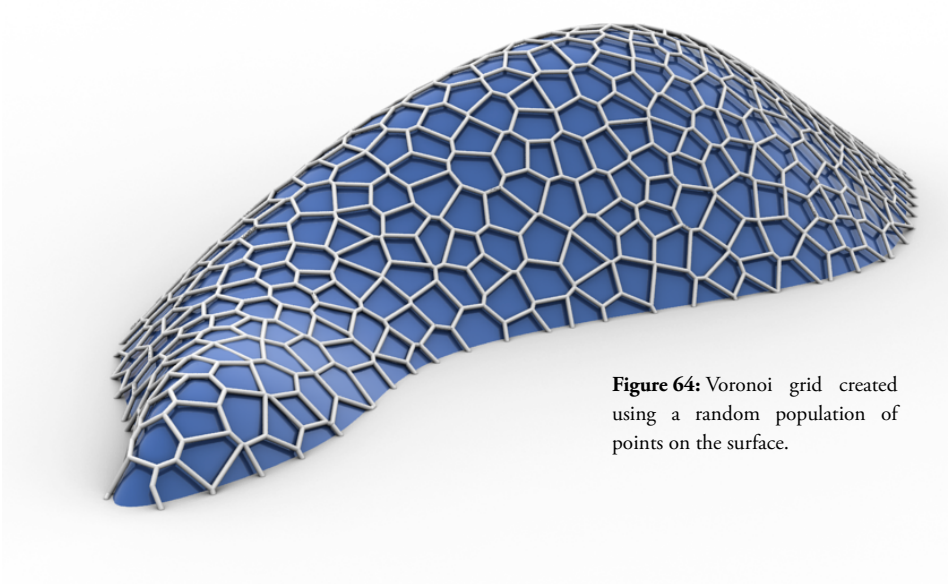


Figure 64: Voronoi grid created using a random population of points on the surface.

The equilateral triangular grid

The equilateral triangular grid creates a better solution to how the elements meet the ground.

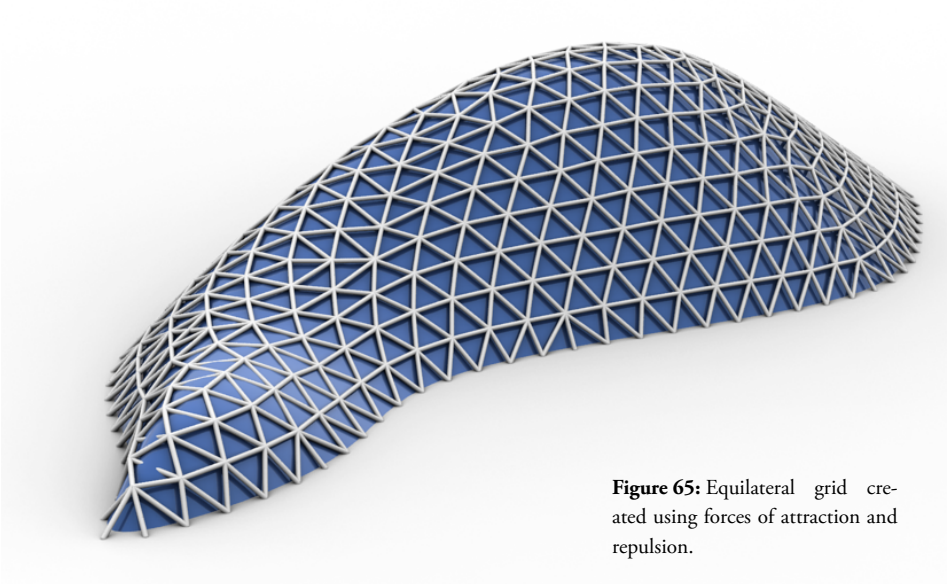


Figure 65: Equilateral grid created using forces of attraction and repulsion.

4.2.6 Panels

The mesh lines have been deliberately chosen to be straight and not be free form curves following the curvature of the form. This is to ensure that panels that cover the grid are as planar as possible to reduce construction costs seeing how planar panels like glass panes are cheaper to produce than panels with single or double curvature. The panels must be triangular to ensure planarity. The panels will not be planar if the mesh is quadrilateral and the overall shape is very curved making the panels hyperbolic and difficult and expensive to manufacture. This problem can be solved by ensuring that two edges of each quadriteral mesh cell are parallel. The overall shape of the gridshell will appear curved with planar panels if the mesh division is sufficiently small. This also reduces the warping angle of member ends described in 4.7.

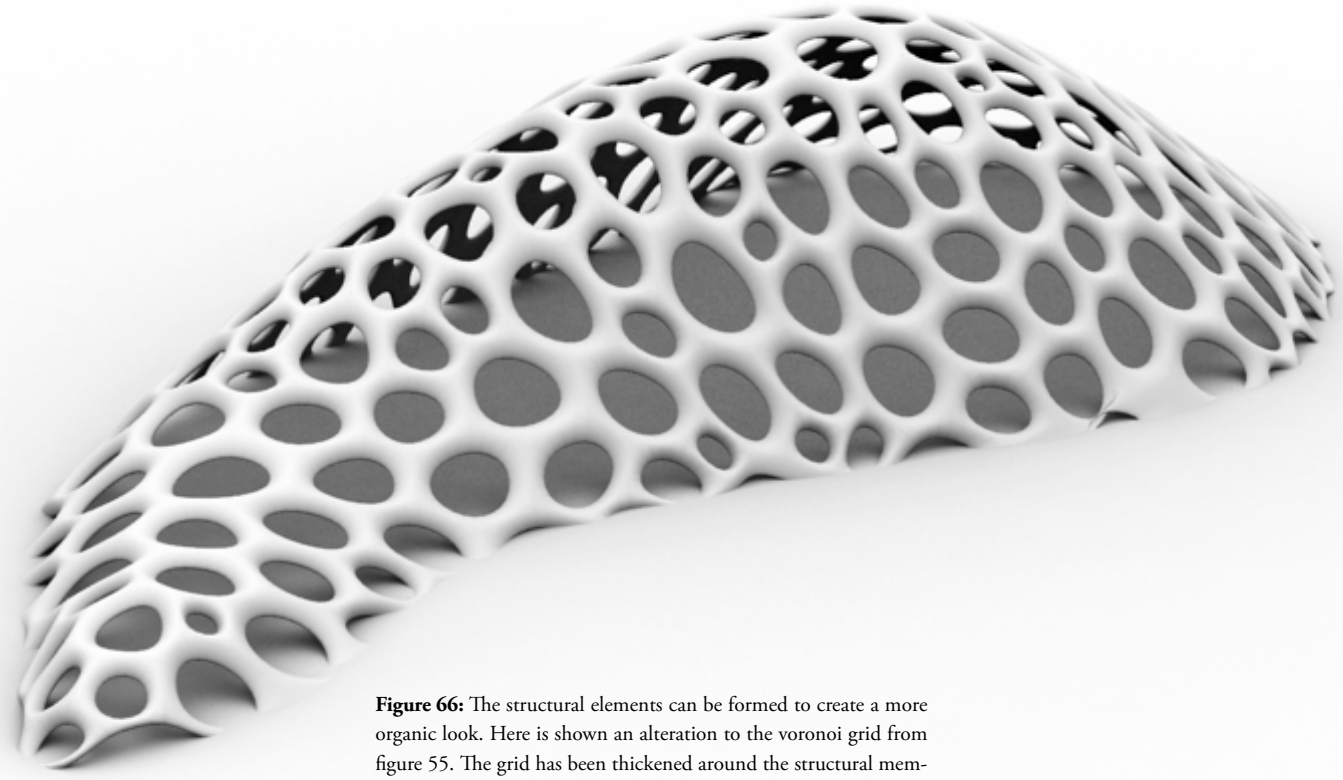


Figure 66: The structural elements can be formed to create a more organic look. Here is shown an alteration to the voronoi grid from figure 55. The grid has been thickened around the structural members to create the effect.

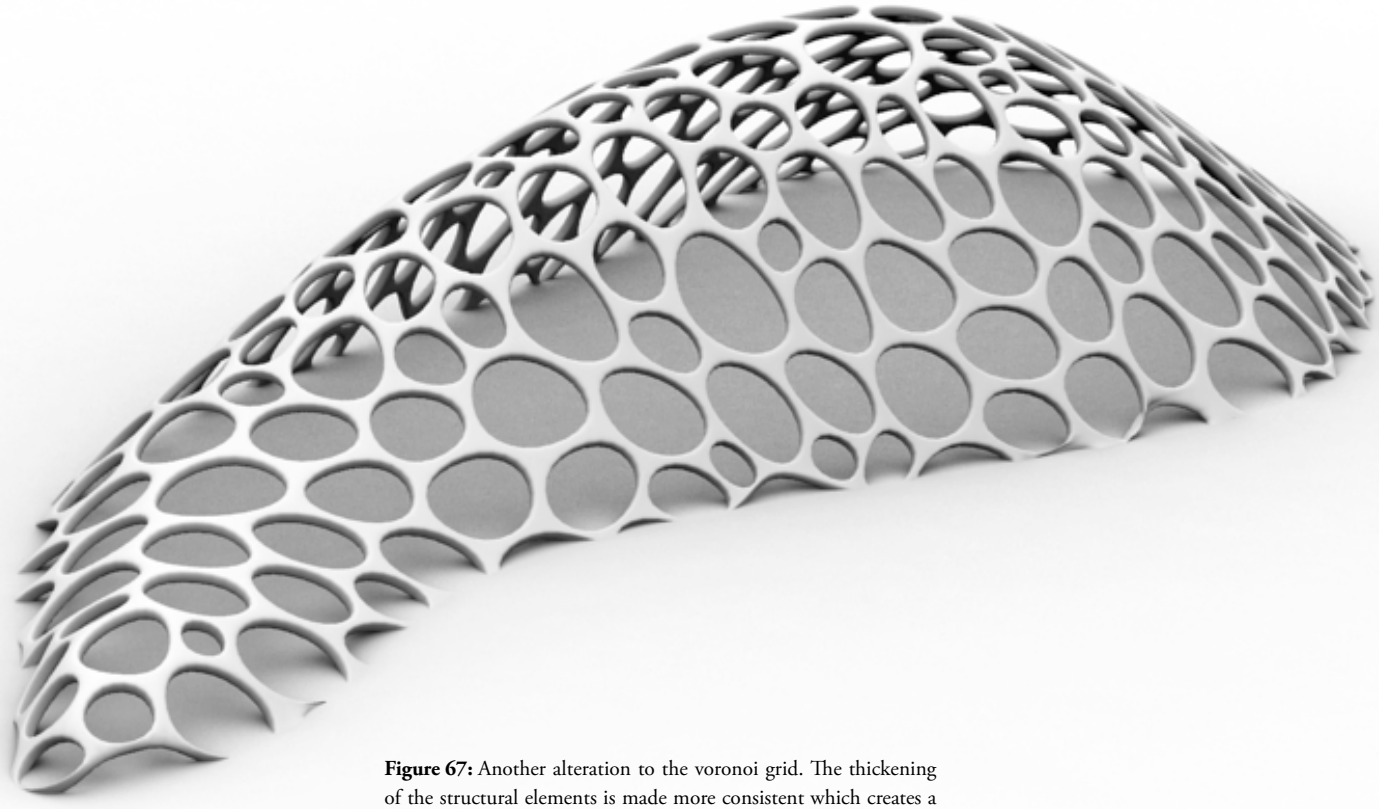


Figure 67: Another alteration to the voronoi grid. The thickening of the structural elements is made more consistent which creates a different visual effect.

4.2.7 Analog to digital

An important issue arises when describing the computational generation of free form surfaces. Many architects and engineers prefer to work outside of the computer and use hand drawn sketches and physical models to do their conceptual work. Physical modelling should still be a valid possibility to create the initial input shape. Cutting edge technologies like 3D laser scanning should help bridge the gap between the analog and digital world by allowing the designing team to create conceptual physical models that can then be scanned and imported to the modelling software for further form finding and structural verification.

Figure 68: A small physical model made in paper clay.

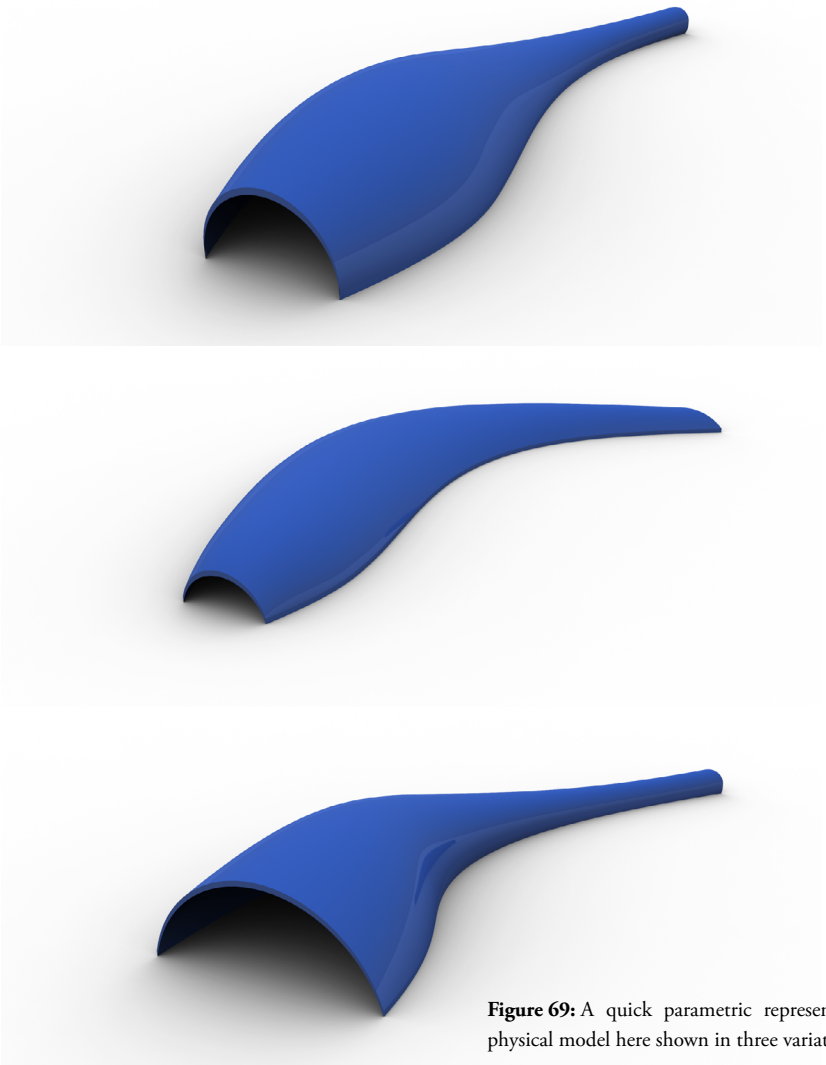


Figure 69: A quick parametric representation of the physical model here shown in three variations.

4.3 The Dynamic Relaxation Component

> "I think everybody in this country should learn how to program a computer — because it teaches you how to think." - Steve Jobs

The grids generated by the algorithms described in the previous segment will be optimized using a developed dynamic relaxation component written in the C# programming language. This chapter will describe the structure of the developed component.
The overall structure of the component is shown in the diagram in figure 70 below. The complete breakdown of the structure of the dynamic relaxation script is described in chapter 4.3.2 on page 50.

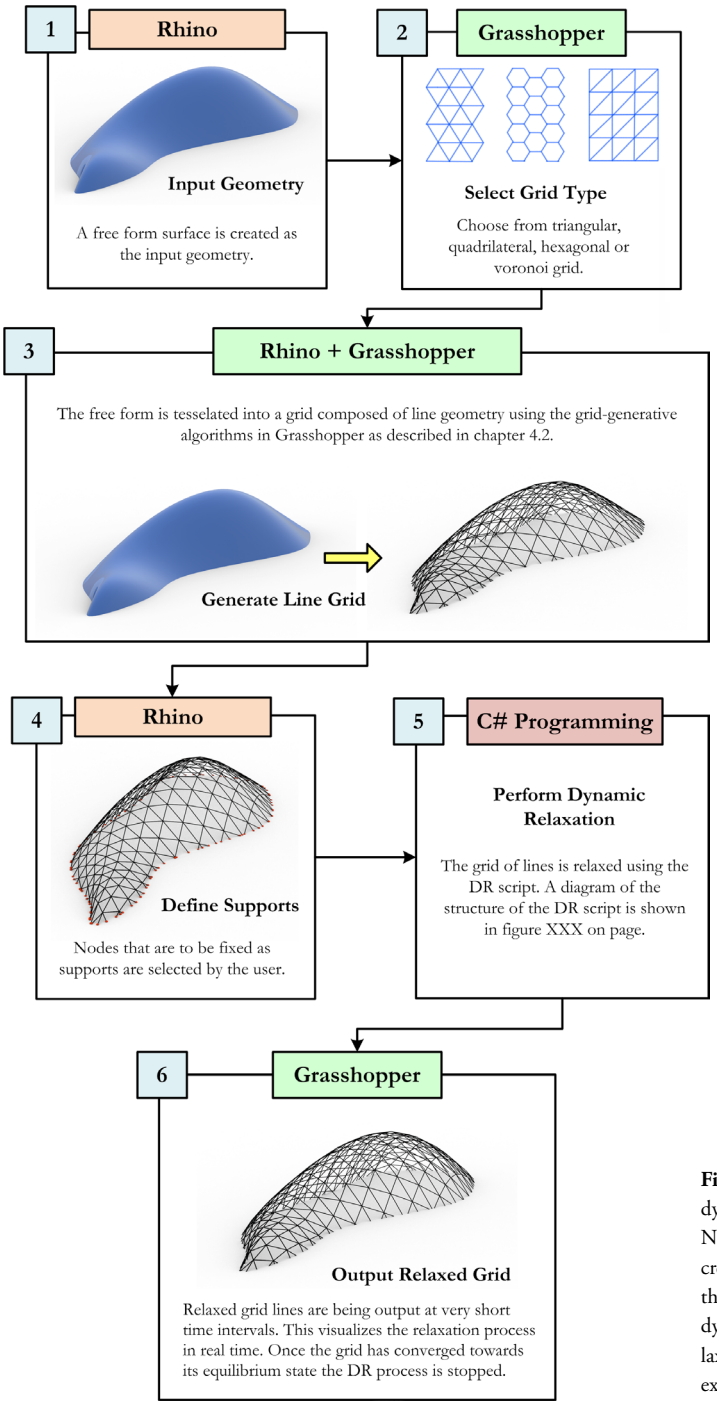


Figure 70: The diagram shows the general structure of the dynamic relaxation component. The user designs an input NURBS surface and uses the grid generating algorithms to create a grid. The grid along with support points is fed to the dynamic relaxation script. The script performs real-time dynamic relaxation. Once the relaxation is complete the relaxed grid can be evaluated using Karamba FEM-software as explained in chapter 4.4.

```
public void SetForces(List < Vector3d > _positions){

  foreach (Node node in nodes){
    Vector3d nodeForce = new Vector3d(0, 0, 0);
    foreach (Spring spring in node.springs){
      if (spring.start == node){
        nodeForce -= spring.Force(_positions);

      } else {
        nodeForce += spring.Force(_positions);
      }
    }
    resForces[node.index] = nodeForce;
  }
}
```

Figure 71: A small piece of the C# syntax. The syntax is the working environment where the code is written.

4.3.1 Programming Components for Grasshopper

Programming is about unlocking the vast computational potential within the computer. Programminletsg the computer do all the heavy repetitive lifting to achieve solutions that would otherwise be too complex or too time consuming. Learning programming is essentially learning how to break down problems into oversimplified instructions the computer can process to provide solutions. This is done by writing algorithms that are basically sequential lists of instructions that solve a particular problem. The computer uses a one-step-at-a-time approach so the programmer needs to break down the problem into discrete pieces.

The dynamic relaxation component was coded from scratch in the C# programming language using the theory provided in chapter 3.5 with a change in numerical integration explained in chapter 4.3.2. The completed component can be distributed to any given computer with Grasshopper to perform the task of dynamic relaxation providing it is given the correct input.

Creating any new components in Grasshopper does require programming knowledge in one of the currently supported languages such as C#, Python and Visual Basic. The level of programming knowledge is tied to the level of complexity of the component. For a dynamic relaxation algorithm a couple of advanced methods has to be applied in the code such as object oriented programming and implementation of iterative numerical integration schemes to solve the equations of the particle-spring-system. This will be discussed in further detail in chapter 4.3.2.

The programming language chosen for this thesis, C# (pronounced c-sharp), was chosen because it is one of the three languages available in Grasshopper and its syntax structure resembles that of the user friendly programming language named Processsing which the author familiarised with during the early phases of writing this thesis.

The algorithm was written from scratch to gain a full understanding of the computational process of the form finding tool. This was also to ensure complete compatibility with the other features in the tool such as grid generation algorithms and dynamic finite element analysis. Programming the algorithms needed gives the user a great flexibility to create whatever tool is needed. It also gives the user the possibility to adapt the tool or improve it by introducing new features. Further improvement to the tool are discussed in chapter 4.9.

As mentioned in chapter 4.1, using programming to solve engineering problems provides an automated process without the need for manual intervention. When dealing with highly complex wide-span structures as gridshells any non-automated processes are unacceptable due to economical and time frame reasons [Barnes & Dickson, 2000]

There are a few commercially available software solutions available on the market that include a build-in dynamic relaxation form finding algorithm such as Oasys GSA. These expensive off-the-shelf software packages does not live up to the requirements stated in chapter 3 and do not allow for parametric cooperation with the Grasshopper GUI.

The fact is that to perform dynamic relaxation expensive software is not needed, seeing how algorithmic components can be tailor made within the Grasshopper environment to fit the needs of the user.

4.3.2 Detailed dynamic relaxation script structure and flow

Implementing the dynamic relaxation theory into a programming environment requires a discrete numerical solution of the equations provided in chapter 3.5.3. This approach will be explained in the following passage.

The computer has an incremental approach to everything it does. In order to model physical behaviour the equations of motion for the particle system must be written on vector form and solved with an iterative numerical integration scheme to match this one-step-at-a-time approach.

If a single particle is considered, its position in Cartesian space can be described by an Euclid-

ean position vector. In order to simulate movement of particle due to applied loads, a change of the position vector must be made. A rate of change of position is defined as velocity and a rate of change of velocity is defined as acceleration. As explained in chapter 3.5 the acceleration vector can be found by analysing the resulting vector forces acting on the particle and dividing with the particle mass. In order to calculate the movement of the particle a numerical integration including additions of acceleration- and velocity vector is required for a given time step.

The computational procedure of dynamic relaxation described in chapter 3.5.3 requires a spring-particle-system to simulate the physical behaviour of the hanging chain model. This particle system will be a collection of programming objects (see end of this passage) in the form of Nodes and Springs. The physical behaviour of the system will then be modelled using Vector positions and a numerical integration scheme as mentioned above.

The numerical integration returns an updated Vector position for a given time-step under the influence of an exterior load. In order to avoid having to find the inverted geometry after the relaxation process the applied load component is added as acting upwards instead of downwards. The simplest explicit numerical integration scheme is forward Euler integration where the central difference method is used to calculate the updated vector position and velocities. Explicit integration methods will calculate the updated vector position based on the current vector position. This explicit method is unstable when the time step used for integration is too large. If the time step is too large the integration can overshoot the equilibrium state and result in an accumulation of error so that the model is in an irreversible instable state [Bak, 2011].

A small change to the theory described in 3.5.3 will be made to achieve a more stable solution. Another numerical integration scheme than the one proposed in chapter 3.5.3 by [Wakefield, 1999] will be implemented in the dynamic relaxation code to provide a more stable solution. This scheme is called Velocity Verlet integration [Swope, 1982]. In Velocity Verlet the velocity and position vectors are calculated using the same time variable resulting in the following equation for the updated nodal geometry

$$\vec{x}(t+\Delta t) = \vec{x}(t) + \vec{v}(t)\Delta t + \frac{1}{2}\vec{a}(t)\Delta t^2 \tag{13}$$

This integration scheme still requires a sufficiently small time-step in order to achieve stability. Convergence can be tracked by plotting the kinetic energy of the system as shown in figure 72.

Code build around object oriented programming (OOP) can be seen as a collection of interacting objects that are sending, receiving and processing data amongst each other. The three objects scripted in this thesis are Springs, Nodes and a SpringSystem. Objects are instances of programming classes that can be seen as a collection of variables contained within the object and the operations the object can perform. In programming the properties of an object is variables and the things it do is called functions or methods. Numerous methods was programmed in order to make up the dynamic relaxation component. The code in its entirety can be seen in Appendix A1 along with specific comments to each line of code.

The structure and flow of the program is presented in a diagram on figure 73 on the next two pages. This diagram illustrates the different functions performed within the dynamic relaxation script and how they are chained together. In the diagram there are references to Appendix A1 and the lines of code that perform the actions in the script.

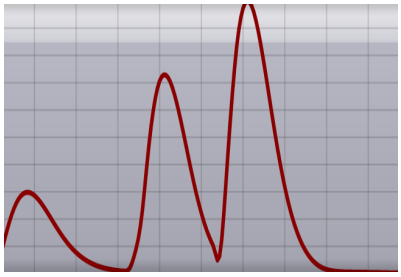
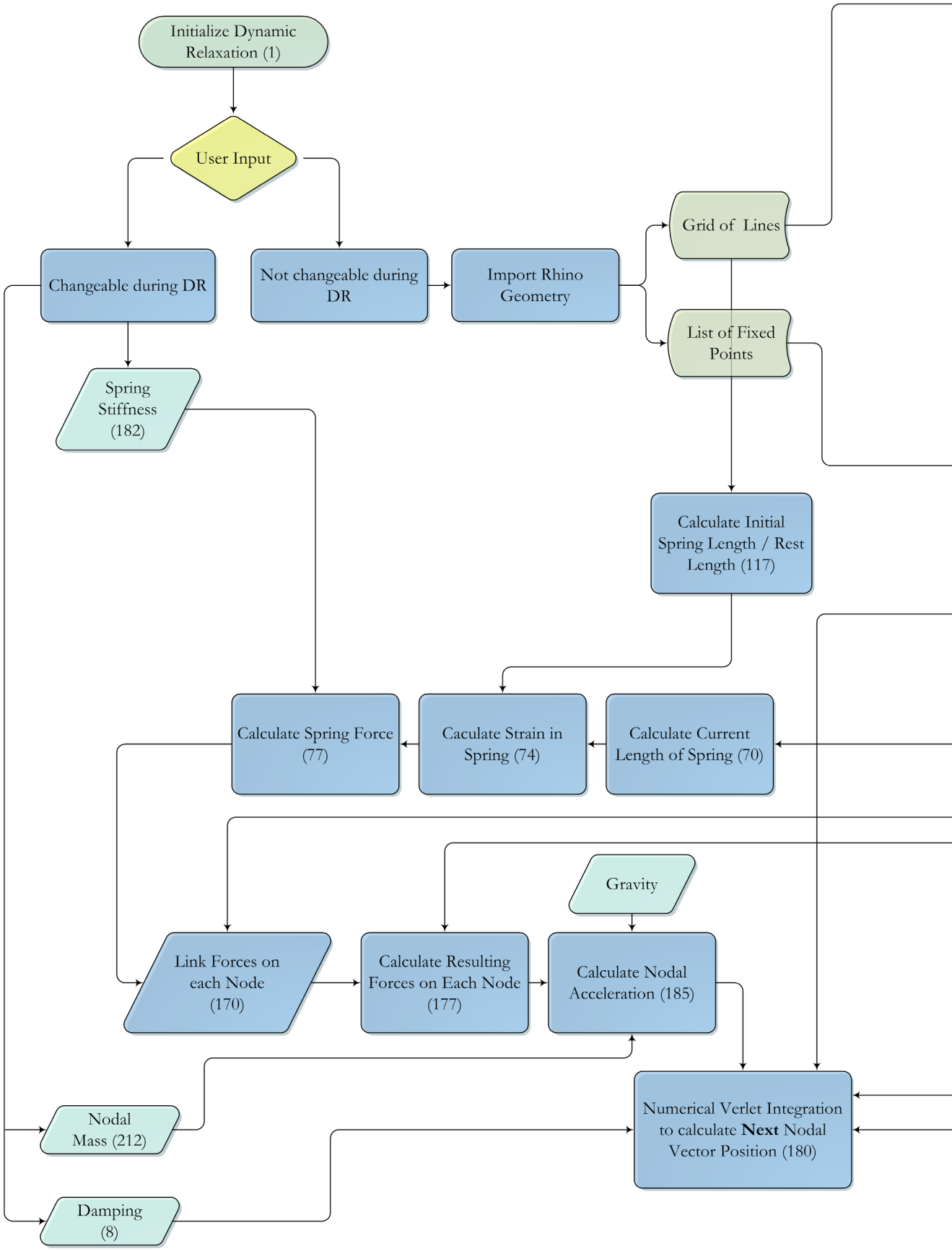


Figure 72: Convergence of the dynamic relaxation can be tracked by plotting the kinetic energy of the process. Once the energy approaches zero the grid will be in an equilibrium state. This figure shows a plot of kinetic energy for a relaxation process. The three spikes in energy are due to the user changing stiffness of the springs to achieve different results.


Initialize Dynamic Relaxation (1)
User Input
Changeable during DR
Not changeable during DR
Import Rhino Geometry
Grid of Lines
List of Fixed Points
Spring Stiffness (182)
Calculate Initial Spring Length / Rest Length (117)
Calculate Spring Force (77)
Caculate Strain in Spring (74)
Calculate Current Length of Spring (70)
Gravity
Link Forces on each Node (170)
Calculate Resulting Forces on Each Node (177)
Calculate Nodal Acceleration (185)
Nodal Mass (212)
Numerical Verlet Integration to calculate **Next** Nodal Vector Position (180)
Damping (8)

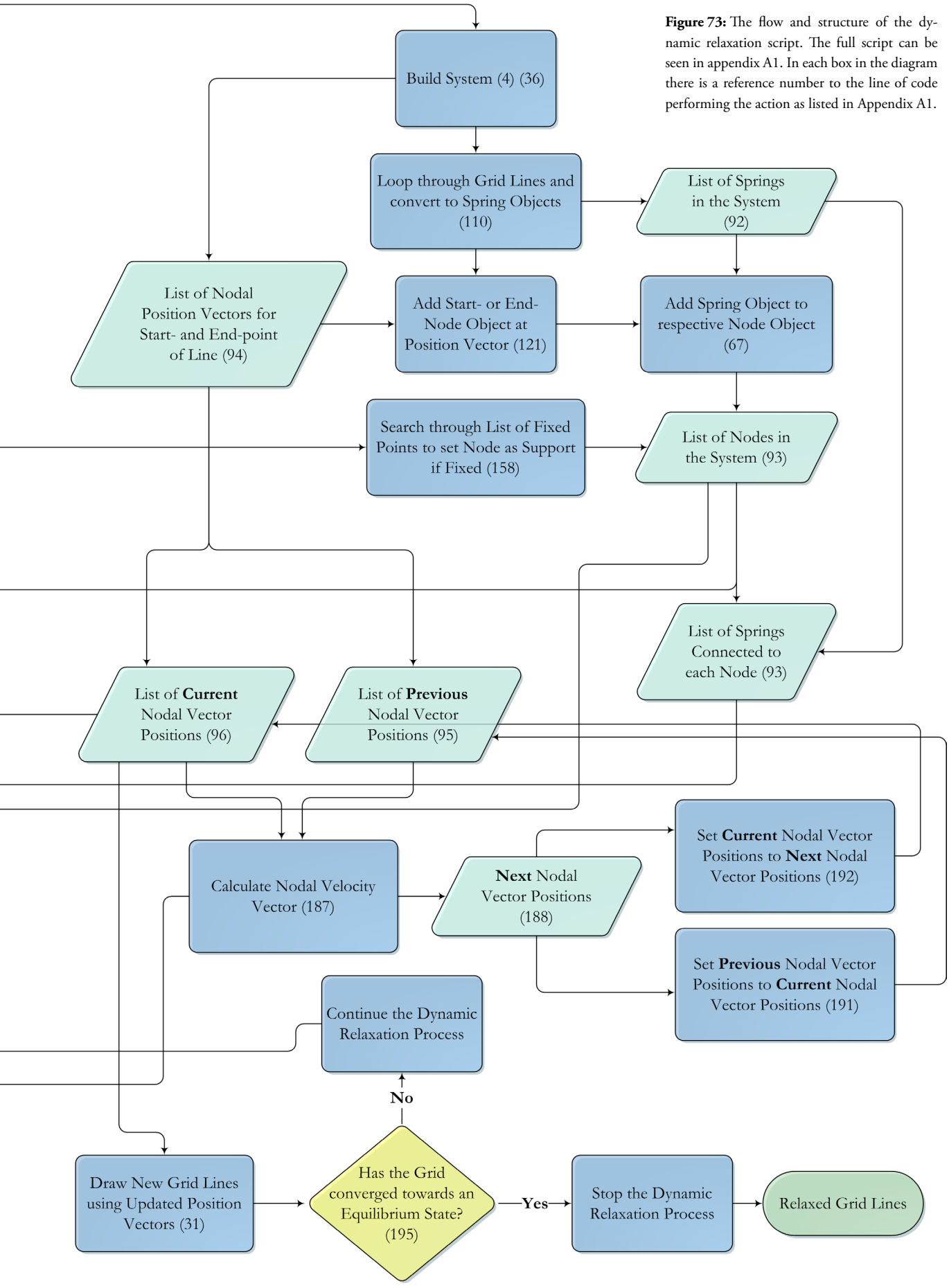


Figure 73: The flow and structure of the dynamic relaxation script. The full script can be seen in appendix A1. In each box in the diagram there is a reference number to the line of code performing the action as listed in Appendix A1.

Figure 75: The roof for the Sicli company building in Geneva designed by Heinz Isler. The form is imitated using the dynamic relaxation component. Image source: http://www.ce.jhu.edu/

4.3.3 The dynamic relaxation component in action

The following pages will showcase two simple examples created with developed dynamic relaxation component. Full geometric and structural testing of the component to verify its efficiency will be done in chapter 4.6 with much more elaborate examples. The tool will also be tested in the solved case starting in chapter 5.
A complete rundown of the entire developed graphical user interface (GUI) in Grasshopper will be presented in chapter 4.5. Chapter 4.5 will also go through the the workflow of the transformation from a free form input to a finished structurally verified gridshell.

The first simple example will try to recreate the form shown in figure 75. The shape is found using a simple planar triangular grid supported only at the outermost four points.
The grid and the support points are imported into the dynamic relaxation component.
The gridshell created in figure 74 will be used in chapter 4.4 to demonstrate the FEM software included in the tool. Grid elements are still straight lines after the relaxation.

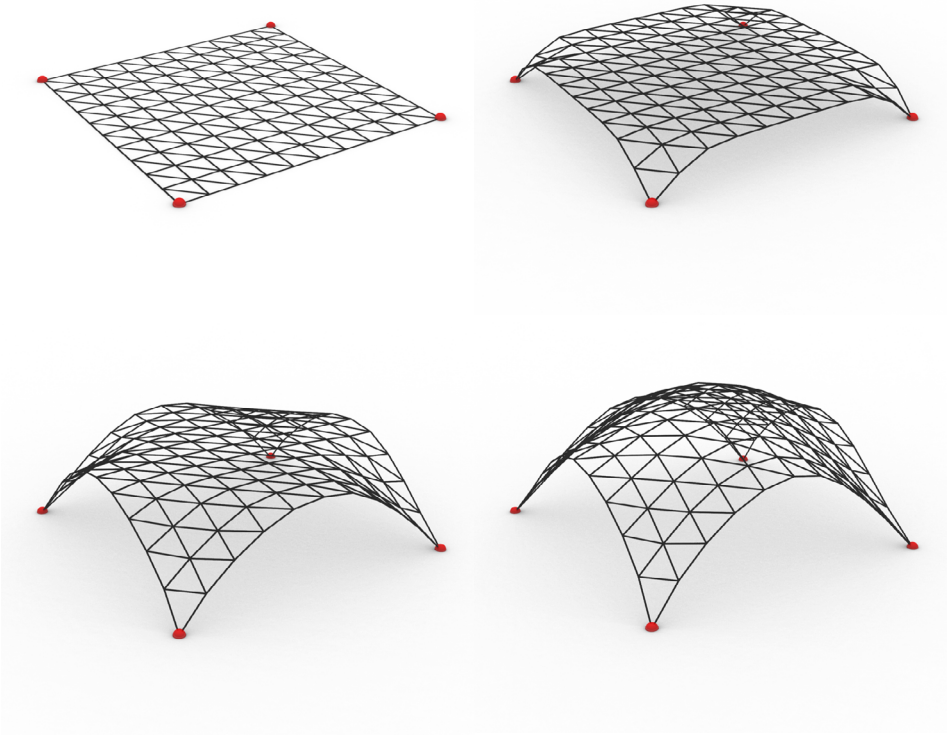
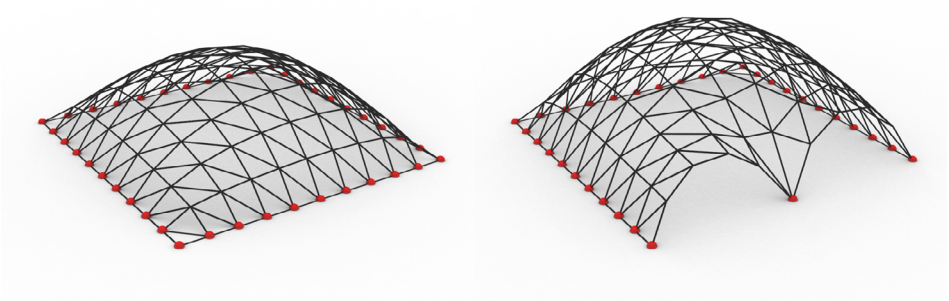


Figure 74: A simple triangular grid during the relaxation process. The planar grid and the support points illustrated with the red dots are fed to the DR component. The geometry is relaxed in real time so the user can see the optimization as it occurs. The shape of the final gridshell is the same that would be found had a physical model been created like Heinz Isler did for the Sicli Company building.

Figure 76: Different initial support points with the same grid as in figure 66.



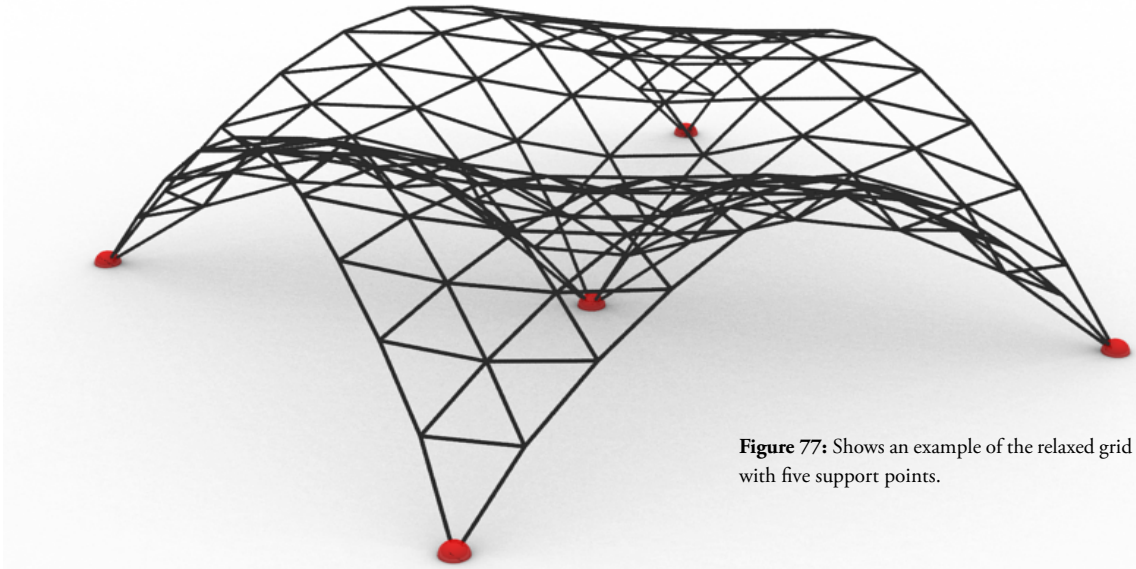


Figure 77: Shows an example of the relaxed grid with five support points.

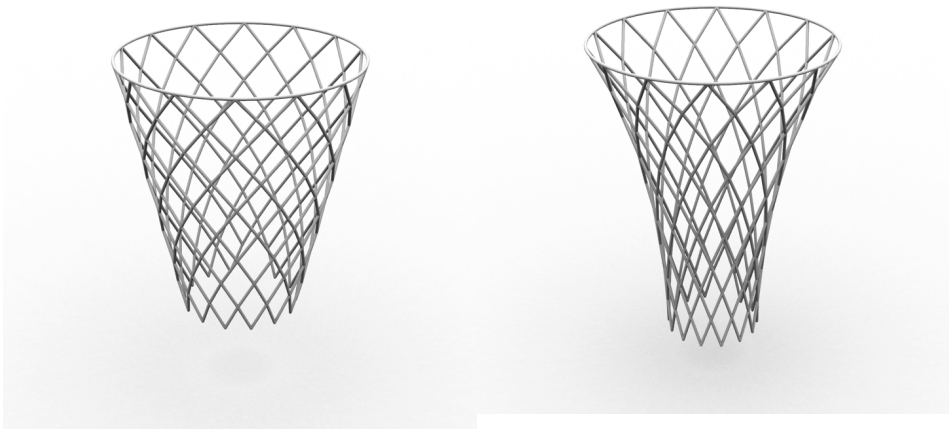


Figure 78: A basket ball net was created using the algorithm to recreate a real world object. This experiment was done to demonstrate the physical realism of the dynamic relaxation algorithm.

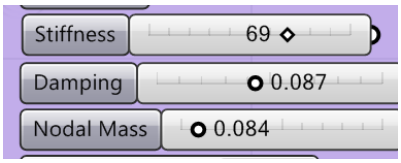


Figure 79: Sliders control the variables that impact the shape of the relaxed grid. Changes made with the sliders are carried all the way to the finite element model so the impact can be seen immediately.

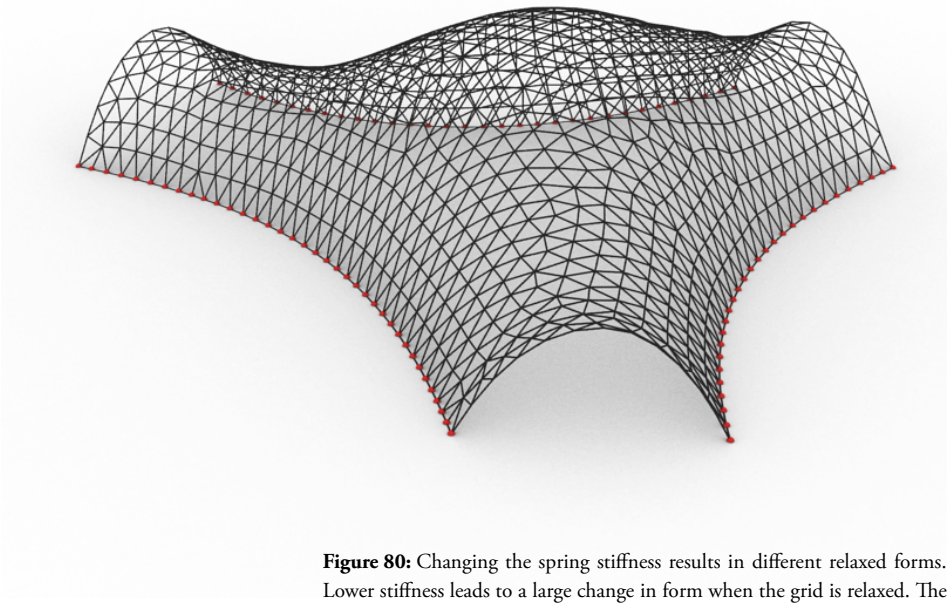


Figure 80: Changing the spring stiffness results in different relaxed forms. Lower stiffness leads to a large change in form when the grid is relaxed. The designing team can try to balance the stiffness to achieve a design that fits the visual, structural and functional requirements of the gridshell.

The geometric alteration of the grid will always increase the size of the grid during dynamic relaxation. This is because the springs can never be shorter than their initial rest length.

4.3.4 Integrated columns

One of the major strong points of optimized free form gridshells are their ability for very long spans without the need for columns. Sometimes columns will be needed if the gridshell must cover a very large area. Columns can be beautifully integrated into the grid-shel. Integrated columns can be designed with the form finding tool. This is done by simply fixing the grid where the base of the integrated column needs to be. The base position of the column can be optimized using genetic algorithms. This is further explained in chapter 4.8.

Figure 81: Integrated column at New Fair Milano. Image source: inhabitat.com

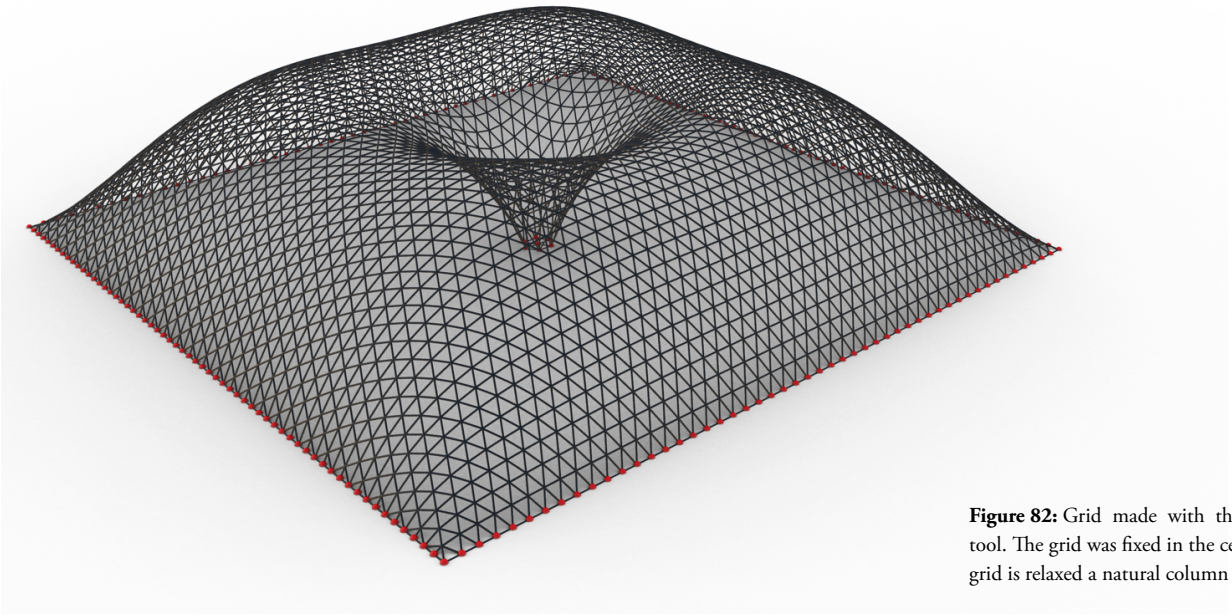
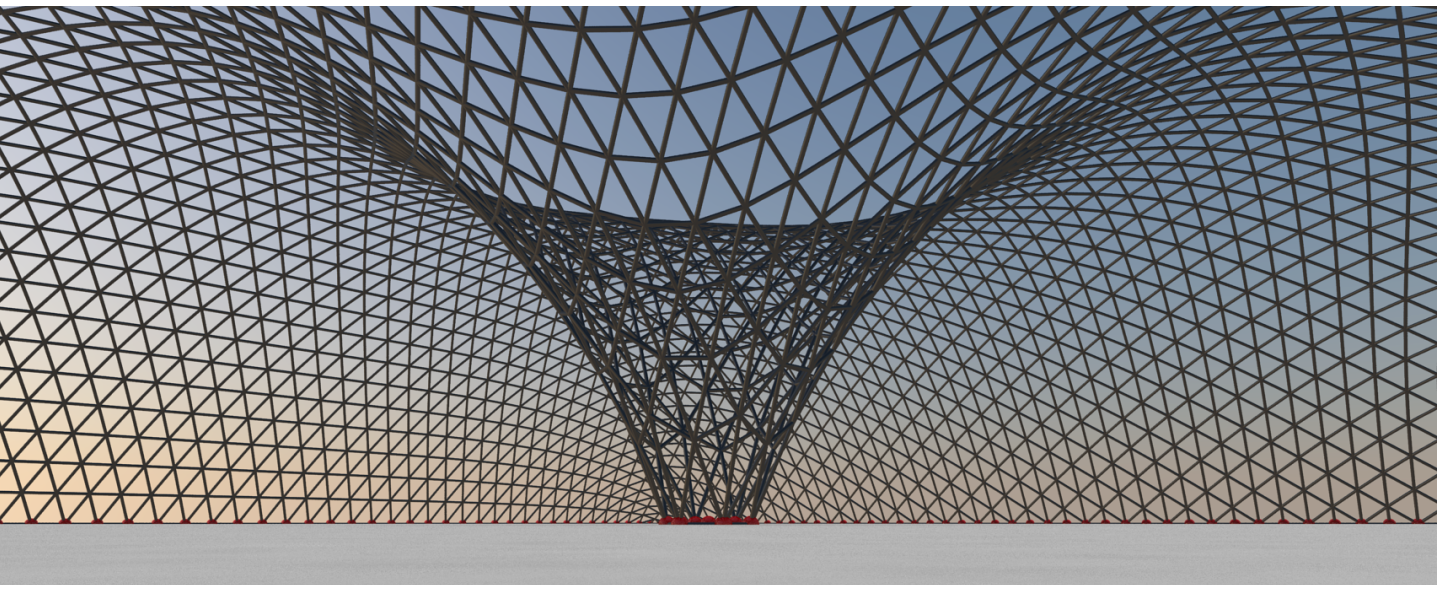


Figure 82: Grid made with the form finding tool. The grid was fixed in the centre. When the grid is relaxed a natural column will form.



4.4 Dynamic structural analysis

Figure 84: Karamba is a lightweight, flexible and user-friendly finite element software.

The dynamic FEM-software Karamba is implemented into the tool in order to quickly verify the structural grids produced with the grid algorithms and the dynamic relaxation component.
Karamba is a typical finite element program with all the standard functionality of a commercial structural analysis program. It does however come with one major difference. It is embedded within the parametric Grasshopper environment. This means that the geometric inheritance and benefits of parametric modelling are carried all the way into the structural analysis. The parameterization means that a change in initial geometry will change the finite element model instantly. This allows the user to experiment with the design while receiving instant structural feedback making it great for conceptual design phases.
The FEM-component is coupled to the dynamic relaxation component so that it uses the relaxed grid lines and the support points for its structural modelling. This way the FEM model is created automatically and FEM analysis results can be seen while the dynamic relaxation algorithm runs. This allows the user to manipulate the slider inputs to the form finding components to find a solution that provides the structural efficiency with a desired.

The grid demonstrated in figure 74 will be used to illustrate some of the functionality of the FEM-component.

4.4.1 Support conditions

The support positions are taken directly from the grid-model used for form finding. The supports can be altered to be any kind of desired support, whether it is simple supports, fixed supports or a combination of both.

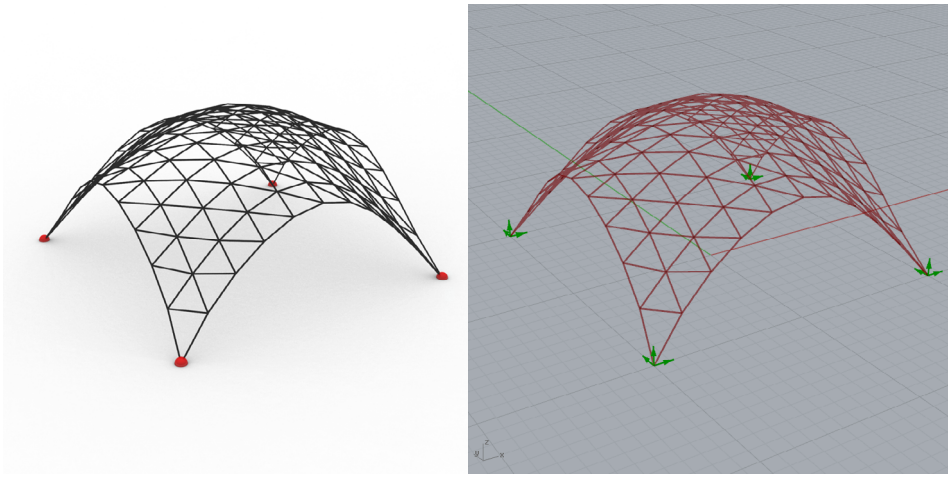


Figure 83: The support points will be directly transferred to the FEM model. The user then has to specify how the supports function.

4.4.2 Loads

Loads can be combined in different ways. Separate loads cases can be created to analyze different situation. Loads are typically applied in three different user specified ways. Once the loads are setup for a type of structure they do not need to be manually changed every time the structure changes. This allows the user to freely change the form of the structure and see how it will resist the applied loads. The desired scenario is that the user defines the governing load cases on the structure so that realistic loads are applied during the form finding process. This ensures that the conceptual model contains a high informative value that are normally found in the detailing of the project.

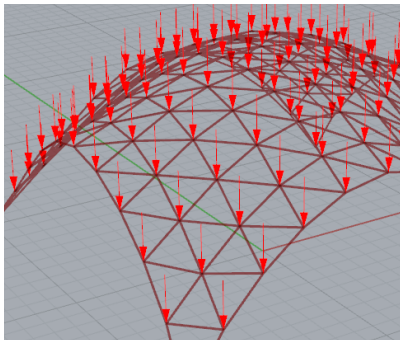


Figure 85: Nodal loads applied to the gridshell model. The Nodal loads will be applied automatically once setup in the FEM component. This only needs to be done once.

The loads on the grid structure can be applied in three ways:

1. **Nodal point loads:** The user defines the point loads as vectors acting on every node. This can be a evenly distributed point loads or a desired combination of unevenly distributed loads in varying sizes and direction.
2. **Line loads:** The user can specify line loads on the structural members. The user specifies the orientation and size of the loads.
3. **Mesh surface loads:** The user can create meshes within the grid-cells and apply load to these meshes. The FEM-software will then distribute the projected surface loads to the structural members as line loads.

4.4.3 FEM analysis results

The results of the FEM-analysis are presented to the user in a very visual way. Section forces such as axial forces, bending forces and shear forces can be mapped on the structural elements to show exactly where and how large they are. The same can be done for member utilization, calculated using the method found in Eurocode 3, and nodal displacements.
This visualization can be colour coded to provide an intuitive verification process. The colour coding also helps communicate the results to people who are not comfortable with FEM analysis results such as most architects.
Member stability is checked with a buckling analysis found in DS/EN 1993-1-1 2007 6.2.

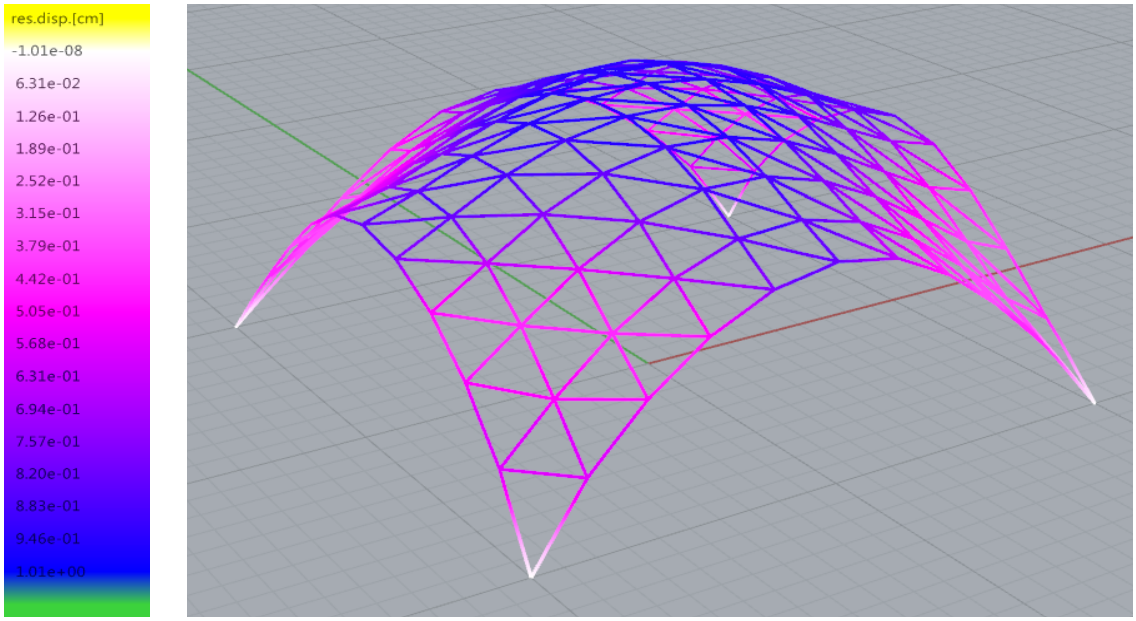


Figure 86: Nodal displacements can be visualised with colour coding to quickly see where nodes deflect the most. The list on the right side of the figure shows the colour gradient to decipher the nodal displacements.

The FEM software can also provide the natural frequencies of the structure and display the deformation shapes of the desired mode shapes. The natural frequency of the structure is dependant on the structural stiffness, support conditions and member mass.

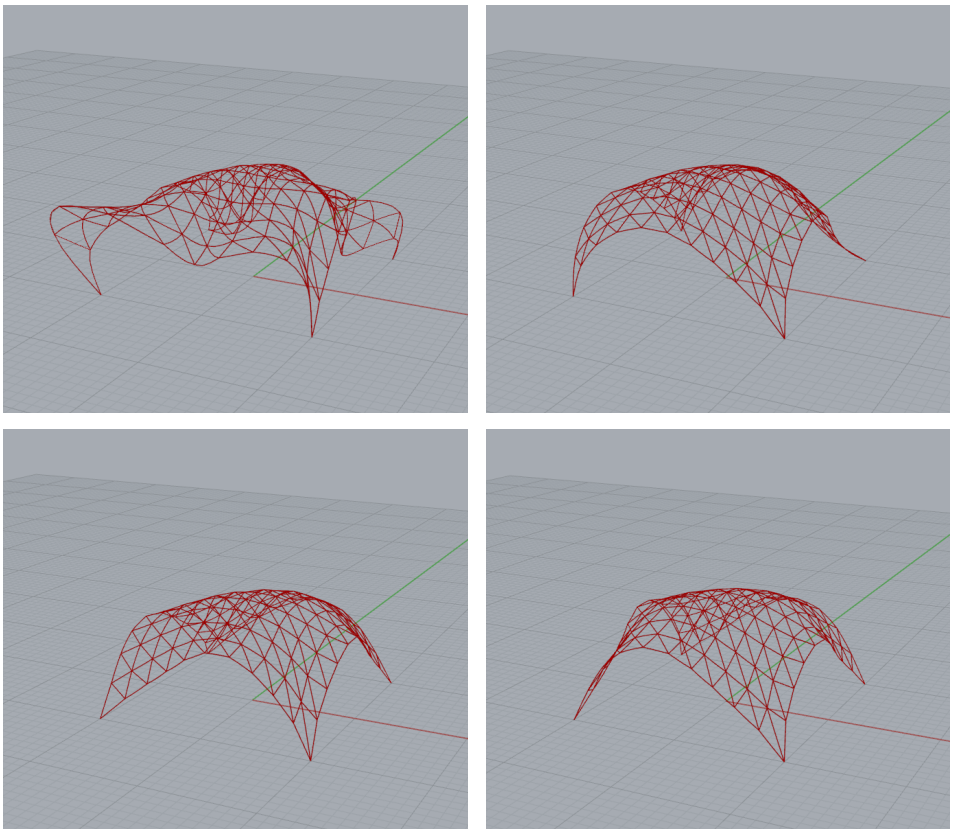


Figure 88: The first four mode shapes of the grid structure.

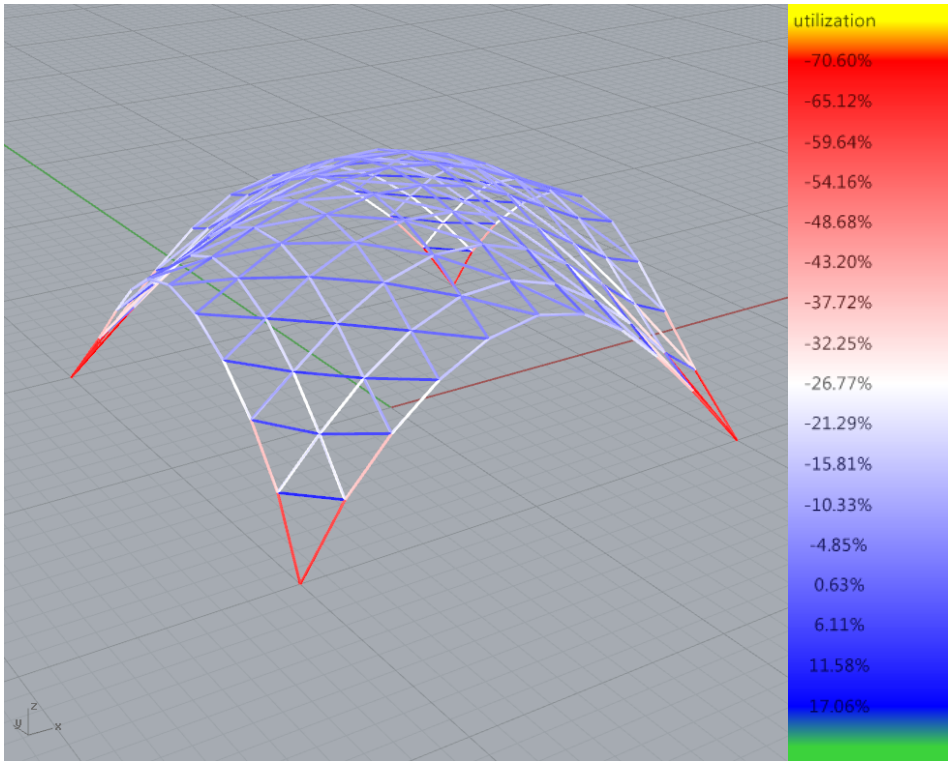


Figure 87: Member utilization can be visualised with colour coding to quickly see where members are utilized the most. This allows for intelligent section optimization. The list on the right contains the colour gradients that lets the user decipher the member utilization.

4.5 The structure and flow of the developed tool

The three key components of the tool in the form of grid generation, dynamic relaxation and structural verification have been described separately in chapter 4.2, 4.3 and 4.4. All three are implemented together in the same graphical user interface through the Grasshopper platform. The diagram below will illustrate the workflow of the design process from start to finish when using the developed tool.

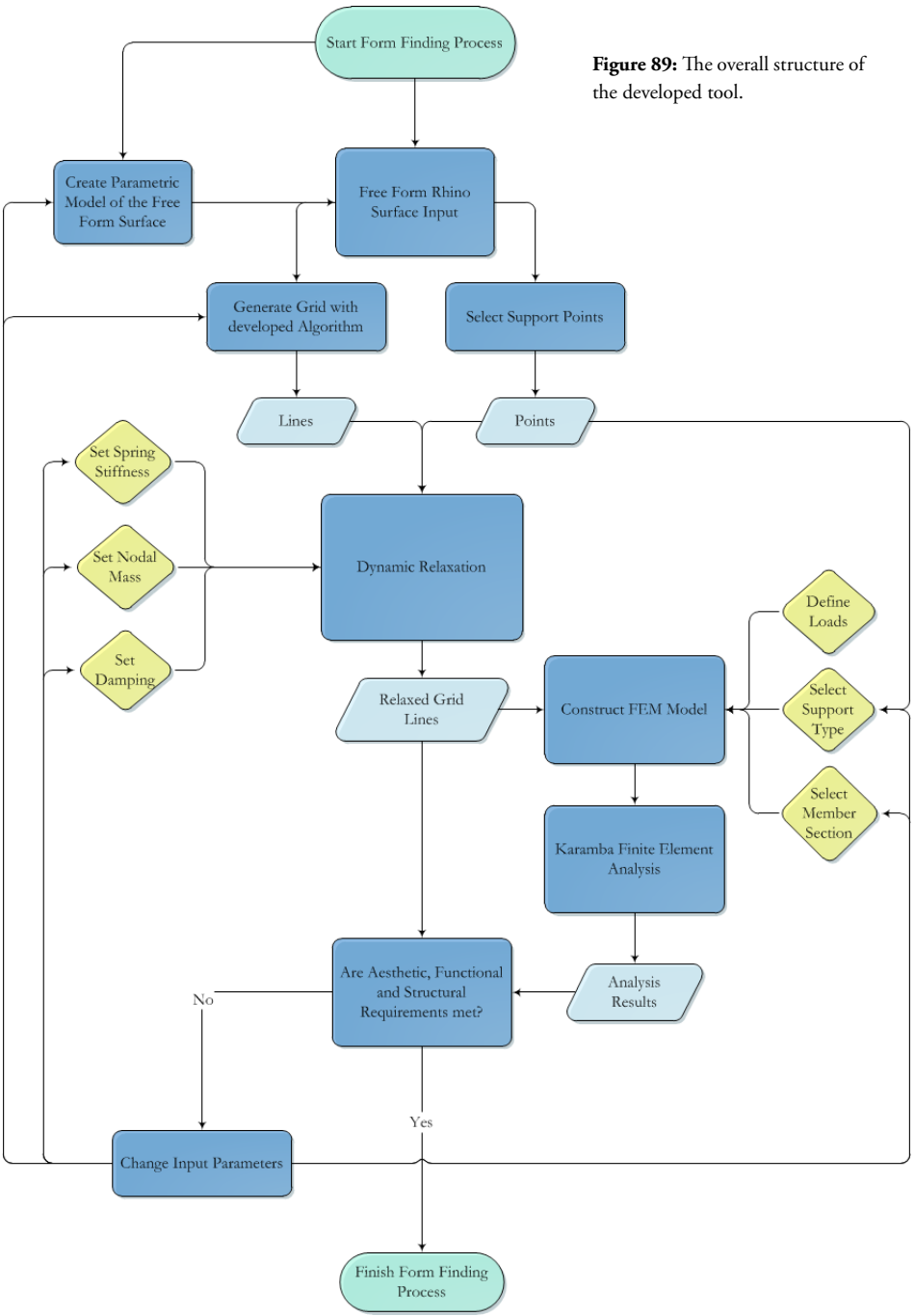


Figure 89: The overall structure of the developed tool.

The process start by designing a parametric model that creates the targeted free form surface. This surface is tesselated using the grid algorithms developed in chapter 4.2. This process results in a grid of lines. The designing team decides where supports should be placed on the grid lines. The grid lines and support points are run through the dynamic relaxation algortihm. The variables such as spring stiffness and node mass can be changed to try out different results.

The relaxed grid lines are then used together with the support points to create a FEM model using Karamba software. The user needs to define loadings, sections types and support options for the FEM model. The FEM model is then analyzed resulting in visually represented structural data.

This process is continued iteratively by chaning the input parameters to quickly massproduce different concepts.

4.6 Performance of the developed tool

This chapter will use small examples to test the developed tool. This is done to provide proof for the claims made in earlier chapters that the relaxed grid will perform better structurally after relaxation. The tool will also be used to solve the case in chapter 5. The examples will also try to illustrate how the tool can be used to rapidly perform form finding that would take a long time to do with physical modelling.

4.6.1 The catenary equation

It is possible to derive an equation that idealizes the physical hanging chain under own weight known as the catenary (Latin for chain) equation. Mathematically the function is a hyperbolic cosine function that when graphed represents the chain form under own weight while supported at both ends. The equation is derived in Appendix A2 and the result is:

$$y = a \cosh \frac{x}{a} \tag{14}$$

This equation is interesting seeing how it can be used to test the shapes found by the dynamic relaxation script. The shapes found using the dynamic relaxation script should resemble the graph of the catenary function given that the two has equal parameters.
In order to test this, a discretized chain of line elements was fed to the dynamic relaxation component and compared to the graph of the catenary equation. The equation and the dynamic relaxation component had their parameters such as nodal mass and stiffness adjusted so to achieve maximal resemblance.

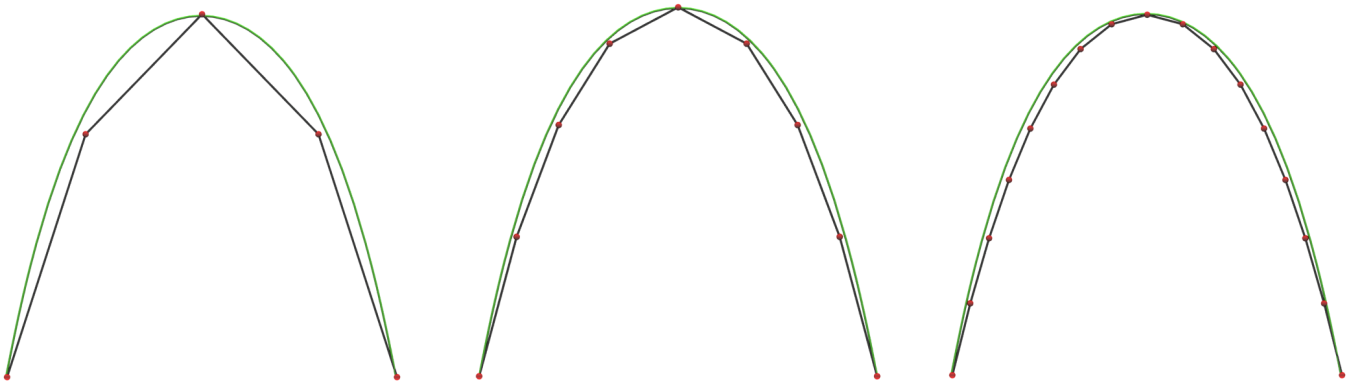


Figure 90: The green line is the graphed catenary equation derived in Appendix A2. The black line is the relaxed chain found with the dynamic relaxation component. When the number of discrete line pieces are increased the resemblance of the catenary equation increases. The figure shows examples with 4, 8 and 16 discrete line pieces.

The experiment shoved that a grid with sufficient amount of discrete lines will resemble the catenary equation. This information is important when developing gridshell designs. There should be enough structural elements so that the dynamic relaxation can relax the grid into a state where membrane behaviour can be achieved.

Figure 91: The Gateway Arch in St. Louis. It was designed by architect Eero Saarinen structural engineer Hannskarl Bandel in 1947 and constructed during the early sixties. The shape was derived using the catenary equation to create an arch in pure compression. Image source: http://en.wikipedia.org/wiki/Gateway_Arch

Figure 93: The Mannheim Multihalle. Image source: http://www.smdarq.net/

Figure 92: Hanging chain model of the Mannheim Multihalle. Image source: [Toussaint, 2007]

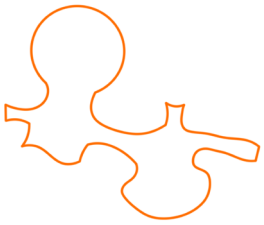


Figure 94: The aerial photo of the gridshell and the outline of the structure.

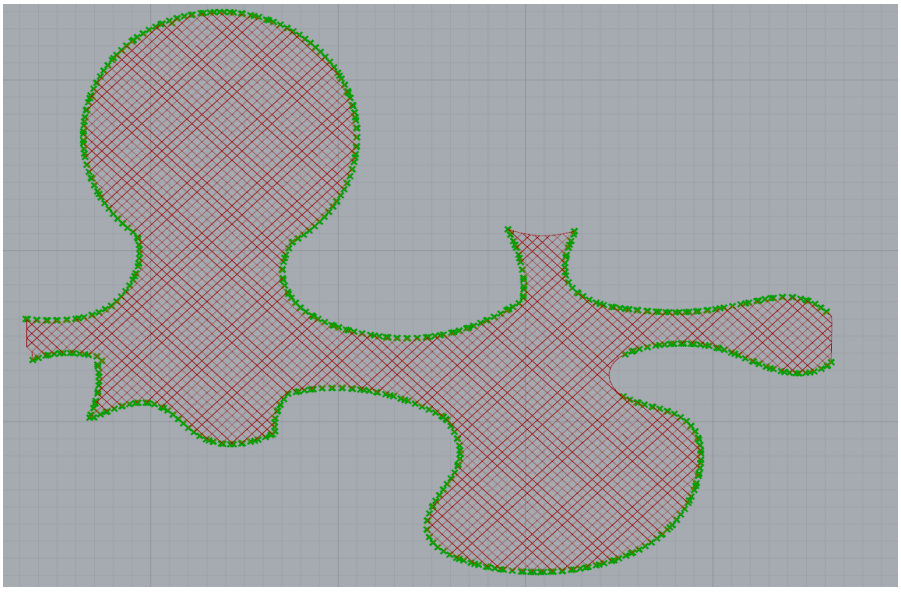


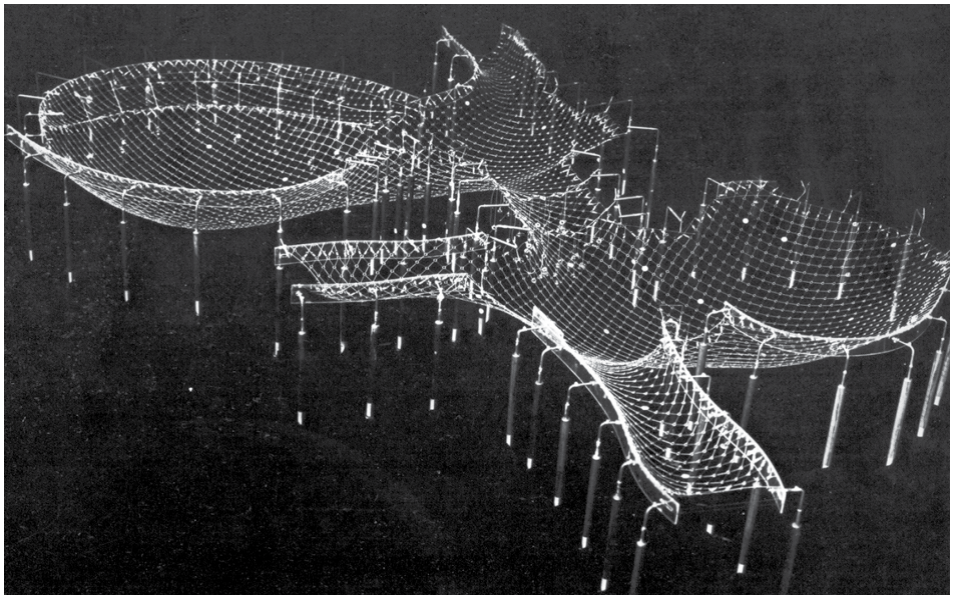
Figure 95: The grid created to simulate the Mannheim physical form finding. The green crosses represent supports where the grid will be fixed for movement during the relaxation process.

4.6.2 Recreating the initial form finding of the Mannheim Multihalle

This example will roughly reproduce the form of a complected gridshell without spending a lot of time modelling. The chosen gridshell is the Mannheim Multihalle.

The Multihalle in Mannheim, Germany, is a timber gridshell designed in collaboration by Frei Otto and Ove Arup and Partners. The gridshell covers a large area and was build to house a garden expo in 1975.

The initial shape of the Mannheim gridshell was found entirely using physical modelling with hanging chain models as described in chapter 3.4.1 [Toussaint, 2007]. The physical model was made in a 1:100 scale where only a third of the total grid lines were represented. The relaxed physical model was converted to three dimensional computational coordinates using stereo photography. This was obviously a long, tedious and inaccurate process. Errors in the initial physical model will be enlarged when transferred to full scale construction [Toussaint, 2007]



The outline of the structure is needed, in order to recreate the physical model shown in figure 92. A simple outline was sketched using an aerial photo of the area shown in figure 94. A flat grid was placed within the structural outline. This was the grid that should be passed through the dynamic relaxation algorithm to achieve the same result as seen in figure 92.

The relaxation of the grid took four minutes without spending time to increase the rate of convergence by altering time step, nodal mass and damping. Figure 96 illustrates the output of the relaxation process that allows the user to follow it in real time.

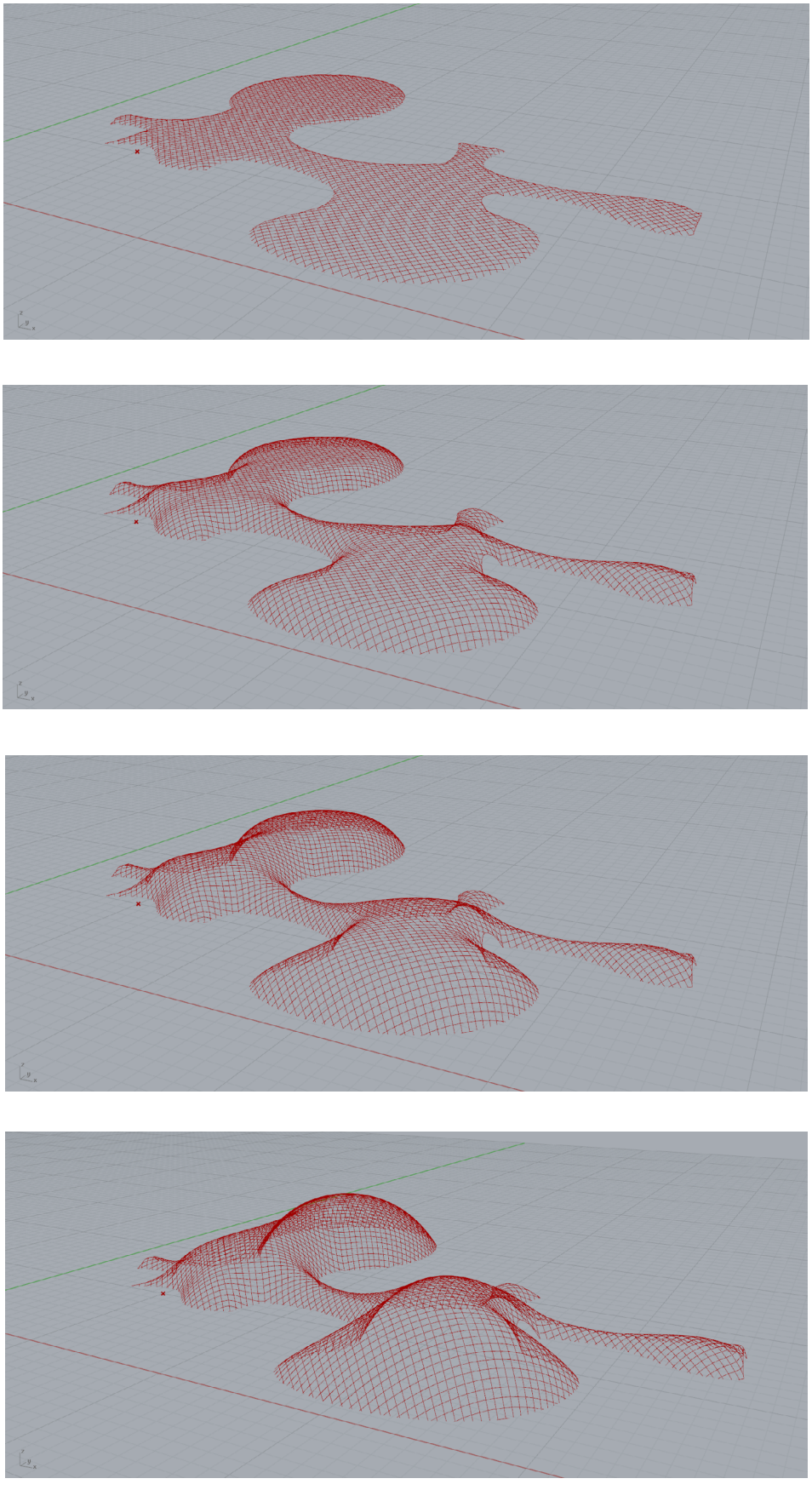
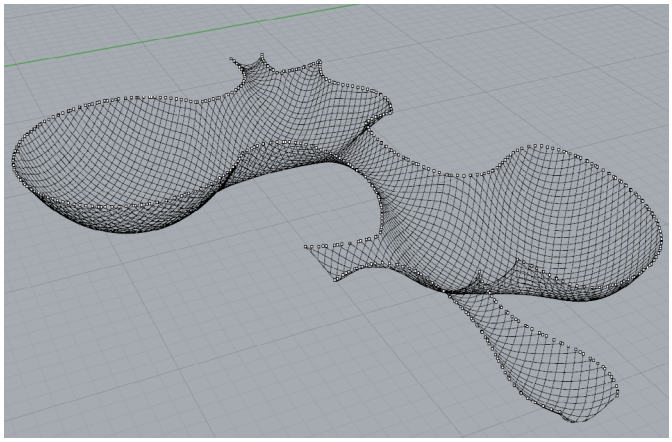


Figure 96: The computational result of the dynamic relaxtion form finding is shown on the four images.

The Mannheim example was produced in less than one hour showcasing the potential of the developed tool to reduce the amount of time spent modelling. The tool also improves upon physical modelling by allowing the user to alter parameters in the process to achieve different iteration of the concept. Doing so with physical models is a very time consuming process.
The results of the dynamic relaxation can be directly analyzed with the implemented FEM software to further explore the structure.



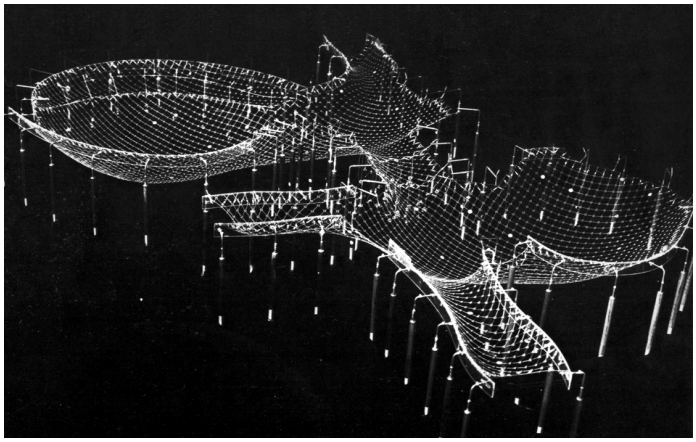


Figure 97: The computational result of dynamic relaxation vs. the original physical model made by Frei Otto and his team. The same form is achieved with a fraction of the time spent modelling.

The parametric model can also be used to give the designing team a good idea of the visuals of the structure. This can be further evolved by creating photorealistic rendered images and videos of the structure. The image in figure 98 is taken directly from the viewport of Rhino and is not a rendered image. Realism can thus be improved to create a genuine aesthetical insight in an conceptual structure.

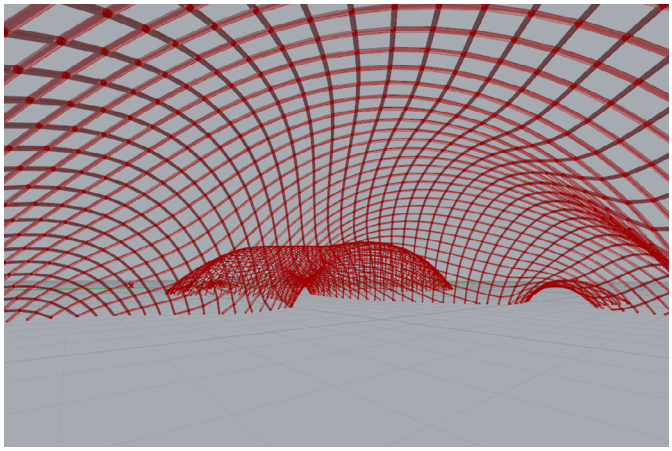


Figure 98: Inside view of the Gridshell.

4.6.3 Geometric structural optimization

In passages 4.6.1 and 4.6.2 it was shown that the developed tool was able to simulate physical form finding with the dynamic relaxation algorithm. It is interesting to see what sort of structural impact the relaxation will have compared to an un-relaxed structure.
The vision for the form finding tool is that it should be used by designers that wish to create free form structures. The example shown in 4.6.2 with the Mannheim Multihalle can not be seen as a completely free form as it is the result of the physical model and its boundaries. The initial input is a completely flat grid, and everything else in terms of shape is controlled by gravity.
The tool developed in this thesis takes a free form surface as initial input. This gives the designing team a much larger freedom in exploring possible shapes which is essential to a unrestricted design process. The relaxation algorithm of the tool should help improve the membrane behaviour of the grid and thus improving upon the selected form. This form of structural optimization is called geometric optimization as it requires that the geometry of the building can be changed. It is therefore essential that this sort of optimization is performed during the initial stages of the development. There are other types of structural optimization that can be performed at a later stage of the process such as material- and member cross-section optimization.

To verify the capabilities in terms of geometric structural optimization a test was performed using the input form shown in chapter 4.2.5. The test had the goal to verify that the dynamic relaxation would radically reduce bending stresses and nodal displacements in a gridshell structure while improving the membrane behavior of the gridshell.
The form was first tessellated into a triangular grid with the developed equilateral triangle grid tool.

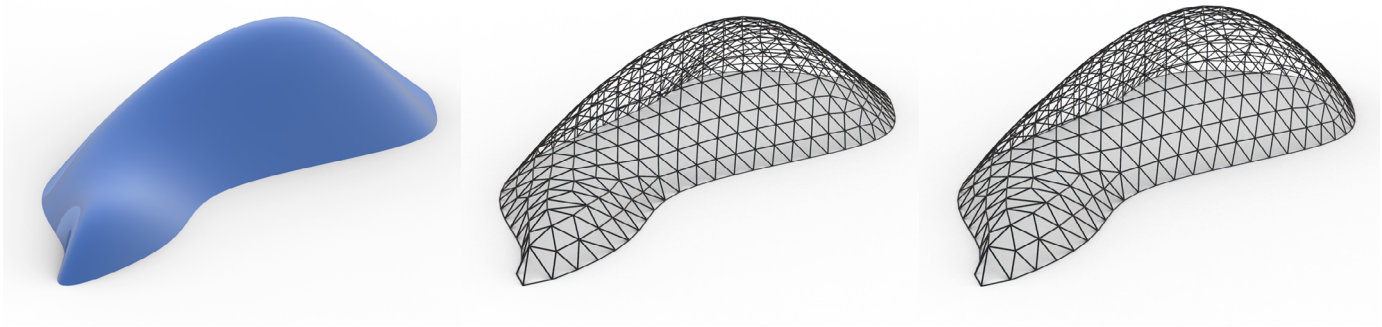


Figure 99: Shows the initial input free form surface to the left, the un-relaxed grid in the middle, and the grid after dynamic relaxation to the right.

The triangular grid was then relaxed using the developed dynamic relaxation component. The dynamic relaxation was done with high spring stiffness to assure that the relaxed grid would be as close as possible to the input grid staying true to the input aesthetics.
The dynamic relaxation will still change the geometry slightly which is the whole point. Lightly rearranging the structural elements will achieve better structural performance. The dimensions of the relaxed and un-relaxed grid were as follows.

Table 1: Dimensions of the example geometry

	Max. Height [m]	Max Width [m]	Max. Length [m]
Un-relaxed	20.5	53.0	114.5
Relaxed	22.0	53.0	114.5
Change	7.3 %	0 %	0 %

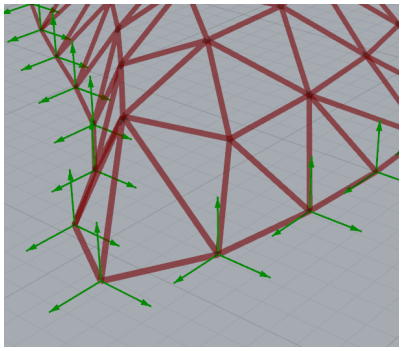


Figure 100: Supports in the FEM model.

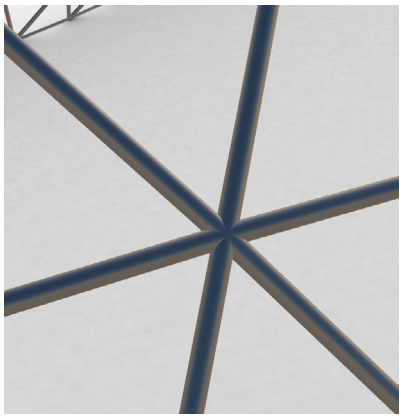


Figure 101: Close up of the circular pipes that was chosen in this FEM model.

A small change in overall structural height was one of the results of the dynamic relaxation. This small change is obviously acceptable in this test scenario, but in a real world situation a smaller increase might be more preferable. This can be achieved by using a higher spring stiffness in the relaxation process at the expense of lesser structural optimization. It is a consideration the designing team has to make in collaboration.

The un-relaxed and relaxed grid was analyzed using the attached Karamba FEM-software in order to compare section forces, maximal nodal displacements and member structural utilization. The relaxed and un-relaxed grid models were given the exact same simple support conditions shown in figure 100.

Dynamic relaxation optimizes the structure to better resist evenly distributed dead loads. In order to test this the two grid models were applied the same loading case with evenly distributed arbitrary nodal forces of 25 kN in addition to the self-weight of the structural members.

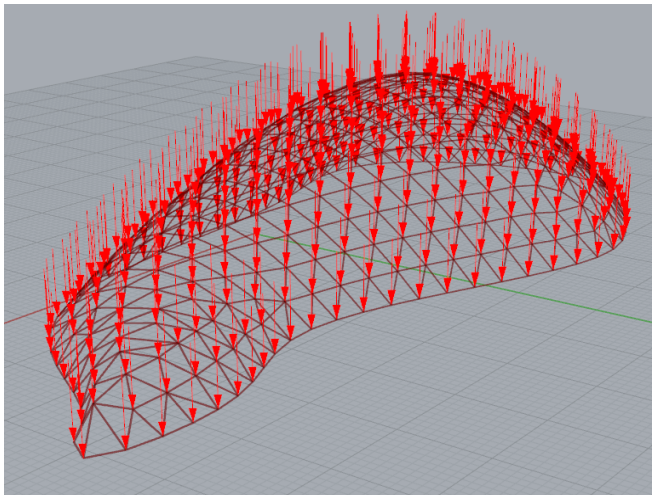


Figure 102: Distributed nodal loads in the FEM-model. Loads are defined with load vectors and act in the structural nodes of the FEM-model. Loading can be changed quickly to check different cases.

Circular 20 centimeter steel pipes with a 1 centimeters thickness were selected as member sections for all grid members. The average member length is 4 m.

The five largest bending moments and axial compression forces in the relaxed and un-relaxed gridshell structures are shown in table 2 and 3. The maximal bending forces in the structure are reduced by 95 %. The axial section forces are much more evenly distributed throughout the structure resulting in much smaller axial forces and almost nonexistent tension forces after relaxation. This improved member behaviour also results in more evenly distributed reactions.

Table 2: Maximal bending forces in the un-relaxed and relaxed grid.

	Maximal section bending forces				
Un-relaxed [kNm]	30.5	28.9	27.1	27.0	26.8
Relaxed [kNm]	1.53	1.48	1.48	1.43	1.39
Change [%]	95.0	94.9	94.5	94.7	94.8

Bending forces are reduced to almost nothing as the theory of dynamic relaxation proclaims.

Table 3: Maximal compression forces.

	Maximal section compression axial forces				
Un-relaxed [kN]	-639.5	-584.2	-515.6	-501.6	-484.7
Relaxed [kN]	-175.1	-173.6	-173.6	-172.5	-172.2
Change [%]	72.6	70.3	66.3	65.6	64.5

Table 4: Maximal measured tension forces.

	Maximal section tension axial forces				
Un-relaxed [kN]	330.2	326.4	322.6	309.1	308.1
Relaxed [kN]	34.5	27.3	27.0	24.5	24.1
Change [%]	89.6	91.6	91.6	92.1	92.2

Axial forces are distributed much more evenly among the structural members due to the highly enhanced membrane behaviour.

This large reduction in bending forces and axial forces in the structure results in huge improvements in member utilization. The structure did not pass the structural verification demands given in EC-3 before relaxation with a maximum member utilization of 123 %. After relaxation the maximal member utilization was only 17 % due to the much better distribution of forces in the structure. This optimization allows for much smaller section dimensions and still have utilization well within the allowable range.

Table 5: Member utilization calculated via Eurocode 3 DS/EN-1993 section 6.3.3

	Member utilization				
Un-relaxed	1.225	1.123	1.092	1.016	0.951
Relaxed	0.171	0.170	0.169	0.168	0.168
Change	86 %	85 %	85 %	83 %	82 %

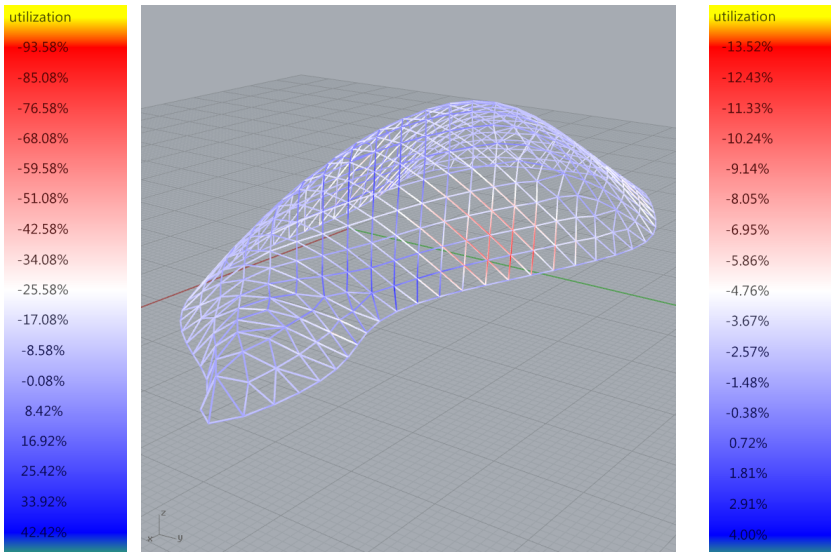


Figure 103: The figure below shows member utilization of the selected cross section. The image on the left shows member utilization before relaxation and the image on the right shows utilization after.

The overall gridshell structure is also much stiffer after relaxation seeing how the structure now posses better membrane behaviour. This results in a 97 % reduction of maximal vertical nodal displacement and an increase in natural structural frequency making the structure less likely to resonate with typical dynamical loads.

Table 6: Max vertical nodal displacements

	Maximum nodal displacement [cm]
Un-relaxed	13.9
Relaxed	0.45
Change	97 %

Table 7: Natural frequence of the relaxed and un-relaxed gridshells

	First natural structural frequency [Hz]
Un-relaxed	2.8
Relaxed	4.2

The overall improvements are even more impressive given the mere 5 seconds the dynamic relaxation algorithm needed to optimize the structure.

While the results look very promising, it has to be mentioned that this sort of optimization will see largest improvements when analysed for uniformly distributed loads. When unevenly distributed or concentrated loads are analysed the gains are not as great percentage wise. But, the optimised grid will be better at resisting uneven loads than the input grid.

4.7 Calculating geometry needed for fabrication of members and nodes

Gridshells are geometrically very complex structures with a very large amount of unique structural members and nodes. Automation is needed in order to fabricate elements and build the structure efficiently as mentioned in chapter 4.1. This passage will explain how the geometry used within the form finding tool can be used to generate the information needed to fuel this automated process. The passage will focus on how the geometry is retrieved, as it relates directly to some of the considerations already made when developing the dynamic relaxation component. The algorithm that actually creates the nodal geometry will not be designed in this thesis, but the form finding tool is prepared to output the necessary information to do so.

4.7.1 Designing the structural joint

Hollow, I- or T-sections are the better choice for free form gridshells. Solid sections will provide only slightly slimmer cross sections at the expense of highly increased structural weight. This increase in weight will negatively impact deflections, bending stresses and buckling factors [Barnes and Dickson, 2000]. The most simple solution is to use circular hollow cross-sections that meet in spherical nodes, but this is not always preferable visually or when designing the panels that rest on the members.

Figure 104: Different types of rectangular hollow cross-sections. Image source: made-in-china.com

Designing structural joints for gridshell structures build with hollow sections is very complex due to the double curvature of the shell. In a free form gridshell every structural joint is unique, and every structural member is oriented differently spatially. In order to design the structural joints, specific orientation vectors for all structural members and structural joints that connect them are needed. The needed vectors are luckily already present within the form finding tool as it is needed to generate the line geometry of the grid. Once the grid is ready for production these vectors can be output. The output orientation vectors will be used for calculating the geometry of the joints and the structural members in a fully automated process. Specific algorithms must be developed that use the vectors to generate specific nodal and member geometry that can be read and produced by laser cutters, flame cutters or CNC-mills. The entire process needs to be designed so everything is fabricated automatically due to the often immense number of unique joints and members.

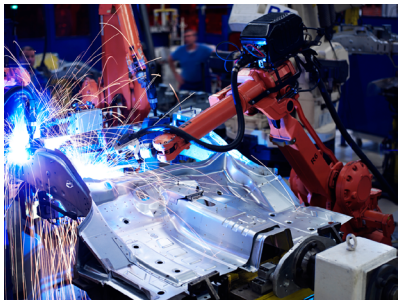


Figure 105: Robot arm here seen welding. The robot arm can be adapted to cut instead. The arm has high movability and is able to cut in many possible angles. Image source: seetech-corp.com

In order to describe the difficulties faced when developing structural joints for gridshells a simple example will be shown. The example shows a gridshell generated with the design tool. The output is a grid of lines with varying orientation that describe the varying curvature of the shell. The process of orienting the structural members so their sections can be connected to structural joints become very complex when not dealing with circular cross sections. The varying curvature of the surface means that the member ends are twisted relatively to each other when they are connected to two adjacent nodes. Information about the normal vector and tangent plane at nodes are needed, in order to correctly orient and place prismatic cross-sections such as rectangular-, T- or I-sections. The tangent plane and normal vector are used to calculate specific vertical, horizontal and warped angles that are processed in the joint algorithm. Figure 106 shows the gridshell used to demonstrate how the structural members are differently oriented. The members that will be looked at are marked in yellow.

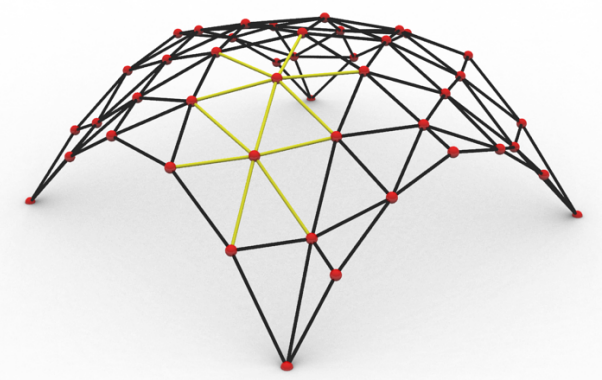


Figure 106: Simple gridshell used to illustrate the different orientation of the structural members. The members looked at are marked in yellow.

The normal vectors of the nodes must be found as the average of the member vectors that all connect at the node. The plane perpendicular to the normal vector at the centre of the node is the tangent plane.

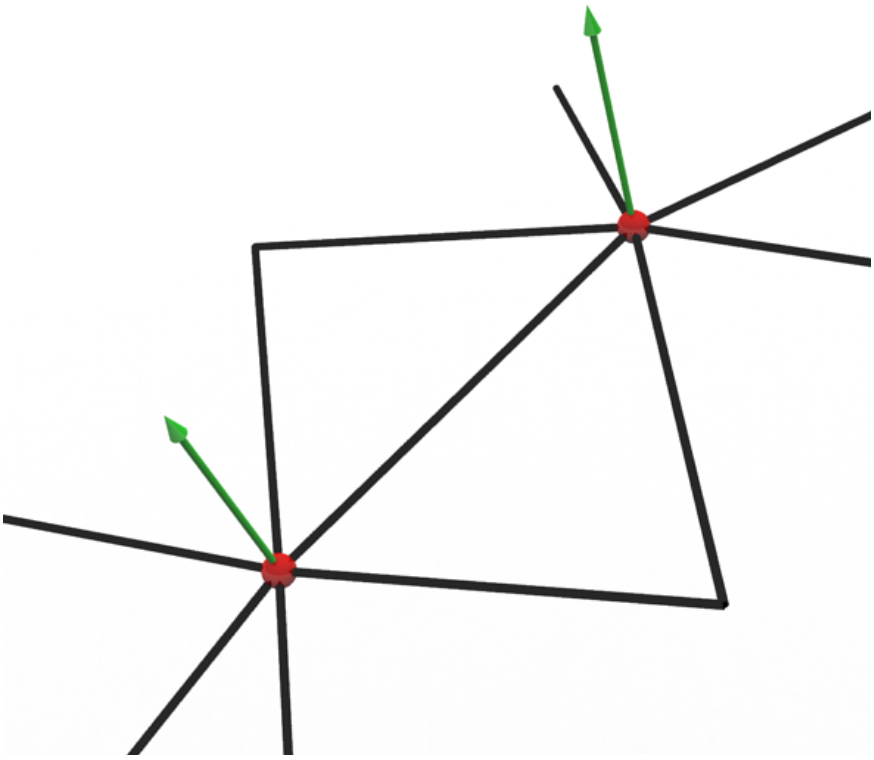


Figure 107: Normal vectors, coloured green, of the structural nodes. These vectors will be used as the basis of orienting the members and nodes correctly. They will also be used to produce geometry that is fed to the automatic cutting and welding devices to produce structural elements.

Every structural element is always connected to two adjacent nodes. When dealing with prismatic cross-sections the different curvature at two adjacent nodes will cause the member ends to be warped in respect to each other. This is illustrated in figure 108 by letting a plane be spanned by both normal vectors and the member vector.

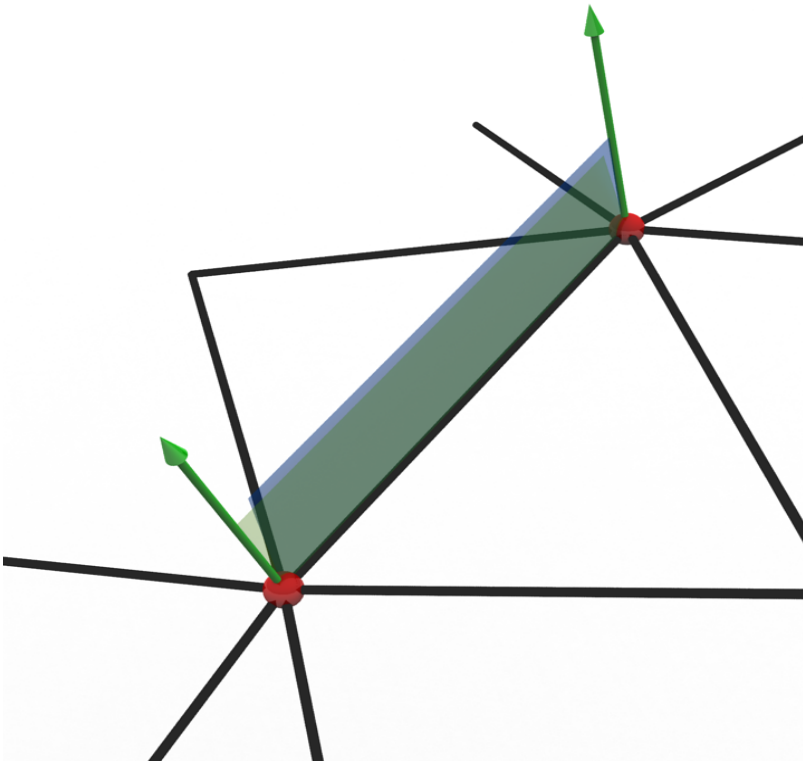


Figure 108: The warping angels of the member ends are illustrated by spanning two planes using the orientation vector of the structural member along with the two nodal normal vectors.

The warping angle is important when designing the structural nodes in order for everything to fit together. The angles grow larger as the curvature grows larger. The design of the structural joint or the structural element must be able to accommodate these warping angles. The warping angle also has to be taken into consideration due the fact that the panels are often supported on the upper flange of the structural members. If the warping angle is very large the panels cannot rest evenly on all flanges of the cell. The warping angle can be accommodated in the design of the node or the end of the structural member by cutting it at an angle.

Nodal deflections due to applied loads have to be taken into consideration when designing the joints. This is because the deflection change the angles between the structural members and the nodes. A way to solve this problem is to add the calculated nodal deflection in the opposite direction and create a new translated grid geometry. This method is often tied to cast concrete slaps that are produced with a slight pitch in order to provide a completely plane surface when loads are applied.
If the gridshell is modelled with rigid joints that allow for bending moments to travel through the nodes, then the structural joint must be capable of performing this task. Most commercially available nodes are able to transfer bending moments and axial forces. The connection between the node and the members is usually done with welds or bolts. The node itself is usually a steel disk or cylinder with a prismatic cross section. A few of these commercial nodes are shown on figure 111.
Any nonautomatic process during fabrication of elements and joints is unacceptable due to the amount of different elements that need to be produced. Node and element production is usually done with robotic flame or laser cutting arms.

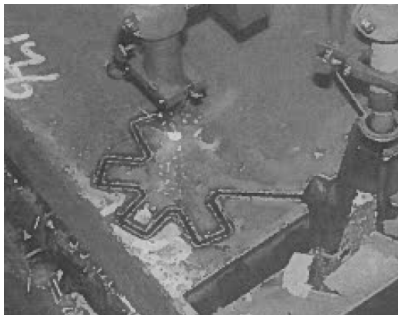


Figure 109: A flame cutter making a structural node for the British Museum Courtyard roof. Image source: [Barnes and Dickson, 2000]

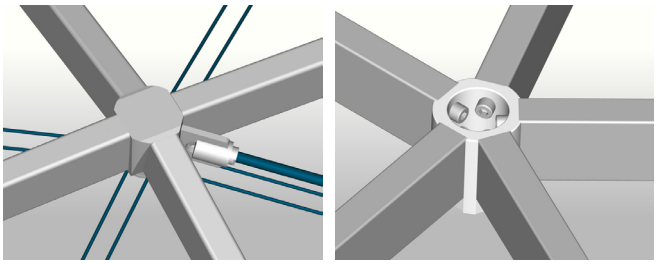


Figure 111: Block and cylinder nodes by Mero. The nodes are individually fabricated so that all joints are unique. Image source: mero.de

The geometry needed to control the cutters will be programmed using special joint algorithms. A gridshell joint algorithm must be able to perform the following operations.

- Retrieve the orientation vectors of all members in grid and organise them in lists according to the nodes they are connected to.
- For each node an average normal vector is calculated as the average of the vectors joined in that node.
- Calculate vertical angles V between the nodal normal vector and orientation vectors connected in the node. This is illustrated in figure 110, top.
- Calculate the horizontal angle U between structural members depending on the grid configuration. This is illustrated in figure 110, middle.
- Calculate the warp angle W. Figure 110, bottom.
- Generate nodal geometry with the provided angles U, V and W along with section type and size.
- Convert the nodal geometry into instructions than can be fabricated by the robot arms.

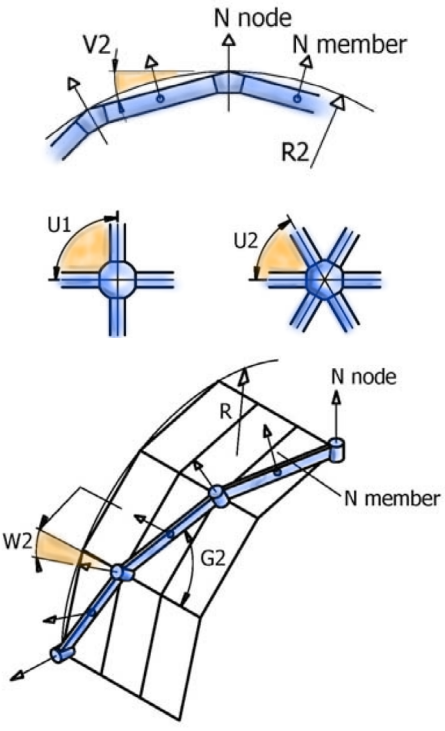


Figure 110: Different angles needed to produce structural joints. Image source: [Stephan, Sánchez & Knebel, 2013].

4.8 Further development of the tool

The following functionality could be implemented in the design tool for further improvement. Not all suggestions listed are tied to structural optimization. The main philosophy is to create a form finding tool that covers as many areas as possible. The result of the design process should be optimized in respect to functionality, technical design, aesthetics, economy, enviromental considerations and indoor climate. Most of the sugestions listed here are related to improving the geometric form finding to improve structural optimization.

- Implement the possibility to change spring rest-length in real time. Chaning the rest-length should provide the option to model prestresses in the structure, if the tool was ever to be used for a fabric structure.

- The generated dynamic relaxation component should be able to generate geometry from user specified load cases. Currently the algorithm generates geometry based on uniformly distributed gravitational loads.

- A form finding tool for wood or fibre glass gridshells could be developed by altering the particle spring system to include rotational springs. The rotational springs model the bending within the flexible materials. The idea is to find a equilibrium geometry by purposely deforming the gridshell into its final shape. The advantages of this approach is that the gridshell can be assembled flat on the ground and then raised or lowered into its final position. This is only possible for flexible materials which is why it will not apply for steel gridshells.

- One of the major issues designers might have with the form finding tool is that the dynamic relaxation algorithm will transform the shape of the input surface into a funicular shape. Even though these changes can be minimized these alterations will always be a part of the dynamic relaxation process. Alteration of overall geometry is necessary to find a relaxed grid where the lines are in equilibrium. This is why the dynamic relaxation component should be applied in a conceptual design process where the geometry is not yet set in stone.

- If the designing team needs the surface to have a specific form that is not allowed to be altered even the slightest, then dynamic relaxation will not prove the most helpful tool as it will make small alterations to the geometry. There are still means of structural optimization if alterations of the input geometry is not possible. Two traditional possibilities have already been mentioned. Cross-sectional optimization and material optimization are some of the most fundamental disciplines of structural engineering. A third possibility, relies on the ability to change the composition of the structural grid without changing the overall shape. Such algorithms can be developed with a matrix calculated form finding method called force density. The force density method can be coupled with genetic algorithms that simulate a survival of the fittest approach to optimizing the structure. A fitness criteria is selected, such as minimizing von Mieses stress or structural weight. The algorithm iteratively rearranges the structural members and when a better solution is achieved it proceeds to improve upon that until no further improvements can be obtained. The algorithm needs to be coupled with FEM calculations for the fitness algorithm to receive information about member stresses.

- The force density method could be coupled to the dynamic relaxation algorithm so that the tool would initially optimize the overall shape. Afterwards the force density algorithm would rearrange the structural grid to find the most structurally efficient solution.

- A fabrication algorithm should be developed that instantaneously generates the geometry of each structural node and member. The algorithm should posses the abilities discussed in chapter 4.7 and be able to produce fabrication plans for joints and members. The algorithm should also be able to structurally validate each node for stresses acting upon it.

- The form finding tool Sun could be coupled with indoor climate optimizations. Seeing how gridshells are often used as transparent roof structures the overall shape and penalization has a large impact on the indoor climate. Algorithms could be created that optimize sun shading and the orientation of the grid in respect to the sun.

Figure 112: Rendition of the UMS project designed by Schmidt Hammer Lassen. Image source: urbanmediaspace.dk

Figure 113: The building project as of June 2013. Image source: urbanmediaspace.dk

5. Case testing the form finding tool

The following case, detailed in 5.1.1, is selected to showcase a real world project that can be approached using the developed gridshell form finding tool. A choice was made to pick a real-world case that has yet to be built in order to provide a uninfluenced perspective to the solution while still have realistic geometric boundaries and context.
The design process of this case will be restricted in comparison to a real world scenario. The developed tool has to be front and center of the process in order to present the possibilities within the tool. Normally a designing team will decide what type of structure is applicable to a given project. If a gridshell is chosen, then the developed tool can be implemented in the design process. The design process of this case is obviously going to be a bit backwards. This is due to the fact that the type of structure must be a gridshell like structure as the form finding tool must be implemented in the process.

5.1 Introduction to the case

The main purpose of the case is to show the functionality of the form finding tool by using it to generate geometry of a free form gridshell structure. It is the intention to do so by developing different concepts to show how the tool can be used to quickly generate and optimize the geometry of different structures. This allows the designing team to explore a wide variety of possibilities .
The concepts are meant to include sufficient considerations that they will be deemed realizable. This will include rough structural dimensioning of members and suggestions to joint detailing and construction of the structure.
It is possible to find other, more obvious cases where the developed tool will perform even better solutions. Another case could be to genere the geometry of a light weight roof structure. But it was allready showcased in 4.6.7 that this type of structure is where the dynamic relaxation algorithm really shines. However, the following case was chosen because of it possibilities to create an controversial exciting complex free-form structure in an area where such type of structures are unusual. The case allows for wild and playful designs that really pushes the geometric limitations of the form finding component.
The case will be analyzed in chapter 5.2 to find several important design criteria that will be translated into a range of conceptual designs made with the form finding tool. The concepts will then be evaluated objectively and a winning concept will be chosen. Limitations to the case will be introduced throughout chapter 5.2. These limitations are introduced to reduce the scope of the case so that the focus will remain on showcasing the developed design tool and not a polished fully detailed design.

5.1.1 The case

Aarhus Municipality is transforming its industrial harbour front close to the city centre to an urban recreational space with a series of projects. One of such projects is the construction of a very large multimedia house, named Urban Media Space, at the water front. On the southern dock near the new Multimedia house is to be build a high quality playground with themes for all ages that aim to aspire creative play and contribute to the blooming city life. The case will focus of designing concepts for a multipurpose controversial free form playground gridshell structure. Information on the exact location of the playground and the wishes for its design cannot be found in written form. The case is an unofficial project that will be put out for

Figure 114: Location of the new multimedia house in Aarhus. Image source: urbanmediaspace.dk

competition in the future. The information on the case is retrieved through conversations with architects and officials working with or for the municipality of Aarhus.
The lack of precise written information results in a high degree of freedom when designing the playground allowing very distinguishable conceptual designs. This freedom was another reason for selecting this case to showcase the wide variety of gridshell geometry that can be generated and optimized using the form finding tool.

5.2 Case analysis

In order to create design concepts it is necessary to analyze the case. Since the Multimedia house project is still under construction, all information must be retrieved from drawings and project material made available by the municipality of Aarhus. This material will analyzed to create a set of design criteria each concept must possess. The surrounding context and location of the playground structure will be analyzed first to set of specific geometric limitations for the project and determine the scale of the project. The context should also be used to help determine the visuals of the structure. The next step is to analyze the users of the playground structure. This will lead to a set of criteria for the type of functionality and characteristics the structure should posses and how large it should ideally be to accommodate the amount of users expected.

5.2.1 The context

Urban Media Space (UMS) is the largest construction project in the municipality of Aarhus. It is part of a series of projects to evolve the industrial harbour front into urban space. The project is completed in partnership with Realdania and Realdania Building.
The UMS building project is set to become a multifarious and active city space for sharing and obtaining knowledge, cultural experiences and lifelong learning. The building itself is a multimedia house with functions outlined and envisioned by the citizens of Aarhus in an involved collaborative design process with architects and total supervisors Schmidt Hammer Lassen in collaboration with Kirstine Jensen architects. The company Alectia A/S functions as engineering sub-consultants. The UMS project includes a modern library, citizen service centre, automatic parking, three new harbour spaces, better infrastructure and the recreation and clearing of the stream promenade running through Aarhus. The Multimedia house along with parking is to be completed at the end of 2014 and the harbour spaces will be finished at the end of 2015.
The building is designed to function as a hinge between the city spaces to the north and south. It will act as a pivotal central point in the city life. The project is key to bringing the city life all the way out to the harbour front where business, residential areas and recreational spaces melt together. The project seeks to make the waterfront spaces a new lively recreational, excursion and meeting point close to the water.
The building is signified by its large seven sided polygon roof structure. The facade is made of large glass and aluminium panels and the overall appearance of the building is sharp straight lines with rough unpainted concrete floors, walls and columns visualised in figure 116.
Two southern faced locations for the playground are in play at the current stage of the project. The two locations are marked in figure 115. The first location, marked by the red area, is on a large outdoor terrace floating 6 meters above ground level . The building sports a full 360 degree view at level 1 and up. By placing a structure on the terrace some of this view will be blocked making the location undesirable. The other area, marked with green, is on the ground level close to the large iceflake-like platforms that are part of the new harbour front. The green area lies just beneath a large south faced stairway. The largest location by far is the one marked in green and has an area of roughly 2500 m^2. This area will be selected of the two

Figure 116: The UMS building is composed of straight lines and sharp corners. Image source: urbanmediaspace.dk

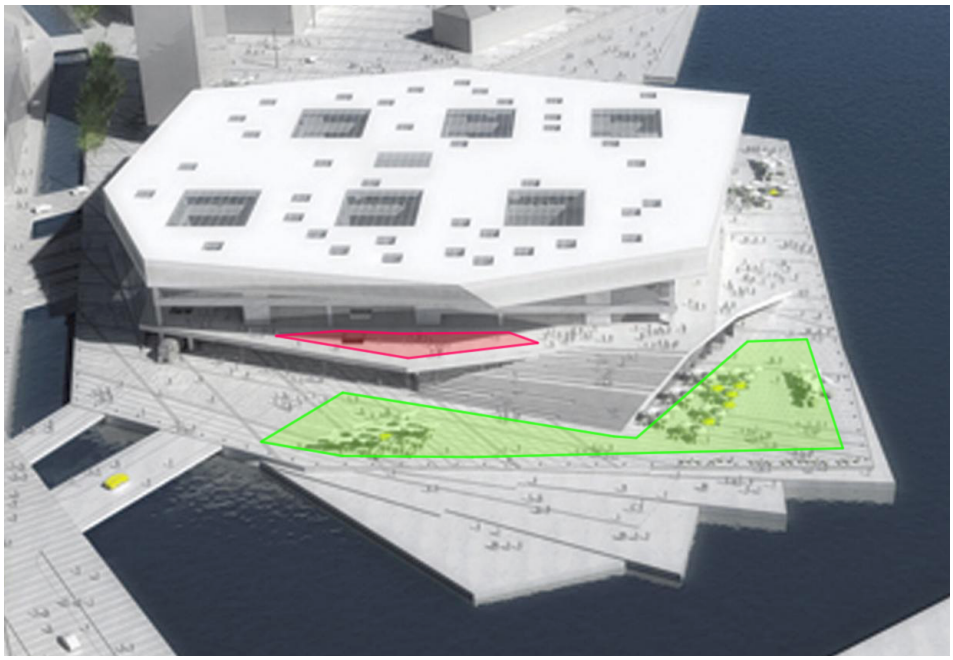


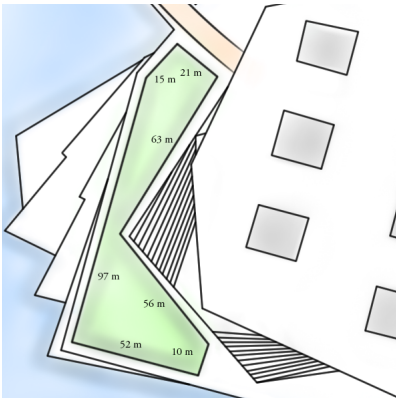
Figure 115: The location of the case is on the southern docks of the UMS platform. Image source: urbanmediaspace.dk

available. Only a fraction of this large area will be used for the playground as a lot of the space will be used for other activities. The iceflake-like platforms are reserved for sitting so the playground structure should not interfere too much with these areas.

At ground level the playground will lie just beneath a large southern faced stairway. The southern faced stairways will be used for seating by people enjoying the location. This means that the playground structure should avoid blocking the view from the stairway.

The interior rooms of the multimedia house facing towards the southern dock serve different purposes. Knowledge of these purposes are needed so that the placement of the playground structure is optimal in respect to not blocking the sea view of important rooms inside the building. Below the large stairway seen in figure 115 is a rental area that presumably will be rented to a cafe. The area closest to the city and the largest bridge is a transit area where blocking of the view is not an issue, but this area is undesirable since it will be close to traffic and noise. Blocking the view in front of the rental area should be avoided as much as possible.

Figure 117: Plan of the selected area. Only a part of the space will be used for the case.



5.2.2 Scale of the location

The area where the structure will be placed is large. In order to better understand the scale of the outdoor space a handball field (dimensions of 40x20 meters) has been laid over the area in figure 118. It is estimated that roughly half of that area should be used for the playground structure to accommodate the 45 people using it simultaneously estimated in 5.2.6.

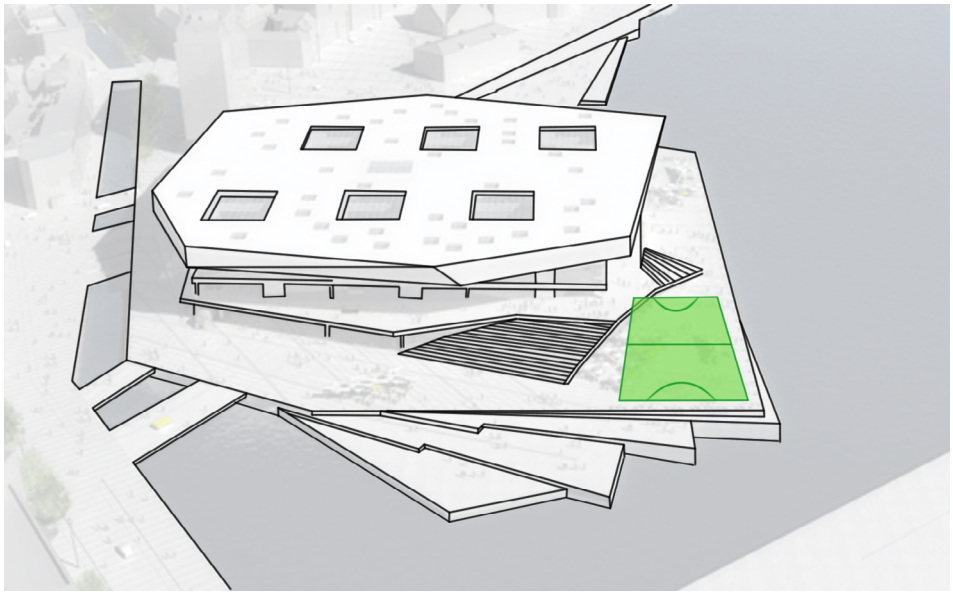


Figure 118: A handball field is placed on the exterior space to provide a comparison of a familiar space

5.2.3 Determining the functionality of the playground

Everywhere around the world adults are worried about the health of children and how it has taken a turn for the worse. Children around the world suffer from obesity due to a lifestyle of inactiveness and bad diet. This is a paradox when faced with the fact that children enjoy being active. WHO recommends that children are active at least one hour per day. Only around 25-50% of the population of 11-15 year-olds live up to that recommendation. Even worse it is for the European adults where less than 40% of the populations manages to be active less than half an hour a day. Boys are generally more active than girls.

More and more research show that physical activity and cognitive awareness is tied closely together. Studies point to the fact that physical active children have it easier when it comes to learning new things. Children learn a lot from the playground. They improve their motor function and learn how to socially interact with other children thus by teaching patience and conflict resolving skills thus preparing them for adulthood.

Play is born by curiosity. It is important to shape the playground to the users and it can be very difficult to design a playground that is fun and challenging for people of all ages. The playground must stimulate their creativity by allowing them to play without being instructed. Usage of the elements in the playground should not be dictated but up to the individual user to help stimulate creative and imaginary play. When elements does not have a pre-described utility the world of play opens up to a unlimited amount of opportunities. The playground seeks to break the traditional one-purpose structures found in traditional playgrounds. The result should be a playground with no predetermined functions such as traditional slides, climbing poles etc. The playground should just be an exciting gridshell structure people will want to explore. It should challenge the mind and body to create with a fun and controversial form.

5.2.4 Limitations to the free form structure

The playground structure will be a free form gridshell. The choice of material is important to ensure the right feel and appearance. The chosen material should focus on being children friendly and improve the safety of the structure. The study of appropriate materials and selection will not be considered in this case. Member material will be limited to steel in order to reduce the scope of the case. The steel structure can be quickly verified using the FEM component. The concepts will all be constructed using circular hollow sections for easy structural comparison and reasons mentioned in 5.2.8.
Most of the gridshell will be covered in a skin of panels of a transparent material. The panels and the gridshell will be strong enough for people to walk, run, jump and sit on. The underlying grid structure will be strategically exposed so people will be able to climb the structural grid at places where the height of the structure is below a certain threshold mentioned in 5.2.8.

5.2.5 The sculptural landmark

The free form gridshell should have a sculptural appearance in order to spark the curiosity of both children and adults. It should function as a meeting point and landmark for the area. This role will be easier to fulfil if the structure is visually interesting. The appearance should draw people near.
One of the major challenges is to find the right balance between context and contrast to the surroundings. The structure should aim to complement the architecture in UMS. Meetings with a local sculptor and the city architect of Arhus helped define the visual direction of the case.
Local artist and sculptor Jeppe Flummer suggested that the structure should be experienced as a sculpture from a distance but as architecture up close. He was interested in creating the right polarity between the playground structure and the surroundings. The polarity can be achieved by breaking the straight linearity found in the surroundings with soft organic shapes. Anish Kapoor, a famous sculptor, is an expert in creating this polarity. This is evident in his sculpture Cloud Gate located in Chicago, shown in figure 119. The sculptural appearance can be achieved by creating an organic free form.

Figure 119: Cloud gate in Chicago. Image source: wikipedia.com

City Architect, Stephen Willacy, of Aarhus suggested that the structure should be transparent to complement the existing surroundings. Transparent is here referred to as a way of being able to decipher the relationship between the playground structure and the surrounding environment. Transparent does not necessarily mean the usage of see-through materials.
The form finding tool can through quick renders visualize the context of the structure. This will be utilized when presenting and judging the aesthetics of the different concepts.

5.2.6 Users

The vision of the project is a high emphasis on a place where people can come to find and share knowledge. Visitors of UMS can be divided into three categories.

1. The children and adults that visit UMS come to play and learn.
2. People accessing the public library and numerous cafes inside UMS.
3. People commuting to other parts of the city via the new light rail track the runs under the UMS building.

A rough estimate of the amount of people simultaneously using the playground structure must be calculated to determine the required size of the playground structure. An average of 3500 visitors are expected to visit UMS each day. If each person on average stays an hour,

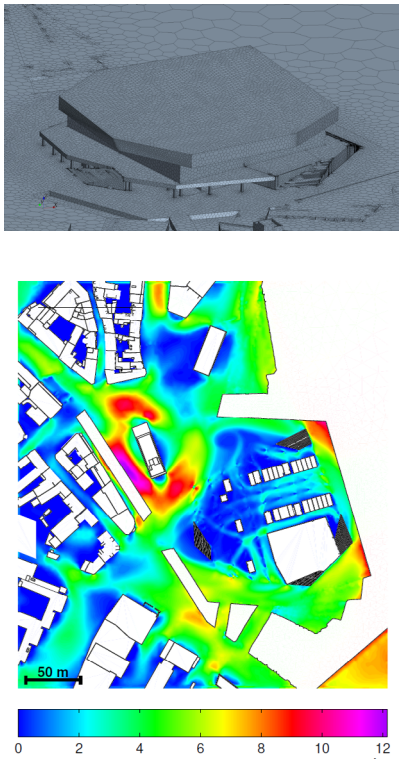


Figure 120: Top: The CFD model of UMS. Bottom: Comfort mapped on the local area. Areas with discomfort due to high wind speeds are marked with red and purple.

due to some people only commuting, and UMS being open and active from 10 a.m. to 20 p.m. that adds up to 350 people at any given time. A rough estimate could place half of these people outside and half of them inside. That is 175 people that has access to the playground structure. Since the playground is targeted for both children and adults all 175 people could potentially be using the playground simultaneously even though this would be unrealistic. It will be estimated that roughly half of the people outside will be around the playground at the same time totalling 90 people. Half of those are estimated to walk or climb around on the structure while the rest runs around or beneath it. That is roughly 45 children and adults. This number is used when structurally analysing the different concepts.

5.2.7 Micro climate

The area around the UMS is locally know as one of the most wind-swept areas in Aarhus. The university of Aalborg performed a wind screening test to computationally analyze the wind climate around UMS to see if the area would be uncomfortable in windy weather.

To the west of UMS is the city center of Aaarhus which will provide a sheltering effect for the most powerful winds. The UMS Building is facing the water and is likely to receive wind directly from north-east

The winds are calculated using CFD (computer fluid dynamics).The analysis simulates the entire region around UMS. The analysis calculates the wind speeds at head level, exactly 1.7 meters over terrain. The calculated wind speeds are compared with studies that shows when people will feel discomfort because of wind. The discomfort can be mapped for the entire region, figure 120. These maps show critical areas where high wind speeds can occur during windy days. The analysis showed that the most critical areas where at corners of the building. The cause for this is that wind flow around building corners will increase wind speed.

The overall result of the wind screening showed that the micro climate around ground level was good and no further improvements should be made to create shelter. The worst possible scenario for the area selected for the playground is when the dominating wind direction is due east, figure 121. Even still, the wind speeds were found to be just within the comfortable zone. The wind screening information can be used to find a form that helps shelter against the worst eastern winds that would otherwise occasionally effect the cafe/rental area under the stairway.

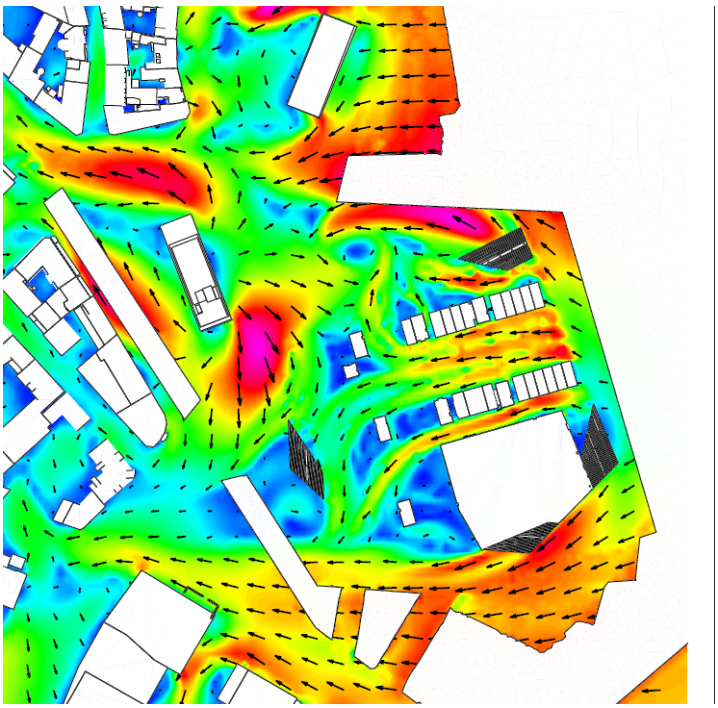


Figure 121: Wind simulation of wind from the east. This is the most critical scenario in respect to the playground location.

5.2.8 Playground safety and geometric boundaries

Designing and detailing playground is normally ruled by safety regulations. These regulations are set to avoid injuries. A few of the most important regulations will be included when conceptualizing designs for the playground structure. As mentioned earlier this case is to showcase the geometric functionality of the form finding tool. This means that the safety regulations taking into consideration are directly translatable into geometric rules the construction must follow. The regulations are found in a safety handbook for playgrounds made by the Consumer Product Safety Commission of the United States. The regulations are not cited, as they may as well have been fictive. They only serve the purpose of introducing geometric restraints for the case.

- Entrapment of the head can be avoided if the structural openings are less than 9 centimeters and above 25 centimeters. This directly translates into a minimal grid-size of the gridshell structure. This criteria should be easy to avoid as a mesh with a member length of even 25 centimeters would be extremely small and very costly to fabricate.
- The maximum climbing height of the structure should be 2.2 meters. This means that highest point where it is possible to fall off should at most be 2.2 meters. Underneath climbing areas the ground should be covered in shock absorbing material to lessen the impact of falling. The 2.2 meter criteria does not mean the structure is not allowed to be higher than 2.2 meters as long as it is not possible to fall off.
- Structural members used for climbing should have diameters that allow for good grip and foot support. Circular sections are preferred. Member height should have a diameter of maximum 8 centimeters and no less than 5 centimeters.

5.2.9 Panels

As mentioned in 5.2.4, glass panels will be chosen to provide the transparent look desired. Anti Slip safety glass is needed to avoid risk of falling or slipped when the panels are wet. Several glass suppliers produce glass capable of resting applied loads from people walking. The glass panels are rippled with patterns that make for sure footing even for slippery wet exterior areas with steeper inclines, figure 122. This case will not concern the dimensioning of the glass panels. It is estimated that a maximum panel surface area of 1.0 m^2 is acceptable in order to provide sufficient load bearing capabilities. This size will limit the grid-cell size of the grid-shells.

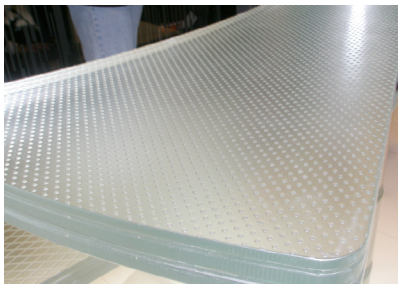


Figure 122: Safety anti-slip glass.

5.2.10 Conceptual design rule-set for the playground structure

The case analysis can be summarized in a list of design criteria that should considered and included when designing the structure concepts.

Geometric boundaries:

- The structure should not have openings where body parts can be trapped. This can be achieved by having a grid-diameter of at least 25 centimeters. The grid panels should have a surface area of maximum 1.0 m^2.
- Areas on the structure where it is possible to fall of should be no higher than 2.2 meters.
- Playground structure should avoid blocking the view from the stairway.
- Playground structure should be far enough away from the rental area under the stairway to allow for outdoor seating of a possible cafe.
- There should be space enough for a fire truck to get around the structure in case of fire.

Functional design:

- The design of the playground should be kept simple to encourage creative imaginative play.
- The playground structure should not have predetermined functions. (By predetermined functions the following examples help clarify; slides, maze, swings, skater park, etc.)
- The playground should help improve the micro-climate on the southern dock by sheltering for winds from the east.

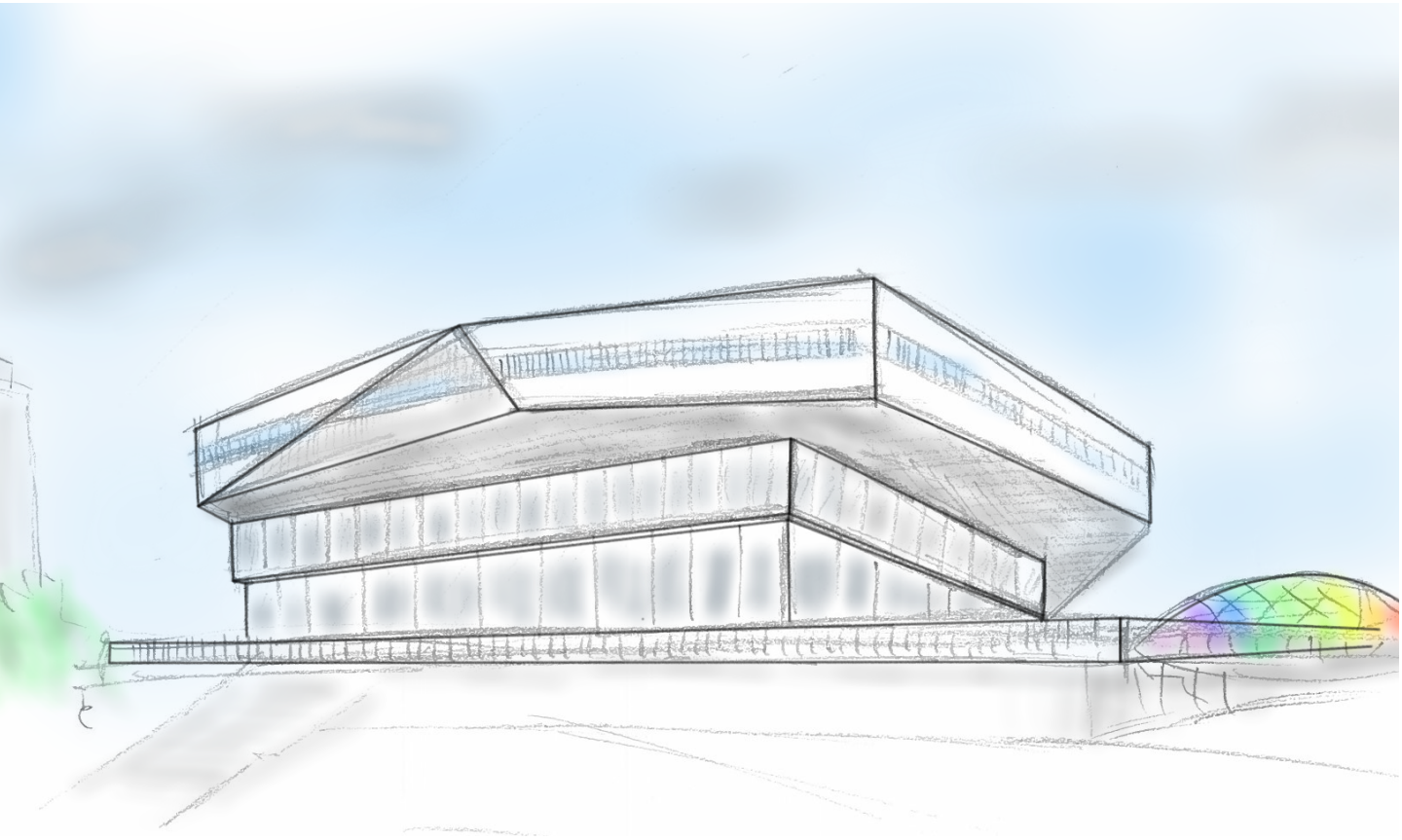
Appearance:

- The structure must in some way be transparent and complement the context with the existing building.
- The form of the structure should also create contrast to the existing building by being organic.
- The structure should further develop the landscape on the southern dock.
- The structure must have a sculptural appearance as to better serve the purpose of also being a meeting point and landmark for the souther dock.

Structural design:

- The playground structure must be a gridshell structure developed with the form finding tool.
- The gridshell structure must be strong enough to be able to carry the load of people walking or climbing on the structure.
- The members of the structure will be made of steel.
- The panels of the structure will be made of safety anti-slip glass.
- The member sections will be circular hollow sections with an 8 centimeter diameter.

Figure 123: Sketch showing concept 2 from the city side.

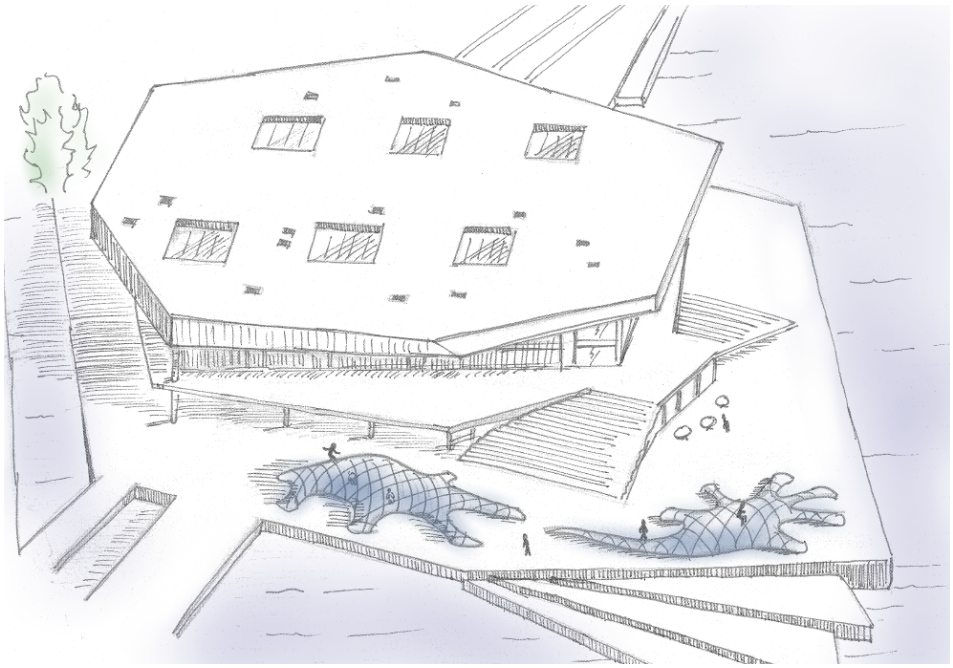


5.3 Sketching the concepts

The different criteria was conceptualized into three different designs. The concepts were chosen to represent, visually and functionally different types of free form geometry that can be made with the form finding tool.

5.3.1 First concept

The first concept, shown in figure 124, is trying to imitate amoebe-like organisms. The organism will be spread around the available location. The concept aims to show the versatility of the form finding tool by creating very complex organic shapes.
The structure will have openings so that people can walk inside.



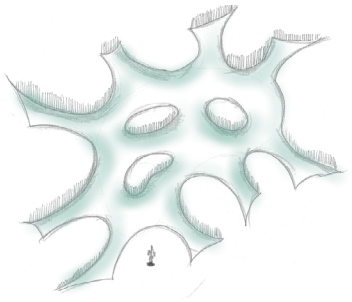


Figure 124: The amoebe structure will be placed differently than shown on the sketch. It will be placed to shelter for the wind from east.

5.3.2 Second concept

The second concept, shown in figure 123 and 125, will possess a blob-like form. The form of the structure is the simplest of the three concepts. The ground beneath the structure will be excavated to create more room beneath the grid-shell. This also allows the grid-shell to be lower while people still being able to walk under it. This principle could be applied to the three other concepts. The glass panels could be colorized to create a sort of glass-cave with spectacular lightning similar to the effects in the rainbow at Aros Art Museum.

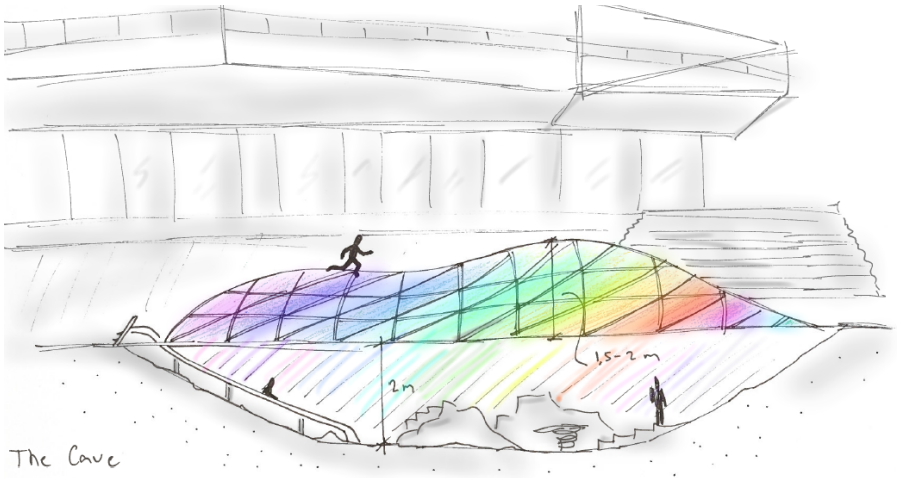


Figure 125: The amoebe structure will be placed differently than shown on the sketch. It will be placed to shelter for the wind from east.

5.3.3 Third concept

The third concept will try to compliment the sorrounding area by "climbing" the dock platforms. This concept will illustrate how the form finding tool can be used to generate geometry from a flat grid with carefully selected support points. This method allows the user to generate a geometry that is integrated with the existing buildings.

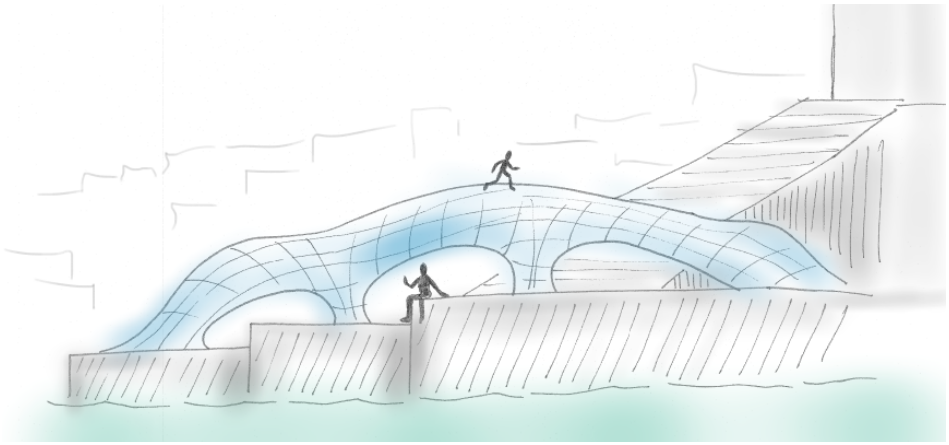


Figure 126: The third concept will use integrated columns.

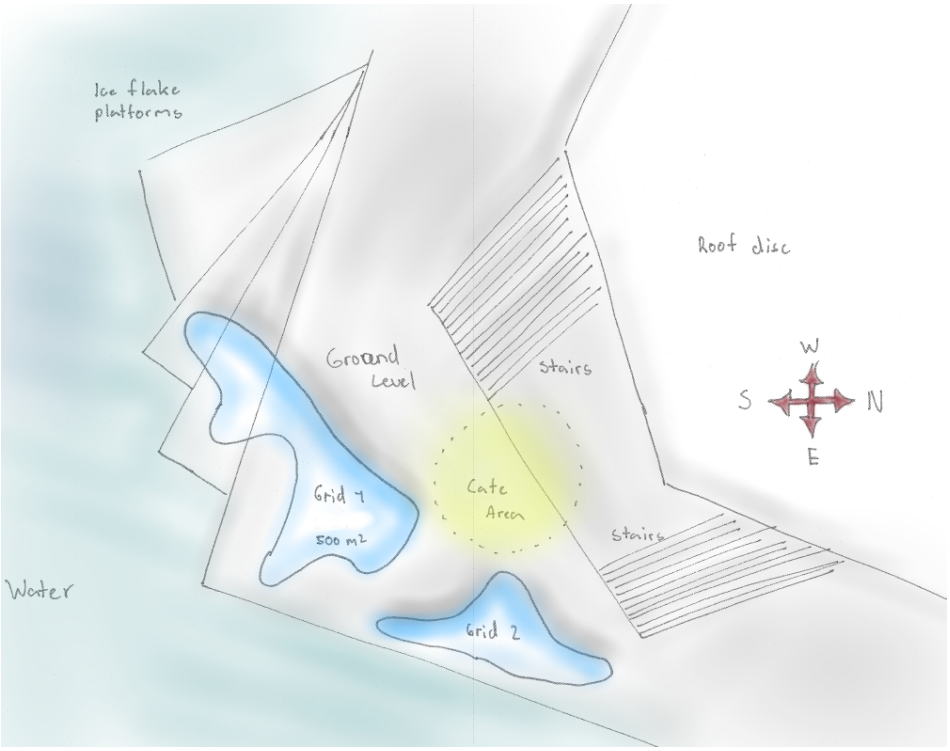


Figure 127: The location of the third concept exceeds the available location. This was chosen to experiment with a grid that is not created on a single planar surface but several platforms.

5.4 Generating the geometry of the concepts

The concepts will be transformed into three-dimensional grid geometry using the form finding tool. Concept 2 will be modelled as a free-form NURBS surface as explained in chapter 4.1. Concept 2 and 3 will be modelled as a flat surfaces and transformed to grids to simulate the traditional physical form finding technique. It is also to avoid having to hand model an amoebe-like surface. The form of the relaxed grid is related to the spring stiffness and member support. Changing these input parameters can help create the desired shape.
Once the surfaces have been made, the grids will be generated. The next step after grid generation is the geometric optimization, using the dynamic relaxation algorithm. Once the gridshells have been optimized they will be structurally analysed. After this the case ends with an evaluation of the results. Normally the process starts over trying to implement newly realisations to iteratively improve the design concepts or the developed concepts could spark the birth of new ideas.

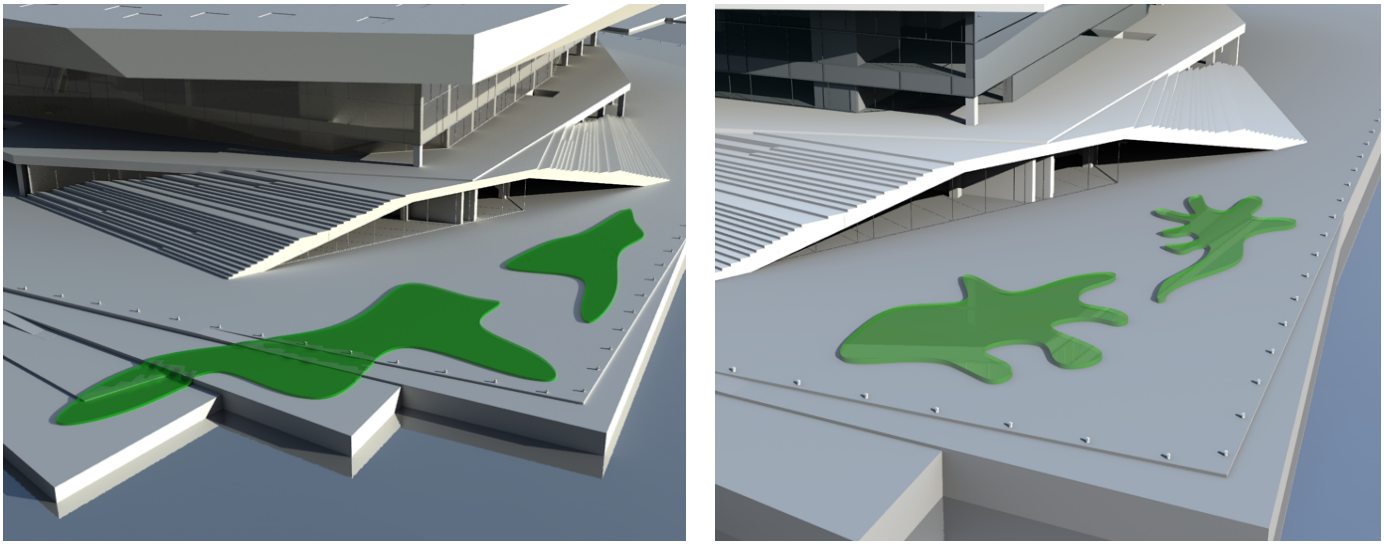


Figure 128: The flat surfaces of concept 1 and 3.

5.4.1 Creating the surfaces

After having decided the overall concepts through the case analysis and sketching, the surfaces are ready to be modelled. There are several ways to go about modelling the surfaces. The most straightforward way is to draw the edge NURBS curves of the surfaces and use commands to transform the edge to a surface. The edge can be drawn on top of scanned image of a sketch to get the exact shape desired. The surfaces can also be represented parametrically so they can be altered throughout the process. Creating the parametric definition of the surface takes a little more time than just drawing it, but once it is done, it can save a lot of time later if alterations needs to be made. The surfaces for this case were quickly modelled to represent the overall shape found through sketching.

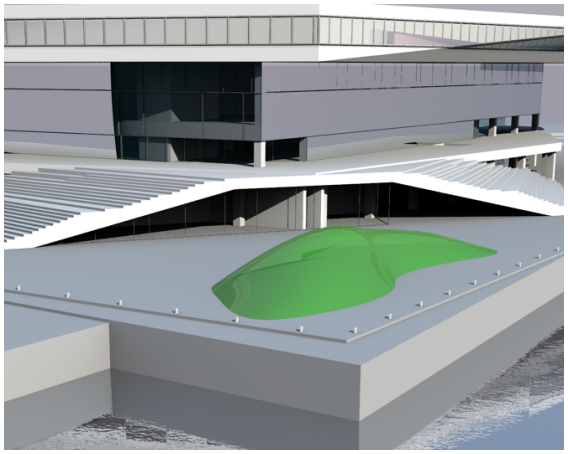


Figure 129: Free form NURBS surface of concept 2.

5.4.2 Tessellation of surfaces

The next step is to tessellate the surfaces using one of the algorithms presented in chapter 4.2. The equilateral triangle grid algorithm was chosen for all the surfaces. This was chosen in order to create grids that could be compared. The member length of the different structural grids will vary very little. The algorithm is explained in chapter 4.2. On figure 130 is shown an example of the creation of the grid used in concept 3. The average member length is 0.7 m.

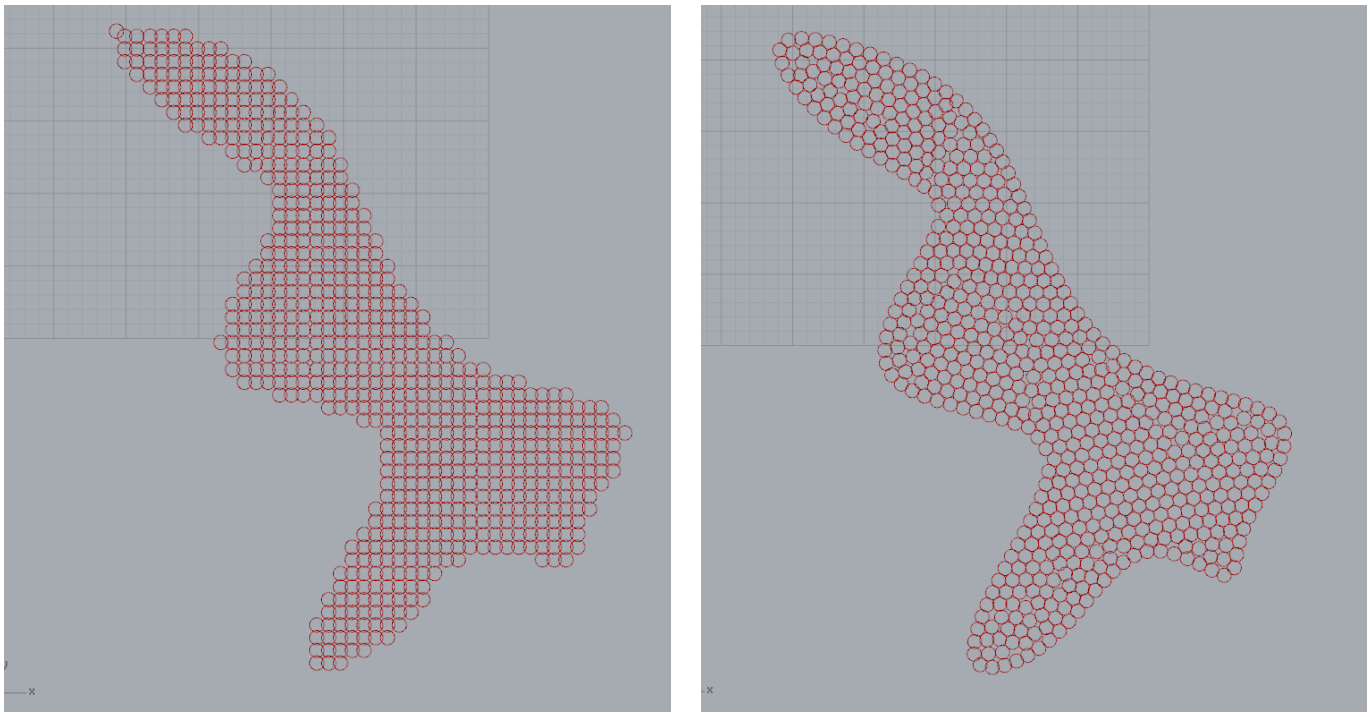


Figure 130: The grid generating algorithm is creates a grid by evenly distributing circles on a surface using springs connected between points.

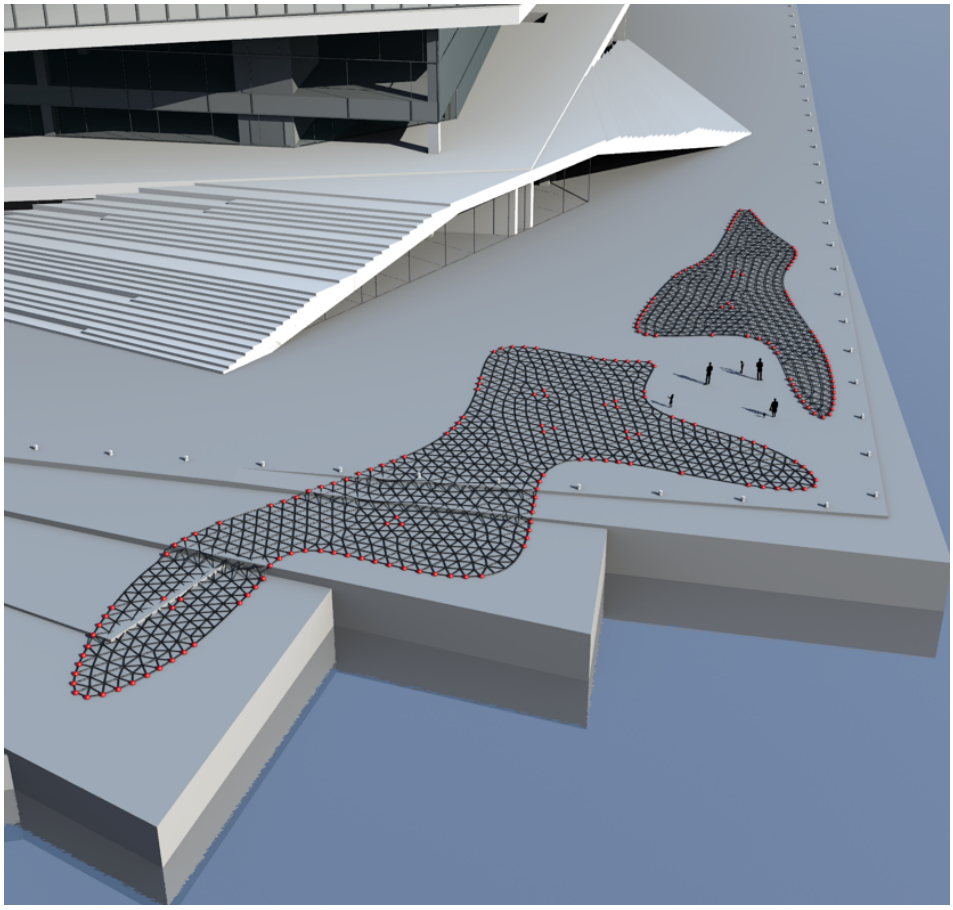


Figure 131: The resulting grid used for dynamic relaxation. The red dots represent where the grid will be fixed.

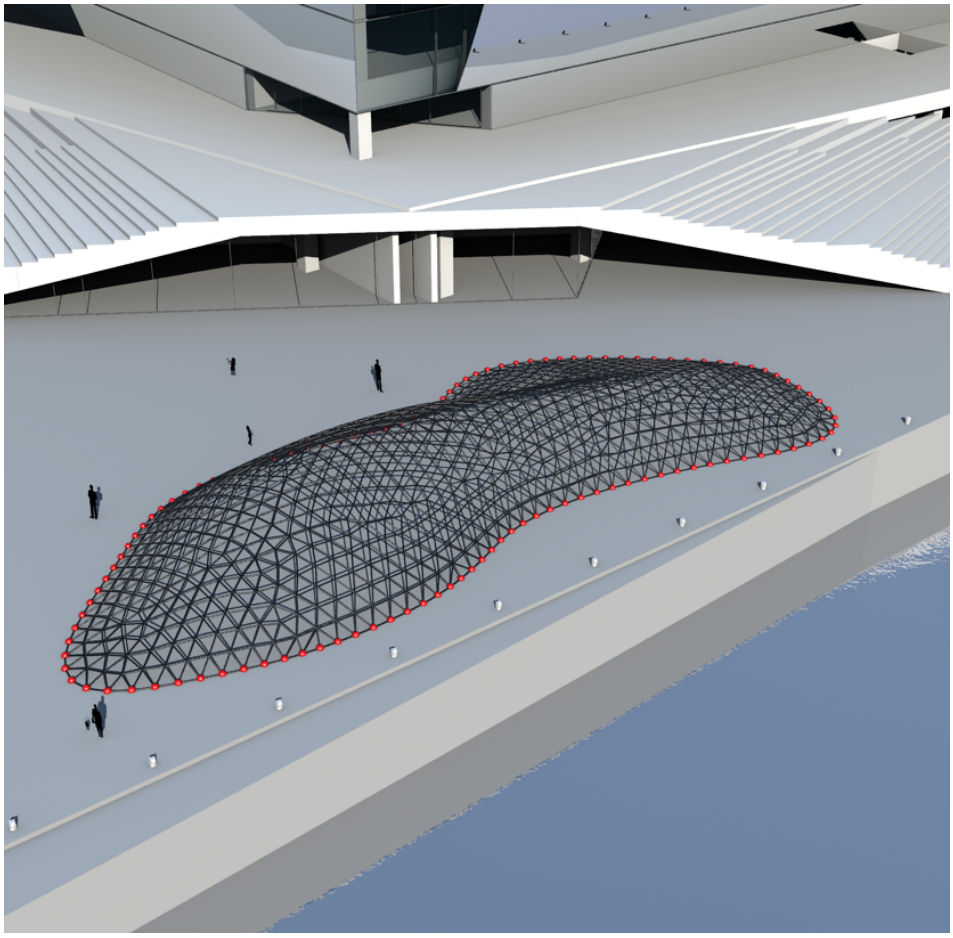


Figure 132: Triangular grid for concept 2. The red dots mark where the grid is fixed. By leaving out fixed points along the edge natural openings will occur during the dynamic relaxation process.

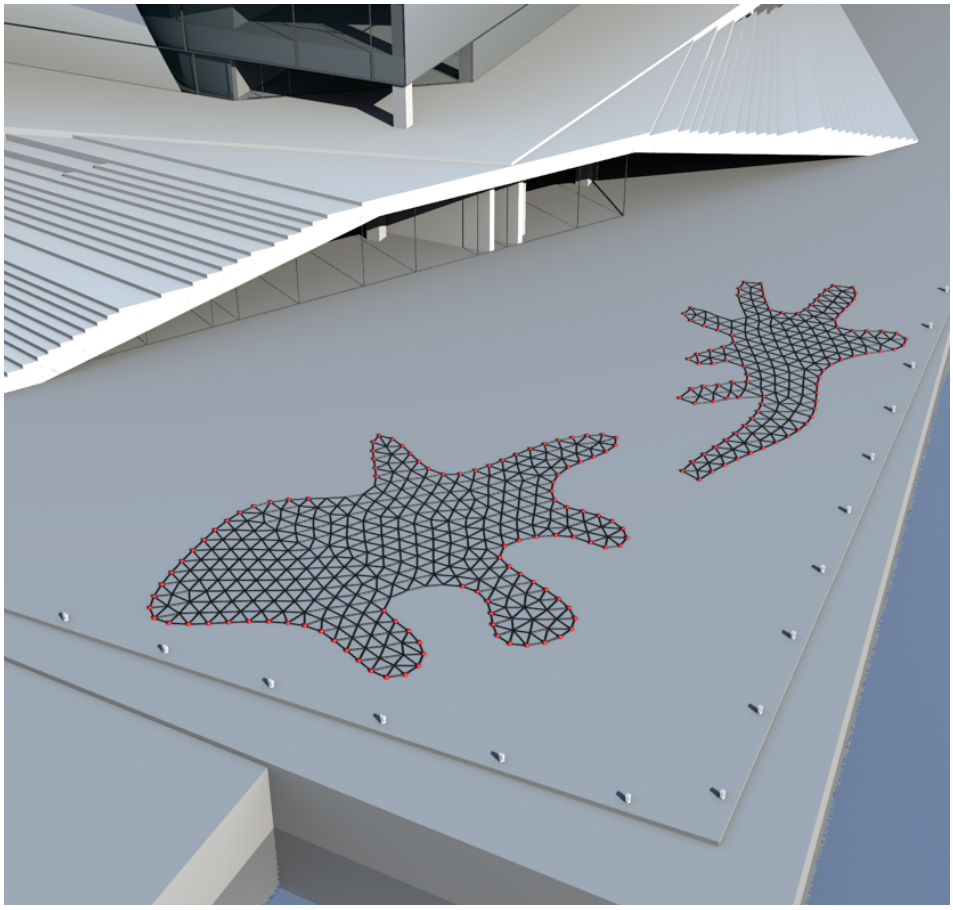
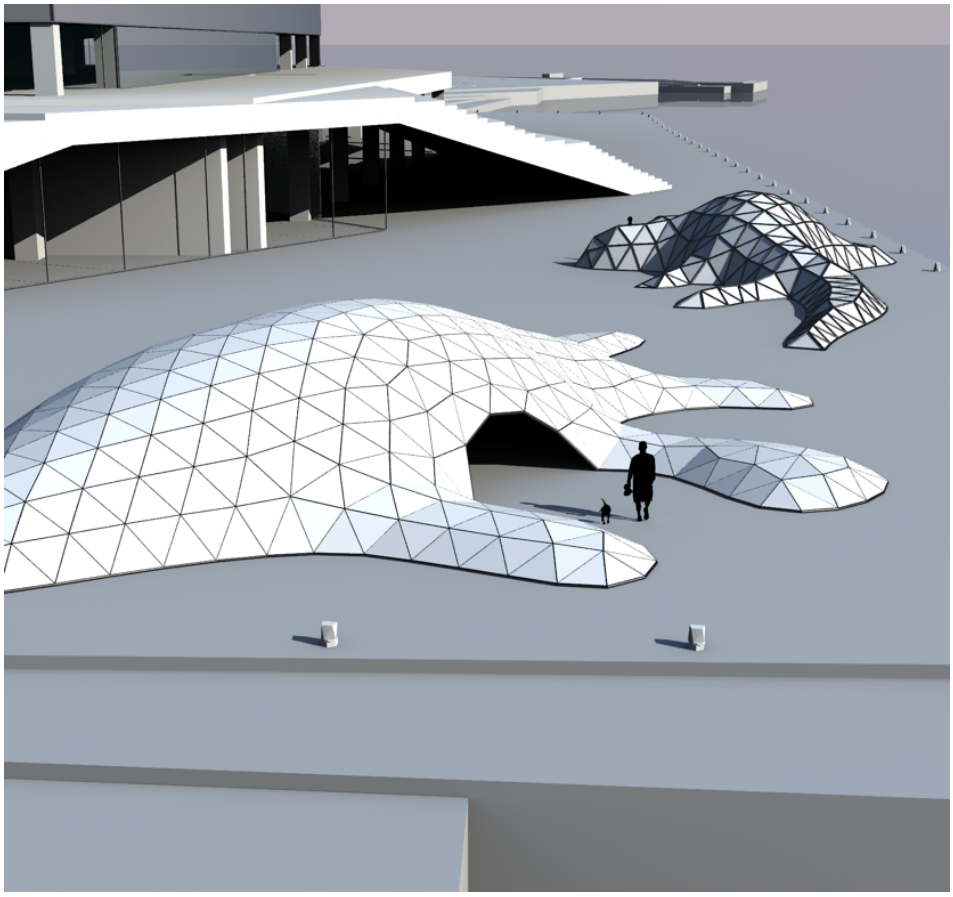


Figure 133: Triangular input grid for concept 1. Fixed members are marked with red dots.

Figure 134: Concept 1 after geometric optimization. Openings have been made by allowing certain edge member to move.



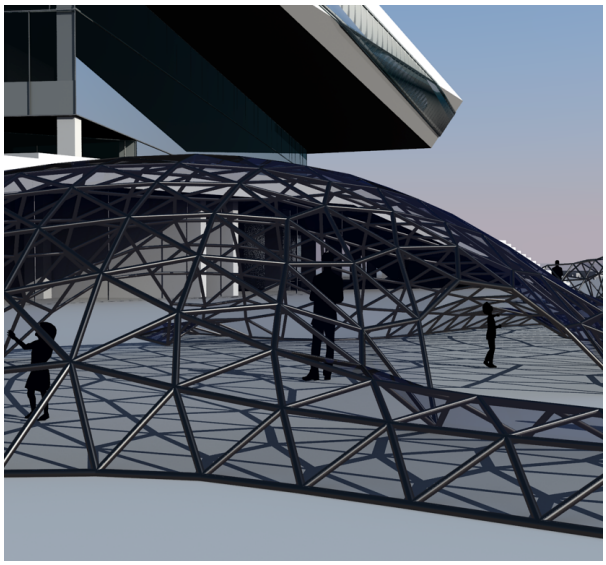
5.4.3 Dynamic relaxation

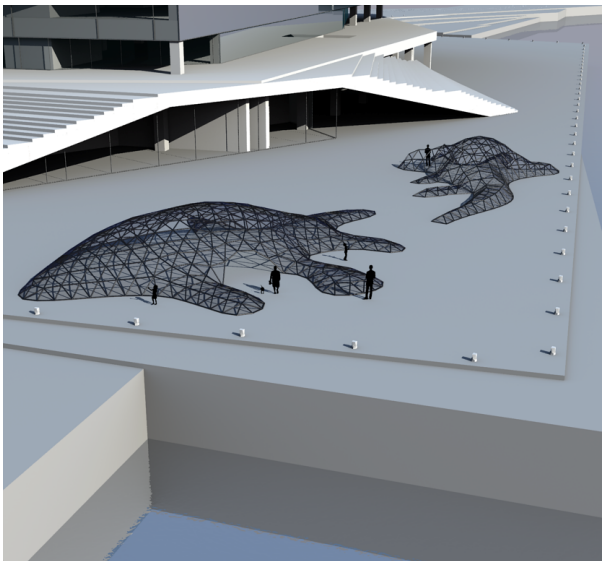
The grids found in the previous passage was relaxed using the dynamic relaxation component developed in chapter 4.3. The input parameters where tweaked to get the desired structural height and appearance.

The panels are shown in the figures to better visualise the form of the structure. Different materials can be applied visually when rendering images in Rhinoceros. A simple white texture was chosen as it amplifies the shape.

Descriptions of the results from the relaxation process can be found in the figure descriptions on the following pages.

Figure 135: Concept 1 with transparent panels.





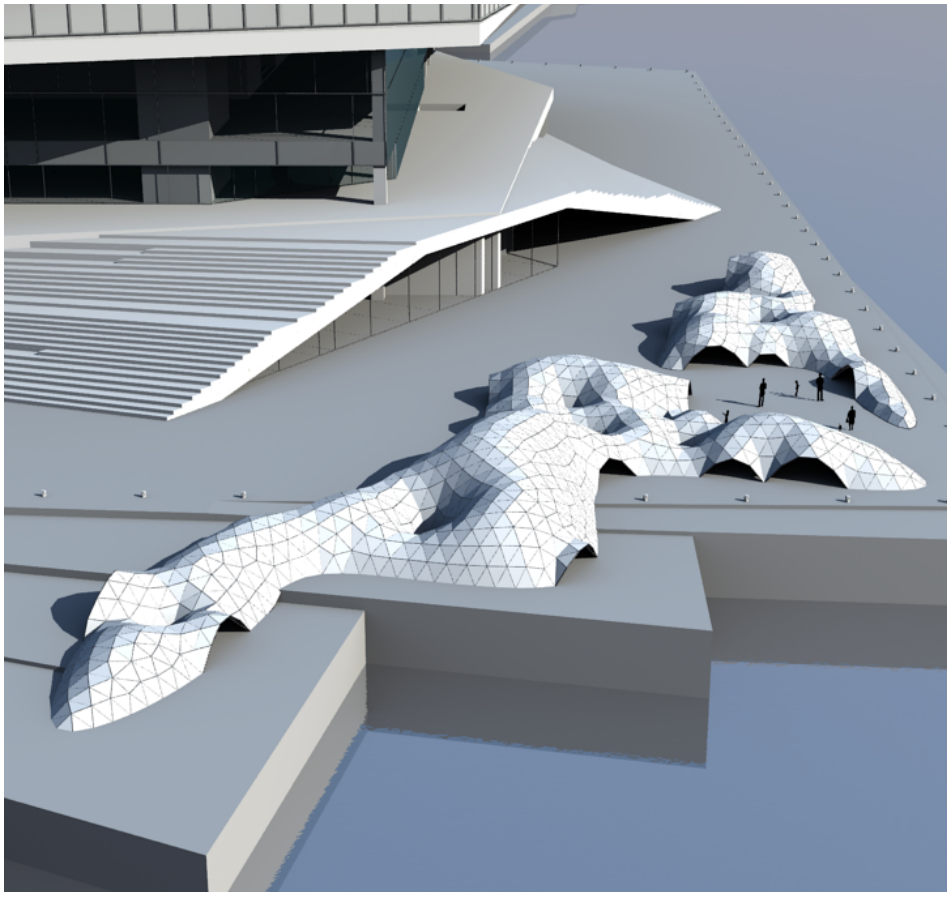


Figure 136: Concept 3 was modelled with strategically placed fixed members. This allows the structure to create a very interesting area to explore.

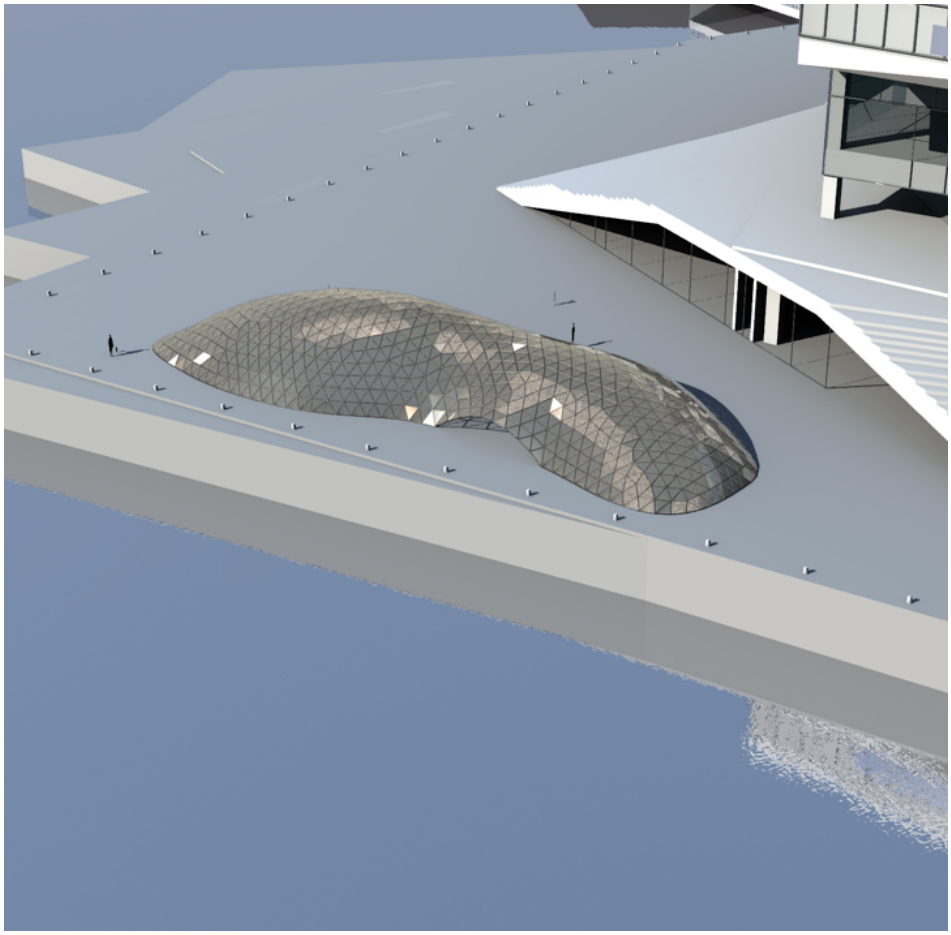


Figure 137: Concept 2 is simple. The openings are smaller than those found in concept 1 and 3. The idea is to excavate the area under the structure to create a large room height. The excavation is not shown on the figure.

5.5 Structural analysis

The different conceptual designs will be structurally analyzed and compared. The comparison will be based on results from the FEM component. Comparisons will include structural deformations, utilization of structural members and natural frequency. The analysis is meant to showcase how the tool can provide a rough estimate to determine the best structural design without spending allot of time. The structural analysis will therefore only include a few load cases described in 5.5.2.
Grid-shells are uniquely suited for resisting uniformly applied loads making them best served as long-span lightweight roofs. This case sees the structures having to resist concentrated loads from people climbing and walking on the structure which is far from an optimal scenario. Concentrated loads will cause large deflections and disturb the membrane behaviour. The dynamic algorithm is best used to optimize a structure to resist uniformly applied loads. However, the algorithm will still provide a better solution than a non-optimized grid-shell, even if it is a subject to concentrated loads. The concentrated loads will likely lead to larger profile requirements.

5.5.1 Member verification

As mentioned in chapter 4.4 the utilization of structural members is calculated by the FEM-component. The method used is from DS/EN 1993-1-1 2007 6.3.3 for uniform members in bending and axial compression. The method includes buckling, so member stability is being checked. The three different concepts will be applied the same loadings and member cross-section to directly compare the different results.

5.5.2 Loads

Dead loads will be estimated and a governing loads case will be selected from a few possible load combinations. The selected load case will be utilized on all conceptual designs and will serve as the base for their structural comparison. It is easy to create many detailed load cases in Karamba, but for this case it will be sufficient to estimate a worst-case-scenario load combination. As mentioned earlier, the worst possible scenarios is when the structure is subject to unevenly distributed loading such as concentrated loads or unequal surface loading.
A simple test is run in order to quickly find what should generally simulate the worst possible load combination for a grid-shell structure. A simple rectangular grid is made with the tool and a FEM-model is made. The grid spans 20x20 meter and average member length is 1 meter. The gridshell is supported with simple supports and the structural members are given hollow circular cross-sections (80x4 millimeter). The weight of the glas panels covering the grid including aluminium profiles for attachment is estimated at 0.5 kN/m^2.

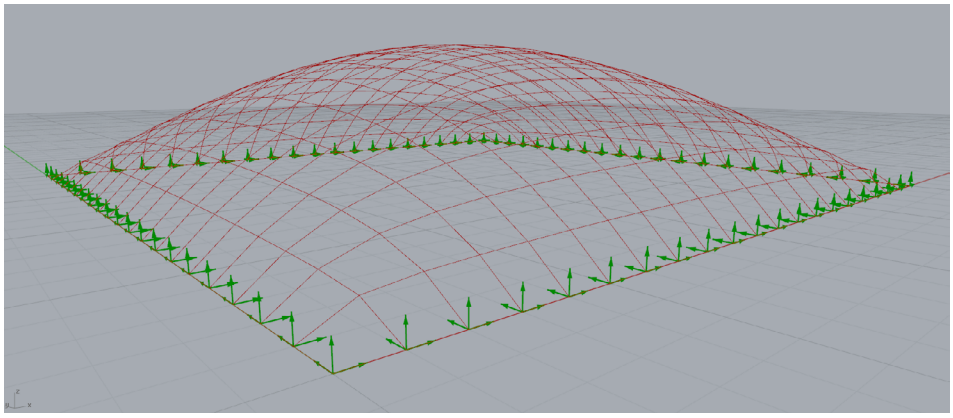


Figure 138: The simple grid-shell model is the same scale of the shells developed for the case. The shells is given simple supports.

Three loads scenarios will be checked. The first scenario will be the maximum gravitational vertical load found by a combination of structural self weight of profiles and glass panels combined with a dominating snow load. The snow loading will be estimated as 1.5 kN/m^2. The results are summarized in table 8.

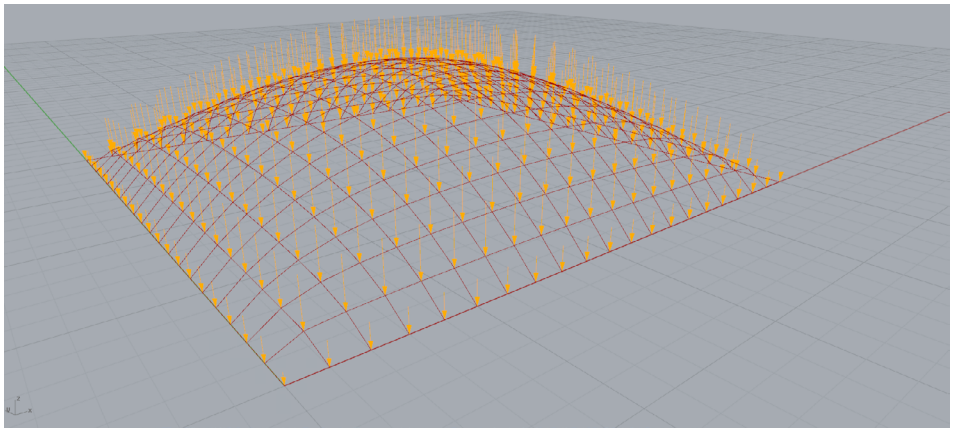


Figure 139: The first loads case is the maximum vertical evenly distributed load.

The next load scenario is unevenly distributed snow load. The loading will be estimated as 2 kN/m^2 to simulate the scenario of snow accumulating on one side of the structure. This load will be combined with structural self weight. This should result in much larger deformations due to increase in bending stresses. The results can be seen in table 8.

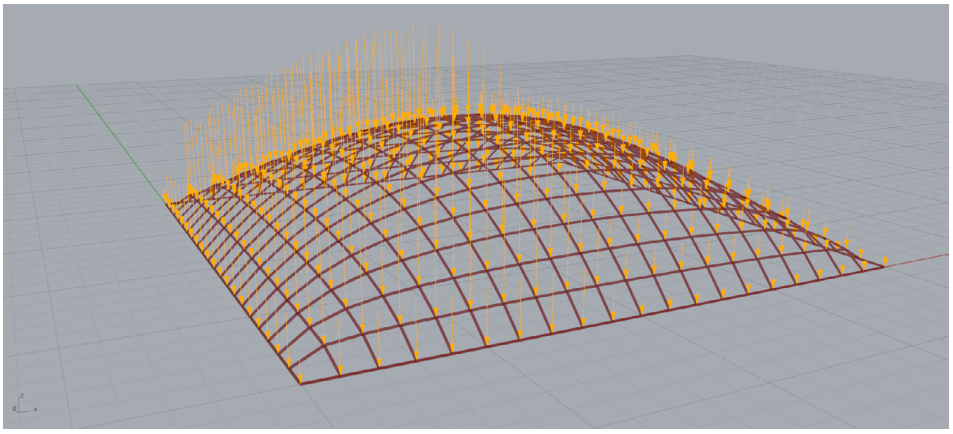
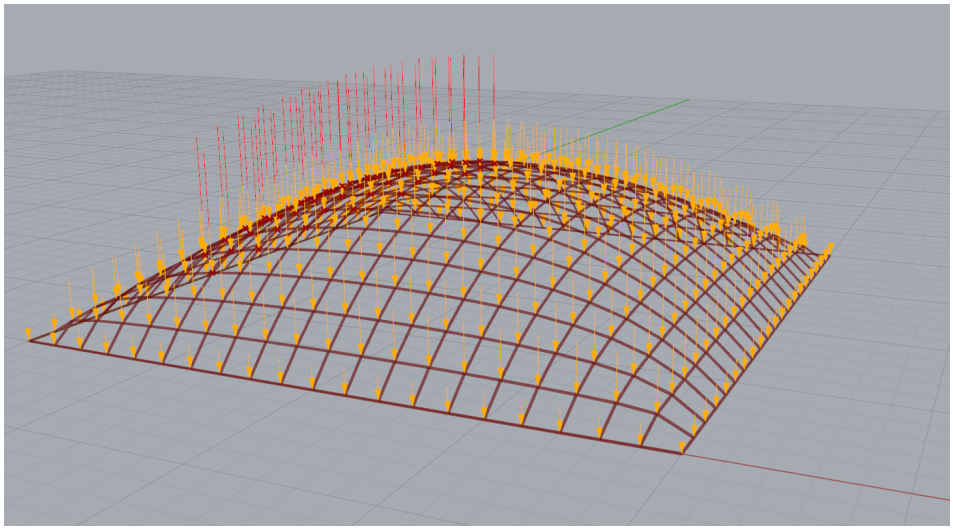


Figure 140: Unevenly distributed snow load simulating snow accumulating on one side of the structure.

The third load case is a combination of unevenly placed concentrated loads to simulate people standing clumped on one side of the structure. Each person weighs 1.5 kN. Several positions and combinations of concentrated loads was tried. This one resulted in the largest deflections and member utilizations. The results can be found in table 8.



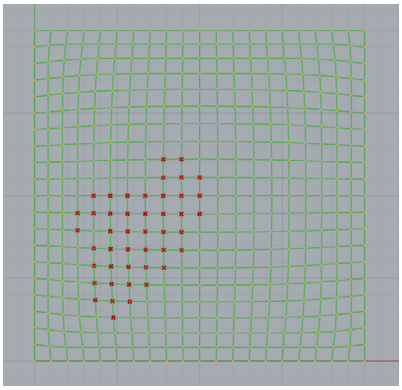


Figure 141: The dead load of 45 people placed unequally on one side of the structure.

Table 8: Results of the different load scenarios

	Max. member utilization [%]	Max. nodal displacement [mm]	Max. section bending force [kNm]
Maximum gravitational load	15.0	4.9	0.1
Unequal distributed snow loading	84.8	44.5	2.35
Unequal concentrated loads	62.0	34.2	1.63

As expected the uniform load is not the cause for large deflections or member utilization. Loads will be resisted entirely through in-plane axial compression forces. The structure has been specially tailored through the dynamic relaxation algorithm to resist this type of loads.
When loading is unevenly distributed the structure begins resisting applied loads with tensile and bending forces. This results in a much higher member utilization and deformation.
The unevenly distributed snow load is the worst scenario for the grid-shell structure even surpassing the concentrated loads. This is due to the fact that the shell used in the example has a very long span. This is not the case for the developed concepts. For the concepts in the case the loadcase with concentrated loads will be the worst scenario.

5.5.3 Live loads

Dynamic loads will not be covered in this case. Dynamic loads could however posses a danger to the grid-shell structure. If a dynamic load resonates with the natural frequency of the structure large deflections will occur. This could be caused by a number of people jumping synchronously. The natural frequency of the structure can be determined with the FEM component. Dynamic wind loads could potentially be calculated using computer fluid dynamics. All the necessary information is available seeing how the structure is 3D modelled. This could result in very precise loads resembling a wind tunnel study.

5.5.4 Analysing the three concepts

The three relaxed grid-models will be applied unevenly distributed concentrated loads like described in 5.5.2. The three FEM models will use the same circular hollow cross-section as in the previous example. The load sizes and support conditions also remain the same. The results are shown in table 9.

Table 9: Concentrated loads on the concept gridshells

	Max. member utilization [%]	Max. nodal displacement [mm]	Natural Frequency [Hz]
Concept 1	2.8	1.5	336
Concept 2	12	2,1	13.1
Concept 3	8.5	2.0	21.5

It is pretty obvious that the member sections are immensely over dimensioned for the applied concentrated load case. The structural analysis should be redone using lesser sections. The structural analysis stops here, as it is only to showcase that it is possible to perform the calculations immediately.

5.6 Building the gridshells

Steel is chosen for the concepts because it provides high strength and stiffness needed at a low material cost.
Scaffolding must be raised to act as temporary support for the gridshell until it is completed.
Structural elements can be pre-fabricated and combined into ladders of connected structural members. This is to help ensure a minimization of on-site work which is especially beneficial with structures that require a large amount of welding. Workshop welding can help improve time spent assembling structural elements and make sure that all welds are made to meet the requirements of the structure. Careful adjustments can be ensured so that the geometry of every structural element is as close to perfect as possible to avoid unwanted interior forces due to imperfections. These ladders can then be transported to the building site on trucks and connected to other ladders.

5.6.1 Structural joints and panel attachment

The structural joints are conceptualized as cylinders. The cylinder will have planar surfaces specially cut to the incoming member angle. This is very similar to systems shown in chapter 4.7. The members must be cut at an angle to acommodate the surface curvature .

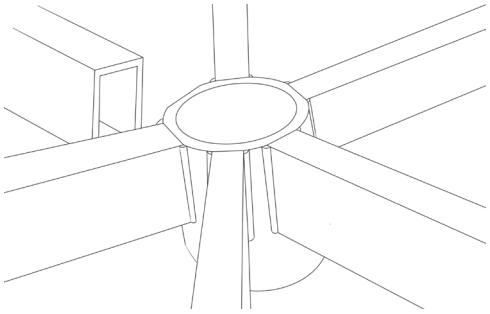


Figure 142: The nodal connection. Members are welded to the steel cylinder.

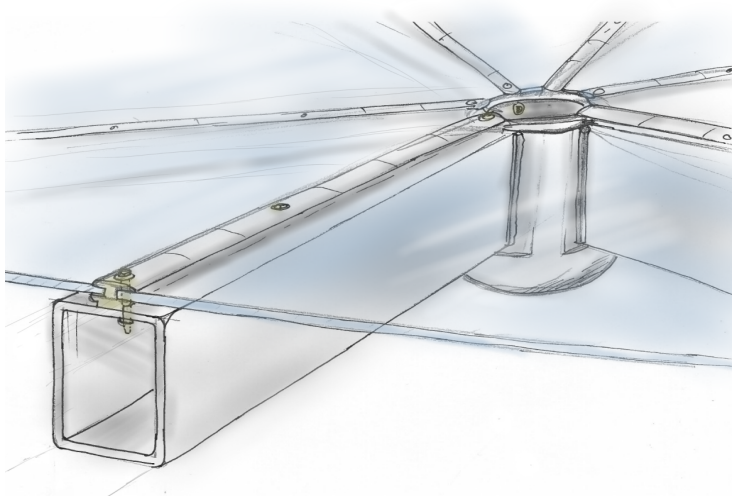


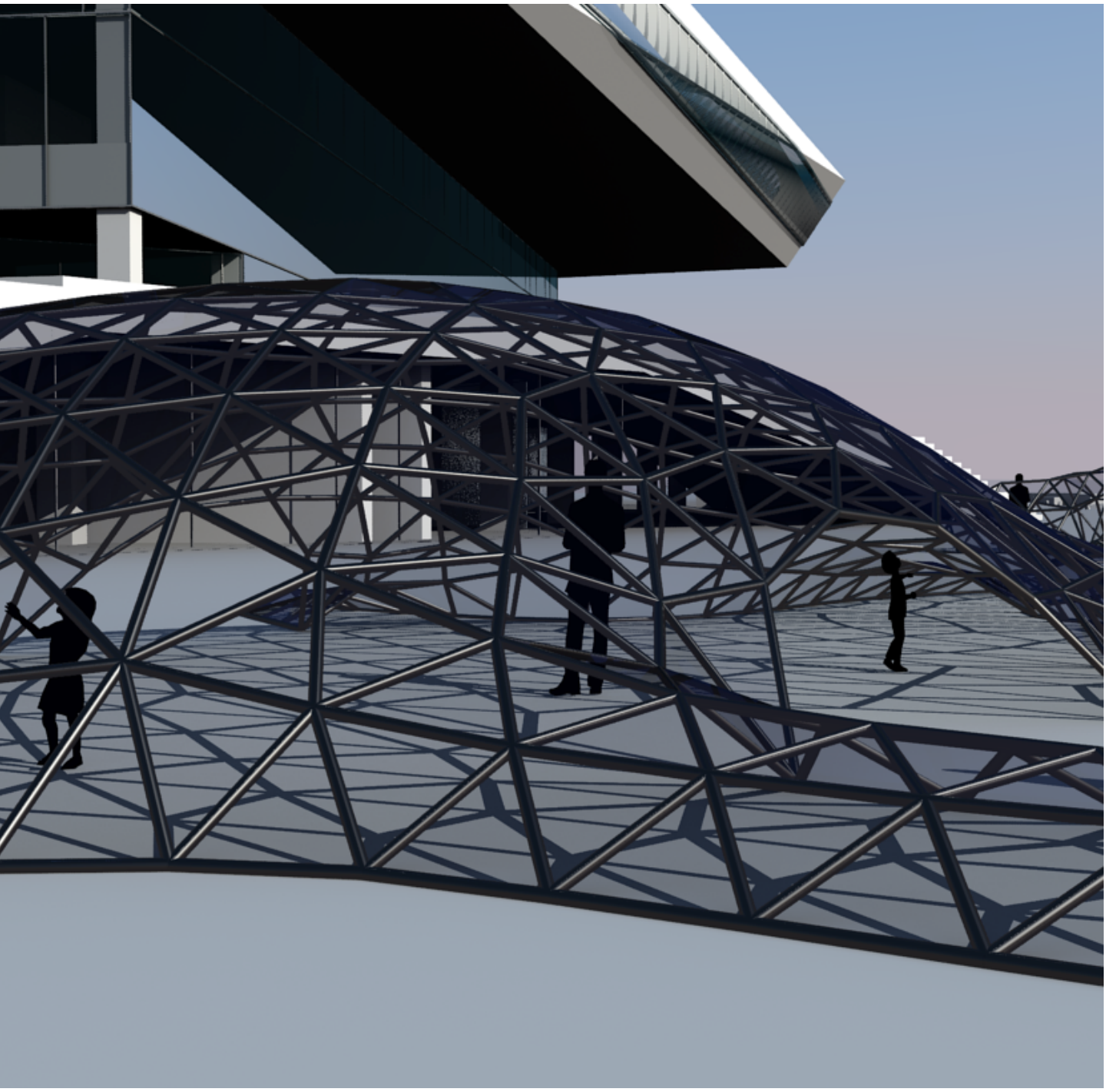
Figure 143: Panels are fixed with clams fixed to a aluminium profile. The profile is bolted into the structural members. Silicone gaskets help secure the glass by keeping it fixed. It also makes the structure waterproof.

5.6.2 Supports

Structural supports should aim to allow the structure membrane behaviour. This can be achieved by letting the support be perpendicular to structural members and introducing a form of rollers in the support.

5.7 Case conclusion

The case was used to illustrate the complex structurally viable geometry that can be generated with the form finding tool. The geometry generated served the purpose of creating something out of the ordinary. All the geometry created in the three concepts was generated and analysed over the course of one evening. It is possible to generate a vast amount of possible designs or many alterations of the same design.
The structural analysis was made very brief to only showcase some of the possibilites available in the form finding tool.



6. Conclusions and evaluation

Generating and optimizing the geometry of complex free form grid structures is traditionally not an easy task. In order to manually perform this geometric optimization a large amount of slow iterative steps involving the creation of physical modelling are needed. This results in a time consuming form finding process. The form finding process can be improved by harnessing the iterative computational power found within normal household computers. The gridshell optimization problem can be transformed into a discrete particle-spring-system that can be numerically solved by the computer.
The goal of this thesis was to create a free form gridshell form finding tool. The tool should be able to generate and optimize the geometry of a gridshell. The geometry should be dynamically structurally validated using a commercial FEM-component. The geometric optimization has been the focus of the developed tool. Optimization was achieved by programming a dynamic relaxation algorithm in the C# programming language. The algorithm is capable of optimizing a gridshell structure in real time by relaxation the geometry into an equilibrium shape. During the optimization the designer can change input parameters that translate into new geometry and immediate structural calculations. The process allows the designer to quickly explore and structurally verify a vast number of possible solutions. Alterations in design can be made up to the very point when fabrication commences. This is due to the fact that everything in the geometric model is linked together parametrically. The plans needed for fabrication are created automatically allowing the designer to create last minute changes without delaying the project.
The goal has been achieved and the tool developed in the parametric platform of Grasshopper posses three major abilities.

1. Generative grid algorithms that tessellate a free form surface into a discrete grid of lines used for the dynamic relaxation and FEM components.
2. A dynamic relaxation component coded in the C# programming language. The algorithm transforms the grid of lines into a particle-spring-system. The differential equations of the system are solved using numerical Verlet Velocity integration. The output of the algorithm is a grid of relaxed lines.
3. The grid of relaxed lines is linked to, Karamba, a finite element program.

The results of the form finding tool look promising. The tool generates visually pleasing grid structures and the relaxation component provides large improvements in structural behaviour of the output gridshells.
There are still possibilities to improve the formfinding tool. An obvious additions would be an algorithm based on the force density method. The algorithm should be able to rearange the members in the grid without changing the surface geometry. This way the designing team will have the choice of dynamic relaxtion optimization, force density optimization or a combination of both methods.

7. Appendix A1

This appendix shows the developed dynamic relaxation script used in the dynamic relaxation component. In the full digital script on the appendix DVD the comments tied to each line of code is within the script. To ease readability of the code the comments have been moved to the margin.

All the lines of code, including methods and classes, seen in the script was developed for this thesis.

7.1 Code Introduction

The Dynamic Relaxation script is given inputs from Grasshopper. The script inputs are *Rhino* geometry in the shape lines and points. The lines form a gridshell model where the points act as fixed supports. The gridshell input does not have to be flat, but can posses a desired shape. The output of the script is a list of relaxed lines making of the relaxed gridshell form.

The script is build using object oriented programming. The objects consists of a Spring system of springs and nodes where the springs join. The overall structure consists of three classes:

- 1. A *SpringSystem* class initializing and managing a set lists of Nodes, Springs and Position vectors. It also calculates nodal forces and performs the numerical Verlet integration.
- 2. A *Node* class constructing Node objects containing variables such as mass, support information and a list of Springs connected to the node.
- 3. A *Spring* class constructing Spring objects containing variables such as start- and end-node of the spring, spring stiffness and rest length of the spring.

The script is connected to a *Timer* mechanism in Grasshopper. The timer updates the script output in a user-set interval. This means, that the output of the script, the new lines, are being output and updated in Grasshopper every time interval.

7.2 Dynamic Relaxation C# code

```
1   if (initialize) {
2     sys.resetLists();
3     sys.Build(lines);
4     sys.SetSupport(supports);
5   }
```

```
6   else if (run) {
7     sys.dt = Math.Sqrt(2 * sys.mass / sys.stiffness) * 0.49;
8     sys.damping = systemDamping;
9     sys.mass = nodeMass;
10    sys.stiffness = springStiffness;
11    sys.UpdateMass();
12    sys.UpdateStiffness();
13    sys.Move();
14  }
```

```
15  Print("Number of nodes: " + sys.nodes.Count);
16  Print("Number of springs: " + sys.springs.Count);
17  Print("Number of supports: " + supports.Count);
18  Print("Current Energy: " + sys.currentEnergy);
19  Print("Previous Energy: " + sys.prevEnergy);
20  Print("Energy difference: " + sys.diffEnergy);
21  foreach (Spring spring in sys.springs) Print("Length: " +
22  SpringLength(sys.currentPositions));
23  foreach (Spring spring in sys.springs) Print("strain: " +
24  spring.Strain(sys.currentPositions));
25  foreach (Spring spring in sys.springs) Print("Force: " +
26  spring.Force(sys.currentPositions));
27  Print("Number of current positions: " + sys.currentPositions.Count);
28  foreach (Node node in sys.nodes) Print("Support: " + node.support);
29  foreach (Vector3d currentPosition in sys.currentPositions)
30  Print("currentPosition: " + currentPosition);
```

Code comments:

(1) The C# script component is connected to a *Boolean* value inside *Grasshopper* named *initialize*. When the *Boolean* value is true this logical operation initializes the system by calling the build-method (3) within the *SpringSystem* object. (2) It also resets all lists, so that the *SpringSytem* object does not contain info of any previous builds. (4) Calls the method that declares if a node is a support or not a support.

(6) This *else if* expression is also connected to a *Boolean toggle* named *run*. When the toggle is *true* the system runs the methods within the *SpringSystem* class to make the system move. If the toggle is switched to *false*, the script will stop running and updating the geometry.
(7) This expression determines an fast converging time step for the *Verlet* integration. Too large time steps will cause failure in the integration and too small time steps will make the convergence take longer [Barnes, 1980].
(8) The *damping* coefficient is updated continuously.
(9) *Node* mass is also possible to change while the script runs.
(10) *Spring* stiffness can also be changed while the script runs. This allows the user to see how the system reacts to a changing *stiffness* in real time.
(11) Functions necessary to update the *Node mass* after the system has been built.
(12) Function necessary to update the *Spring Stiffness* after the system has been built.
(13) This is where the magic happens. This function calls the methods to calculate the forces within the springs and moves the nodes using the numerical *Verlet* integration.

(15-33) The following *Print* statements has been used to track stats within the script. It has also been used to locate errors at an earlier state.
(15) Number of *Nodes* in the system. Useful to quickly see amount of joints in the model.
(16) Number of *Springs* in the model. Also the number of structural members in the Gridshell.
(17) Number of *fixed Nodes* in the system.
(18) Current *kinetic* energy of Nodal movements during the DR process.
(19) Previous *kinetic* energy of Nodal movements during the DR process.
(20) Difference between current- and previous *kinetic* energy. This can be used to create an automatic *convergence* break.
(21) Current *length* of each spring.
(23) Current *strain* of each spring.
(25) Current *spring force* in each spring.
(27) Number of current *position vectors* in the script.
(28) List of support information of all *Nodes*.
(29) List of *position vectors* in the script. Precise information of each nodal position and its orientation in space. Useful for automated fabrication of structural joints.

(31-35) The following expressions deal with the *output* of the script. These lines of code allows the script to output data and geometry to *Grasshopper*.
(31) Main *output*. The *Draw*-methods creates new *Rhino Geometry Lines* using the relaxed current *Nodal position vectors*.
(32) A continuously growing list of *kinetic energy* in the system. The energy is tracked and plotted to visualize when the grid has *converged*.
(33) A list of *vectors* representing the *vectors* of the relaxed *Nodes*.
(34) The current *kinetic energy* of the script.
(35) Difference of energy in the script compared to an earlier state. This part is used to track *convergence* for a possible addition of a break-feature in the script.

```
31  A = sys.Draw(sys.currentPositions);
32  B = sys.histEnergy;
33  C = sys.currentPositions;
34  D = sys.currentEnergy;
35  E = sys.diffEnergy;
```

(36) Creation of the *SpringSystem* object.

```
36  SpringSystem sys = new SpringSystem();
```

(37-51) The *Node class* contains information if the position is a *fixed support* and keeps track of the *Spring* objects connected to the *Node*.
(38-41) Properties of *Node class*.
(38) *Index* of the *Node*. Used later to keep track of the specific *Node* in the list of *Nodes*.
(39) Fictitious *mass* of *Node*. This value can be altered via the *Grasshopper mass slider input*.
(40) List of *Spring* objects connected to the *Node* object.
(41) *True* or *false* value if the *Node* is *fixed* or not. This value is set by a method called within the *SpringSystem* object. The *support* value is later checked to see if the node should be moved during the *Verlet* integration.
(42) Blank *constructor*, used to initialize the object.

```
37  public class Node{
38    public int index;
39    public double mass;
40    public List<Spring> springs;
41    public bool support;

42    public Node() {
43    }
```

(44-50) Constructor that initializes a *Node* object with a specific *index* and *mass*. Inserts a blank list of *Springs* connected to the node and by default sets the *support* value to *false*.
(45) *Integer index* value of the *Node* in the list of *Nodes* in the *SpringSystem*.
(46) Fictitious *mass* of the *Node*.
(47) Default *support* value. This value is changed when the script is initialized which calls the method *SetSupports* shown in (4).
(48) Empty list of *Springs* connected to the node object. This list is populated when *Spring* objects are created with the *AddSpring* method in (110)

```
44    public Node(double _mass, int _index) {
45      index = _index;
46      mass = _mass;
47      support = false;
48      springs = new List<Spring>();
50      }
51  }
```

(52-90)The Spring-class contains methods to calculate the *length,* the *strain* and the *force* of the *Spring* object . It also contains a method to create new *Rhino Line Geometry* output in (31). The Spring class requires a start *Node*, an end *Node*, the spring *stiffness*, the *index* of the *Spring* in the list of *Springs* and the *rest length* of the *Spring* to create a new *Spring* object.
(53-57) *Spring* class properties.
(53) *Spring index*.
(54) Start *Node*.
(55) End *Node*
(56) *Spring stiffness*.
(57) *Spring rest length*.
(58) Blank *constructor*.

```
52  public class Spring {
53    public int index;
54    public Node start;
55    public Node end;
56    public double stiffness;
57    public double restLength;

58    public Spring() {
59    }
```

```
60    public Spring(Node _start, Node _end, double _stiffness, int _index,
61    double _restLength) {
62      start = _start;
63      end = _end;
64      stiffness = _stiffness;
65      index = _index;
66      restLength = _restLength;
67      start.springs.Add(this);
68      end.springs.Add(this);
69    }
```

(60-69) *Constructor* that takes five arguments; start *Node*, end *Node*, *Spring stiffness*, *index* of *Spring* and *rest length* of *Spring*.
(62) Start *Node* of *Spring*.
(63) End *Node* of *Spring*.
(64) *Spring stiffness*.
(65) *Index* of *Spring*.
(66) *Rest length* of *Spring*.
(67) The *constructor* also adds the current *Spring* (*this*) to the list of *Springs* within the start *Node* object. This assures that the *Node* object keeps track of which *Springs* are connected to the *Node*.
(68) This expression does the same as the above but for the end *Node* of the *Spring*.

```
70    public double Length(List < Vector3d > _positions) {
71      Vector3d dir = _positions[end.index] - _positions[start.index];
72      return  dir.Length;
73    }
```

(70-73) Method that calculates the current *length* of the *Spring*. The *Length* method takes a list of *Nodal vector positions* as input.
(71) A temporary *direction vector* is created by subtracting the two Nodal start- and end-*vectors* from each other. The resulting vector runs from the end *Node position* to the start *Node position* and is a vector representation of the *Spring* in both direction and length.
(72) The *length* of the *Spring* is calculated by retrieving the length of the temporary vector.

```
74    public double Strain(List < Vector3d > _positions) {
75      return (Length(_positions) - restLength) / restLength;
76    }
```

(74-76) Method that calculates the *strain* within the *Spring*. The *Strain* method takes the same list of *Nodal vector positions* as the Length method.
(75) The strain in the spring is calculated as the relative difference between the current the *length* of the *Spring* and the original *length* of the *Spring* (*rest length*).

```
77    public Vector3d Force(List < Vector3d > _positions) {
78      Vector3d dir = _positions[start.index] - _positions[end.index];
79      dir.Unitize();
80      double mag = Strain(_positions) * stiffness;
81      return dir * mag;
82    }
```

(77-82) Method that calculates the *Spring force*. The *Spring force* method takes the same list of *positions* as an input.
(78) A *directional vector* is created as in the *Length* method in (71).
(79) The *directional vector* is *unitized* so the *length* of the vector is one.
(80) The magnitude of the spring force is calculated using *Hooke's law* as the multiplicative of the *strain* and the *stiffness* of the *Spring*.
(81) The *Spring force vector* is calculated by multiplying the *unitized directional vector* with the magnitude *scalar*.

```
83    public Line NewLine(List < Vector3d > _positions) {
84      Point3d point1 = new Point3d(_positions[start.index].X,
85      _positions[start.index].Y, _positions[start.index].Z);
86      Point3d point2 = new Point3d(_positions[end.index].X,
87      _positions[end.index].Y, _positions[end.index].Z);
88      return new Line(point1, point2);
89    }
90  }
```

(83-89) Method to draw a new *Line* to be output in the *Draw* method (221) within the *SpringSystem*.
(84) Converts the *vector position* of the start *Node* of the *Spring* into a *point* using the vector coordinates.
(86) Converts the *vector position* of the end *Node* of the *Spring* into a *point* using the vector coordinates.
(88) Returns a new *Line* between the two created points.

(91-228) The *SpringSystem* class contains numerous methods. These methods will be described in the comments as they are too numerous to mention all here.
(92) List of *Springs* in the system. The list is created with *AddSpring* method in (110).
(93) List of *Nodes* in the system. The list is craeted with the AddNode method in (121).
(94-96) List of initial, previous and current *Nodal vector positions.*
(98) Changeable *damping* value.
(99) Value tracking previous *kinetic energy*.
(100) Value tracking current *kinetic energy*.
(101) Difference between current and previous energy in the system.
(102) An expanding list of kinetic energy values. A plot of this list is used to visualize *convergence* in the system.
(103) Changeable *Nodal mass.*
(104) *Time step* used for the numerical *Verlet* integration in (180)
(105) Creation of the gravity vector. The value is inverted to get the structure to have the inverse form needed for an optimal gridshell.
(106) Changeable *Spring stiffness*.
(107) Blank *constructor* of the *SpringSystem*.
(108) When a new SpringSystem is created all previous lists are reset using the *resetLists* method.

```
91   public class SpringSystem {
92     public List<Spring> springs;
93     public List<Node> nodes;
94     public List<Vector3d> positions;
95     public List<Vector3d> prevPositions;
96     public List<Vector3d> currentPositions;
97     public List<Vector3d> resForces;
98     public double damping;
99     public double prevEnergy;
100    public double currentEnergy;
101    public double diffEnergy;
102    public List<double> histEnergy;
103    public double mass;
104    public double dt;
105    public Vector3d gravity = new Vector3d(0, 0, 9.82);
106    public double stiffness;

107    public SpringSystem() {
108      resetLists();
109    }
```

(110-120) Method that creates a *Spring* object from *Rhino Line Geometry* by calling the *Spring* class.
(111-113) Create *position vectors* for the start- and end-point of the *line*.
(114-115) Create start- and end-*Nodes* using the *AddNode* method in (121)
(116) Create the *Spring* Object by calling the *Spring* class.
(118) Add the *Spring* object to the lists of *Springs* in the system.
(119) Return the *Spring* object.

```
110    public Spring AddSpring(Line line, double stiffness) {
111      Vector3d startvec = new Vector3d(line.From.X, line.From.Y,
112      line.From.Z);
113      Vector3d endvec = new Vector3d(line.To.X, line.To.Y, line.To.Z);
114      Node start = AddNode(startvec, mass);
115      Node end = AddNode(endvec, mass);
116      Spring spring = new Spring(start, end, stiffness, springs.Count(),
117      line.Length);
118      springs.Add(spring);
119      return spring;
120    }
```

(121-134) Method that creates a *Node* with a *position vector*. The method then adds them to a list of *Nodes* and *positions*. The method has a built-in check to see if the *Node* already exists.
(122) Searches *positions* list for the specified *position*. Returns -1 if *position* is not found.
(123) If the position does not exist continue and create a new node.
(124) Calling *Node*-class to create a new *Node*. The index of the *Node* is set as the current length of the *Nodes* list. The first *Node* has the index 0, next one 1 and so on.
(125) Add *Node* to list of *Nodes*.
(126) Add *vector position* to list of *positions*.
(127) Return existing node
(128) If the search in (122) finds an existing *position vector* it means that the *Node* has already been created.
(129) Return the *Node* with the found *index*.

```
121    public Node AddNode(Vector3d position, double mass) {
122      int index = positions.IndexOf(position);
123      if (index == -1) {
124         Node node = new Node(mass, nodes.Count());
125         nodes.Add(node);
126         positions.Add(position);
127         return node;
128       } else {
129        return nodes[index];
130       }
134    }
```

```
135   public void resetLists() {
136     springs = new List<Spring>();
137     nodes = new List<Node>();
138     positions = new List<Vector3d>();
139     prevPositions = new List<Vector3d>();
140     currentPositions = new List<Vector3d>();
141     resForces = new List<Vector3d>();
142     histEnergy = new List<double>();
143     prevEnergy = 0;
144     currentEnergy = 0;
145   }
```

(135) Method that resets the lists used in the *SpringSystem* class. This is to ensure that the script starts anew when a new system is build.
(136-144) The lists are reset by creating new empty lists to be populated when a new system is build.

```
146   public void Build(List < Line > lines) {
147     foreach (Line line in lines) {
148       AddSpring(line, stiffness);
149     }
150     foreach (Vector3d position in positions) {
151       prevPositions.Add(new Vector3d(position.X, position.Y,
152       position.Z));
153       currentPositions.Add(new Vector3d(position.X, position.Y,
154       position.Z));
155       resForces.Add(new Vector3d());
156     }
157   }
```

(146-158)The build method initiates creating the objects by calling the *AddSpring* method for all *Lines* in the input model. It also initiates the *position vector* lists that are to be used in the numerical *Verlet* integration in (180).
(148) Call *AddSpring* method on all *Lines* in the list.
(150-154) Create the previous- and current *position vector* lists.
(155) Create an empty resulting force *vector*. Applied loads and link forces from the *Springs* will be added to this vector and used in the *Verlet* integration.

```
158   public void SetSupport(List < Point3d > supports){
159     foreach (Point3d point in supports){
160       Vector3d temp = new Vector3d(point.X, point.Y, point.Z);
161       int nodeIndex = positions.IndexOf(temp);
162       if (nodeIndex != -1) {
163         nodes[nodeIndex].support = true;
164       }
165     }
166   }
```

(158-166) Method that sets the Boolean support value in all Nodes.
(159) Loops through each *point* in the input of fixed *Rhino points*.
(160) Create a temporary *position vector* for each support point.
(161) Uses same search function as in (122). If the temporary *position vector* matches any of the *position vectors* build in (126) the function will return the *index* of the *position vector*. If that is the case, the support value is changed to True. *Nodes* with support value true will be skipped in the *Verlet* integration as they are fixed.

```
167   public void SetForces(List < Vector3d > _positions){
168     foreach (Node node in nodes){
169       Vector3d nodeForce = new Vector3d(0, 0, 0);
170       foreach (Spring spring in node.springs){
171         if (spring.start == node){
172           nodeForce -= spring.Force(_positions);
173         } else {
174           nodeForce += spring.Force(_positions);
175         }
176       }
177       resForces[node.index] = nodeForce;
178     }
179   }
```

(167-179) Method that takes the *Spring* force calculated in (77) and adds it to the resulting forces on the respective *Nodes*.
(168) Loops through all *Nodes* in the system.
(169) Create an empty *Node* force vector.
(170) Loop through the list of *Springs* connected to each node.
(171-175) Adds the *Spring* force from (77). The force vectors are oriented correctly by separating start- and end *Nodes*. This ensures that all *Spring* forces on the *Node* are acting against each other in equilibrium as would happen in reality.
(177) Add the resulting *Nodal* force to a list of resulting forces on all the *Nodes*.

(180-206) The method that performs the *Verlet integration*. The method first checks if the *Node* is a support. If not, the *Verlet* integration continues.
(182) The numerical *Verlet* integration loops through all the *Nodes* in the system.
(183) If the *Node* is not a *support* the code continues and reforms the numerical integration.
(184) The *position vector* that is to be moved is selected using the index of the *Node*.
(185) The *acceleration vector* of the *Node* is calculated using Newton's second law of motion by dividing the resulting forces on the node with the *Nodal* mass.
(186) Gravity is added to the resulting *acceleration vector*.
(187) The current *velocity vector* of the *Node* is found by subtracting the current *position vector* of the Node with the previous *position vector* from the previous iteration.
(188) The *Verlet* integration is performed to calculate the next *position vector* of the *Node*.
(190) The current *kinetic energy* is calculated.
(191) The previous *position vectors* are updated to current *position vectors*.
(192) The current *position vectors* are updated to next *positions vectors* just found with the *Verlet* integration.
(195) Adding the current energy to a histogram list to illustrate convergence in Grasshopper. The value is multiplied with a large number because the kinetic energies are very small numbers.
(196-204) Is an example of a break function not implemented in the script. It tracks the difference in kinetic energy over a larger number of integration iterations. Once the difference reaches zero the grid has converged.

```
180   public void Verlet(){
181     currentEnergy = 0;
182     foreach (Node node in nodes) {
183       if (node.support == false) {
184         Vector3d currentPosition = currentPositions[node.index];
185         Vector3d a = resForces[node.index] / node.mass;
186         a += gravity;
187         Vector3d v = currentPosition - prevPositions[node.index];
188         Vector3d nextPosition = currentPosition + (1 - damping) * v +
189         0.5 * a * dt * dt;
190         currentEnergy += 0.5 * node.mass * v.Length * v.Length;
191         prevPositions[node.index] = currentPosition;
192         currentPositions[node.index] = nextPosition;
193       }
194     }
195     histEnergy.Add(currentEnergy * 1000000);
196     double sum1 = 0;
197     double sum2 = 0;
198     int numEnergy = histEnergy.Count;
199     if (numEnergy > 11){
200       for (int i = numEnergy - 1 ; i > numEnergy - 6 ; i--){
201         sum1 += Math.Abs(histEnergy[i] - histEnergy[i - 5]);
202         sum2 += histEnergy[i];
203       }
204       diffEnergy = sum1 / sum2;
205     }
206   }
```

(207-211) Method necessary to update the *Spring stiffness* in real-time without having to restart the dynamic relaxation algorithm.

```
207   public void UpdateStiffness() {
208     foreach (Spring spring in springs){
209       spring.stiffness = stiffness;
210     }
211   }
```

(212-216) Method necessary to update the *Nodal mass* in real-time without having to restart the dynamic relaxation algorithm.

```
212   public void UpdateMass() {
213     foreach (Node node in nodes){
214       node.mass = mass;
215     }
216   }
```

(217-220) Method that calls the Verlet and SetForces methods in (167) and (180). This method is only active if (6) is true, thus acting as a start button for the script.

```
217   public void Move() {
218     SetForces(currentPositions);
219     Verlet();
220   }
```

(221-229) Method calling all the required methods to create new *Rhino Line Geometry* to visualize the dynamic relaxation process in real-time.
(222) Create an empty list of *Rhino Line Geometry*.
(223) Loop through all *Springs* in the system.
(224) Add a new *Line* using the *NewLine* method within the *Spring* class (83).
(226) Return the lines for output.

```
221   public List<Line> Draw(List < Vector3d > _positions) {
222     List<Line> newlines = new List<Line>();
223     foreach (Spring spring in springs) {
224       newlines.Add(spring.NewLine(_positions));
225     }
226     return newlines;
227   }
228 }
```

8. Appendix A2

8.1 Deriving the catenary equation

The mathematical equation for the chain can be derived using the following assumptions:

- The chain is thin enough to be regarded as a curve.
- The chain is flexible enough so that the tension force in the curve is parallel to the curve.

The parametric path of the chain is given parametrically by the position vector $\bar{r}$ and the length of the chain is given by s. It can be written as follows

$$\bar{r} = (x, y) = (x(s), y(s)) \tag{15}$$

The unit tangent vector of the path is given by

$$\bar{u} = \frac{dr}{ds} \tag{16}$$

To derive a differential equation for the curve further assumptions must be made:

- The lowest point of the chain is named c, this is commonly referred to as the catenary vertex.
- Only the right side of the chain is being considered due to symmetry. So the length s is measured from c to the right.

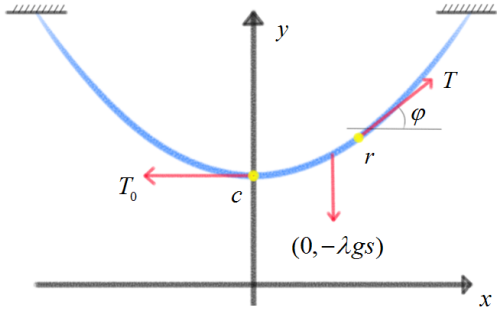


Figure 144: The catenary equation is derived using the variables shown in the figure.

The differential equation can be found by considering that there must be equilibrium in the chain. The three forces acting on the section of the chain from c to the arbitrary point r are as follows:

Horizontal tension at c : $(-T_0, 0)$

Tension of chain at r : $T \cdot \bar{u} = (T \cdot \cos\varphi, T \cdot \sin\varphi)$

Weight of chain: $(0, -\lambda g s)$

Where λ is the mass pr. unit length and g is gravitational acceleration.

The sum of horizontal and vertical forces must be zero. $\sum F_x = 0$ and $\sum F_y = 0$ hence the following equilibrium equations can be found

$$T_0 = T \cdot \cos\varphi \tag{17}$$

$$T \cdot \sin\varphi = \lambda g s \tag{18}$$

Dividing the two equations with each other gives

$$\frac{T \cdot \sin\varphi}{T \cdot \cos\varphi} = \frac{\lambda g s}{T_0} = \frac{\sin\varphi}{\cos\varphi} = \tan\varphi = \frac{dy}{dx} \tag{19}$$

Introducing the parameter $a = T_0 / \lambda g$ (the length of a chain section whose weight is equal to the tension at *c*) into the expression returns a simplified differential equation

$$\frac{dy}{dx} = \frac{s}{a} \tag{20}$$

In order to solve the differential equation to find the catenary equation an expression for the curve length must be known. The curve is represented by a continuous function such that

$$f'(x) = \frac{dx}{dy} \tag{21}$$

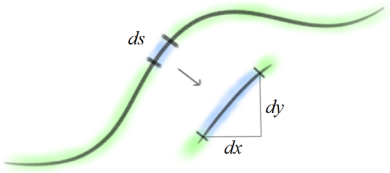


Figure 145: The infinitesimal section of the function.

If an infinitesimal section of a given arch curve is considered, the length can be described using Pythagoras's theorem.

$$ds^2 = dx^2 + dy^2 \tag{22}$$

With a little re-writing and insertion into the differential equation the following can be found

$$\frac{ds}{dx} = \sqrt{1 + \left(\frac{dy}{dx}\right)^2} = \frac{\sqrt{a^2 + s^2}}{a} \tag{23}$$

Further re-writing gives

$$\frac{dx}{ds} = \frac{1}{\frac{ds}{dx}} = \frac{a}{\sqrt{a^2 + s^2}} \tag{24}$$

$$\frac{dy}{ds} = \frac{\frac{dy}{dx}}{\frac{ds}{dx}} = \frac{s}{\sqrt{a^2 + s^2}} \tag{25}$$

Integration gives

$$\int \frac{s}{\sqrt{a^2 + s^2}}\, ds = y = \sqrt{a^2 + s^2} \tag{26}$$

$$\int \frac{a}{\sqrt{a^2 + s^2}}\, ds = x = a \operatorname{arcsinh}\left(\frac{s}{a}\right) \tag{27}$$

By combining the two equations for *x* and *y*, the parameter *s* can be eliminated and the catenary equation can be expressed as

$$y = a \cosh\frac{x}{a} \tag{28}$$

9. Appendix A3

The third appendix is a DVD with the developed tools and algorithms made during the writing of this thesis. In order to open the files Rhinoceros and Grasshopper is needed. Both can be downloaded as evaluation versions for free at rhino3d.com. The appendix also includes videos of the form finding tool in action. The videos showcase the real-time relaxation of different grids and the equilateral grid-generation algorithm.

10. References

Larsen, Dr. Olga Popovic. *Bridging the gap between architecture and structural engineering*. The University of Sheffield, School of Architecture .Thomas Telford Limited, 2003.

Chilton, John. *Heinz Isler (Engineer's Contribution to Architecture)*. Thomas Telford Ltd, 2000.

Arup, *Traces of Peter Rice* . Documentary used in exhibition to honor Peter Rice. Made by Arup. Source: http://www.youtube.com/watch?v=v8ubOlkQCf4

Rice, Peter, *An Engineer Imagines,* Artemis, London 1994.

Dimčić, Miloš. S*tructural Optimization of Gridshells based on Genetic Algorithms.* institut für Tragkonstruktionen und Konstruktives Entwerfen der University Stuttgart, 2011.

Kolarevic, Branko. *Architecture in Digital Age - Design and Manufacturing*. Taylor and Francis, 2005.

Toussaint, M.H. *A Design Tool for Timber Gridshells*. Delft University of Technology, 2007.

Hoefakker, Jeroen Hendrik. T*heory Review for Cylindrical Shells and Parametric Study of Chimneys and Tanks*. 2010.

Sartwell, Crispin, *"Beauty", The Stanford Encyclopaedia of Philosophy (Fall 2012 Edition), Edward N. Zalta (ed.)*. Source: http://plato.stanford.edu/archives/fall2012/entries/beauty/ - accessed 31-05-2013.

Osserman, Robert. *Mathematics of the Gateway Arch*. Notices of the American Mathematical Society, 2010.

Wakefield , D.S. *Engineering analysis of tension structures: theory and practice.* Tensys Limited, 1999.

Douthe & Baverel. *Form-finding of a Gridshell in Composite Materials*. Journal of the International Association for Shell and Spatial Structures, 2006.

Hagsten, L.G. & **Nielsen**, M.P. *Mekanik/Plasticitetsteori - Grundlag for modellering af konstruktioner* (Danish), Aarhus University, 2010.

Otto, Frei. *Spannweiten: Ideen und Versuche zum Leichtbau ; ein Werkstattbericht*. Ullstein, 1965.

Bak, Andreas. *Interactive Formfinding For Optimized Fabric-Cast Concrete*, University of Bath, 2011.

Swope, William. *A computer simulation method for the calculation of equilibrium constants for the formation of physical clusters of molecules: Application to small water clusters*. The Journal of Chemical Physics 76, 1982. .

Barnes, Michael & **Dickson**, Michael. *Widespan Roof Structures*. Thomas Telford, 2000.

Stephan, S, **Sánchez-Alvarez**, J & **Knebel, K**. *Reticulated Structures on Free-form Surfaces*. Mero, 2013.

11. Declaration of honesty

I hereby declare that this thesis is my own work.

Peter Vejrum, 2013

List of Figures