Politecnico di Torino

Master's Thesis in Civil Engineering

Parametric Design in Structural Optimization

Long span Roof Optimization through parametric design

A.A. 2021-2022

Supervisors:
Prof.Marano Giuseppe Carlo
Laura Sardone

Company:
LGC Ingegneria-Architetture

Candidate:
Sherzod Salimov

Contents

Table of Figures

1. Abstract:

This study represents a snapshot of generative design technology looking at what it is, how it works, how it's being used and why it's changing the way we design the next generation of things. This research objective is demonstration of possibilities in design process, different parametric ways and their usage in the optimization process. The idea is providing clear image of relationships between long and big structures and parameters that may cause variation in results even after optimization process depending on the parameters amount.

The structure was modelled through the parametric method, on Rhinoceros with the help of Grasshopper. Rhino 3D is a free form surface modeler that uses the modelling by curves technique (NURBS or Non-Uniform Rational Basis Spline). NURBS is a mathematical model that renders curves and surfaces in computer graphics. This mathematical model offers great flexibility and precision during the whole 3D modelling process. With this technique, you work with curves when modelling rather than by linking polygons. You create your three-dimensional surface by manipulating curves. This kind of modelling uses adaptive mesh, which enables you to optimize the number of faces forming the object's surface. This modelling technique is the most accurate there is. All the necessary data which represent the long span roof were collected from database of Polytechnic University of Turin. By constructing the model in a parametric way, which means making the structural components dimensions variable, successful results in terms of optimization were reached.

The key findings of this research were finding relations between different aspects of structure such as: mass, displacement, shape, etc..

2. Introduction

Optimization methods applied to building performance simulation, or simple BSO (Building Simulation Optimization), are beneficial since many variables affect building performance, such as form, layout, envelope materials, orientation, and landscape design. These variables are usually defined qualitatively and mainly considered in the conceptual design phase. Consequently, designers lack sufficient information to make effective and appropriate decisions that lead to high-performance buildings, and the subject has been the focus of many investigations. Many applications demand skills that most architects are unfamiliar with, and a smooth connection with standard and parametric modelling programs is still missing. In this sense, using the available tools must happen with parsimony. On the one hand, they are credited to assist the design and decision process, optimizing the final building performance with relatively low cost. On the other hand, the tools also present limitations and may not accurately represent the study's objective. Therefore, the more structured the processes and the more substantiated the variables considered in the project, the greater the chances of finding optimal solutions.

BSO has been explored to find optimum alternatives among potential combinations of several parameters that involve passive design or responsive climate strategies. Tian et al. studied the integration of optimization algorithms into simulation-based design processes and

summarized the different procedures that can be applied within the technique, which involve multiple steps, described in a tutorial form by Konak et al.

Optimization procedures	Description
Three phase optimization	The optimization process occurs in three phases: preprocessing, running the optimization, and postprocessing
Multitime design optimization	Building performance simulation with optimization methods is applied at each stage of building design
Sensitivity analyses and optimization	Sensitivity analyses are used to narrow the variables range, determine the significant ones, and filter those with little impact on the objectives. Optimization is then conducted with a narrow variable range

Figure 1 Optimization procedures

Predominant is the plugin for the Rhinoceros software, the Grasshopper. Published studies that use the 3D parametric platform include building optimization concerning both its overall passive performance and energy efficiency.

In this context, parametric 3D modeling and BSO-based process guided this research development. Both procedures are implemented within the design platform to facilitate optimization practice in the early design stages. The study aims to optimize long span roof in terms of topology and geometry.

This thesis studies the integration of optimization into architectural design processes by developing, benchmarking, and user-testing novel computational design tools that are more efficient and better acknowledge designers' preferences than existing ones. Improving the integration of architectural design in a parametric way helps to achieve optimum results in terms of shape, topology, and size. The important thing in generating such kind of parametrised models is that it allows a designer to have thousands of choices and possibilities which can also be expressed by term "space". This methodology has become popular in a recent year, because of highly economic benefits and design ideas. Dealing with regular shaped structures do not require to introduce something to be created and tested, because regulations and codes already exist to solve stability problems and etc. On the other hand, nowadays the companies or countries that estimates themselves as a modern, more and more push towards using optimization. As written above, optimization is easier when it is possible to vary structural elements or substructural elements which will be discussed later on chapter related to generative design.

Mathematical optimization defines the possible solutions to a problem and their evaluation criteria unambiguously, in contrast to co-evolving architectural design problems. Nevertheless, given the potential of mathematical optimization for automated DSE on the one hand, and the increased use of computational methods such as parametric design and simulations by architectural designers—described as a "sea change" by Scheer (2014)—on the other, one might expect optimization to be integral to contemporary computational design processes. But, due to several challenges, this is not the case. These challenges—which are

discussed further in the next chapter—include: Skepticism about using computers for architectural design, the limited application of parametric design and performance simulations, lack of knowledge on and incentives for optimization, a bias for inefficient optimization methods, a lack of state-of-the-art, easy-to-use optimization tools, and the problematic integration of optimization with architectural design.

Finally, this paper's contents are as follows: **Section 2** provided general overview on generative design, organized in two steps. Connections between generative design and structural optimization are presented in **Section 3**, while **Section 4** summarizes the gridshell models which describes the long span roof in study. Problem description, results' presentation, and discussions are the **Section 5** themes. At last, **Section 6** closes the paper with the mains conclusions found, remarks, and future works.

3. Generative Design: an overview

Generative approaches are recently becoming more and more applied in a variety of technical fields. By implementing artificial intelligence tools, they can elaborate and propose to a human user a series of plausible solutions for a design problem. That is, a number of alternative configurations that i) satisfy a set of imposed design constraints and ii) try to maximize a goal function passed to the algorithm. The proposed alternatives are the result of an iterative exploration of the related solution space that is guided by an artificial intelligence. Thanks to the significant increase of computing power available, these tools have recently observed a growing interest in the design community. Generative Design (GD) has taken its first steps in the architectural field and has generally been first applied in open-problem scenarios characterized by large design spaces. In this context, the term "Generative Design" Computer-Aided Design & Applications, refers to a series of tools, implementing artificial intelligence methods and algorithms, applied to solve design problems. From a practical point of view, GD tools essentially seek for a solution of a problem expressed with a mathematical formulation; this often results in an iterative optimization process that tries to minimize an objective function. Accordingly, GD has proven itself useful to identify uncommon solutions that do not fall within the typical set of shapes or configurations used. As a result, GD tools have been first exploited mainly to encourage divergent thinking and creativity. This remains a distinctive aspect of the technology that has brought to its application in areas where aesthetics and innovativeness are important in the product development process, such as product design or the automotive sector.

Generative design is used to provide practitioners the ability to quickly explore, optimize, and make informed decisions to complex design problems. Think of generative design software as an assistant that helps with creating, testing, and evaluating options.

Generative design tools that produce optimum forms for products and buildings without human intervention are set to transform both the physical world and the role of the designer, according to software experts. The software can automatically make aircraft lighter, buildings stronger and trainers more comfortable – with the designer acting as a "curator", rather than making all the decisions. The emerging technology uses algorithms to generate every possible permutation of a design solution. The designer simply enters a set of parameters and then chooses the best outcome generated by the software.

Generative design, which mimics the way organisms evolve in the natural world, was the hot topic at Autodesk University – a three-day technology conference held in Las Vegas last November.

"Generative design is a departure from the way that we have traditionally done design," said Jeff Kowalski, chief technology officer of software company Autodesk. "But these technologies are not a threat, they're more like superpowers." The digital design method relies on cloud computing to create a multitude options based on a set of fixed parameters. It often produces fluid, lattice-like forms, as the software imitates the way nature creates the most efficient structures possible.

Generative design is similar to parametric design, which involves the user gradually tweaking spatial parameters until a desired form is reached. It is responsible for some of the futuristic-looking buildings by Zaha Hadid Architects, MAD Architects and more. But with generative design the parameters are fixed, and the computer quickly builds all the possible solutions for the user to chose from – learning their preferences as it does so. Architects and designers can specify their desired result – a chair, a floor plan or a facade pattern – along with parameters like materials, manufacturing methods, and cost constraints.

"The computer generates not only the shape, but many, many, many options," said Autodesk strategist Diego Tamburini, who described the method as a "brute-force approach" to design. Many designers are experimenting with generative design to produce new forms and improve existing products.

Generative design has been around since the early 1990s, and was first used to create simple artwork and animations. But it is now trickling into a wide range of CAD programmes and therefore becoming accessible to more designers. The improved processing power of computers and availability of cloud computing – using a network of remote internet servers to store, manage and process data – are also accelerating the uptake.

Generative design platforms currently available include NodeBox, Element and Generate – all of which provide similar tools. Autodesk is set to launch its own generative design tool Dreamcatcher early this year. It will be integrated into some of the company's current 3D design software programmes, like Fusion 360 and Inventor.

Generative design promises to save time on the design process, save material by creating the most efficient structures and save money by working out the most cost-effective way to manufacture them. But it is not without potential sceptics. There's a risk that designers will feel threatened by removing a large part of the creative process from their work – simply left to chose between a set of options each time.

i. What goes into a generative design process?

Generative design allows for a more integrated workflow between human and computer. This workflow involves the following stages:

1. Defining the problem
2. Gathering data
3. Setting evaluation criteria
4. Generating the model
5. Evaluating the results
6. Evolving the design
7. Selection and refinement

As you can see, some of the steps require human input. This goes through many iterations before you can arrive at the final design. Below they are elaborated more in detail:

Defining the Problem

The first step in generative design is to define the problem. That means the designer and their client need to decide on what they're building. They should set the basic design parameters and conditions, as well as what to exclude.
Besides that, they also have to decide on what makes a suitable design. And that's in addition to what aspects they want to minimise or maximise.
The central role of this phase is to help you understand and define the project. In the process of breaking the project into smaller and simpler components, you'd want to ask questions and find answers for them.

Gathering Data

With the definition phase done, it's time to start gathering data. This stage focuses on project requirements and constraints. They can vary greatly, depending on the building type and location.
For example, in the design of an exhibition hall, you'd have to know the space's properties. These include boundaries and the locations of restrooms, entrances, and exits. Also, the desired column positions and the overall size and shape are potential input parameters.
Next on the list are design limitations. Part of these may be client requests and others pursuant to pre-existing boundaries.
This phase further defines the project and supplies the necessary details for the starting iterations.

Setting Evaluation Criteria

In the third phase, the designer would determine the project goals or evaluation criteria with the client. And the client would have to define what they want from the design and formulate goals and criteria for evaluating the project.
It's vital to define the criteria as precisely as possible. This might make the design and solution more complex. But modern computers can handle that, and GD software returns better and more relevant results with more input parameters.
With poorly defined goals, GD software packages may not be able to offer any workable solutions. The results are likely random and potentially useless. And this would only serve to delay the project.

Generating the Model

After defining the project goals and evaluation criteria, it's time to generate the model. It'd be good to write down all the design steps because that's going to make the model generation a lot easier.
To get the GD software to work, you have to input the design constraints and other important aspects of the project. The size, quantity, and cost, for example.
Finally, the designer should consider the relationships between the design elements. Furthermore, there's also the design's relationship with the environment to consider. If there are any relative constraints, now is the time to enter them.
After that, you can execute the software to come up with a range of design alternatives.

Evaluating the Results

In this step, the software uses the established metrics to score and rank all the design alternatives it came up with. It then selects the highest-ranking solutions in each category and uses them as the basis for the next set. That's how it learns and improves on the overall quality of each set.
The quality of the results directly depends on the criteria set in previous steps. So if the GD software received poorly defined metrics, the overall quality of designs will be off. Due to that, this evaluation phase will likely not identify much improvement. And you probably won't get the best possible design.

Evolving the Design

In the sixth phase of generative design, the software picks the best design alternatives and bases new designs on them.
The role of this phase is to cut out non-optimal solutions and find the best ones in each category. You might need to refine the search metrics in this phase to increase the chances of getting the best final result.
Typically, a GD process has several generations of designs called iterations. Some projects might need as many as 100 or even more iterations. And it's not unusual to have tens or hundreds of designs per iteration.

Selection and Refinement

The final phase of the generative design process is selection and refinement. This is where you explore the best design alternatives that the software has come up with. Then, you'd select a narrow range of designs, preferably those that perform well in all categories.
The refinement part focuses on a small number of hand-picked designs. The designer manually improves them to meet all the criteria before selecting the best solution with the client's input or consent.

ii. Difference between generative and parametric design

First of all, we have to ask ourselves why are we trying to distinguish these factors, how was it before? So, the answer is, previously we have been using passive design which is the interaction of human plus computer.

$$\boldsymbol{Human + Computer = Limited\ Design\ Options}\ (\textbf{Passive Design})$$

Now, what if we talk about Generative Design, it still uses the human, but with other tool called algorithms and the output is the millions of design options. So, the difference

between Generative and Passive Design is in the output of the process, as in traditional way we might lack of many things, whereas in PD and GD process is much more integrated and has a lot of advantages. So, the answer for the question is PD and GD are so close to each other that some people may not be able to distinguish them.

The Oxford Dictionary defines parameter as “a numerical or other measurable factor forming one of a set that defines a system or sets the conditions of its operation,” or as “a limit. which defines the scope of a particular process or activity,” and the word parametric as “relating to or expressed in terms of a parameter or parameters.”

On the basis of the literature, we can synthesize PD into a design process based on algorithmic thinking, that uses parameters and rules to constrain them. Therefore, PD is an approach that describes a design symbolically based on the use of parameters. As an example, instead of designing walls using exact positions, lengths, heights, and thicknesses, these properties are replaced by symbolic parameters that have specific domains. The result is a symbolic representation of a set of walls. This approach is commonly used in BIM tools and is expressed in the concept of a family/object that describes sets of building elements. For example, in the case of a wall family, each combination of parameter values corresponds to a different wall. In this example, a direct relation exists between the parameters and the resulting design, but in other cases, this relation might not be evident because the parameters can be used in an intricate way to produce complex designs.

The proposed definition encompasses everything without unnecessarily constraining its applicability. In this regard, associative geometry is not a requirement for PD, although it emerges from its practical use. In sum, if a design depends on parameters, it is PD.

Nowadays, PD might be confused with Passive Design method which was explained at the beginning, because in a BIM platform some can use Revit or some other software. In Revit the created model can be changed in two different ways, one is directly through the model where you work with output, while another way is to change the parameters. Working by means of parameters (Parametricism) is basically flexing the geometry by set of parameters.

Figure 2 Wald Disney Concery Hall

Did you know that the Walt Disney Concert Hall in Los Angeles was one of the first parametrically designed buildings?
Frank Gehry designed the building using a variety of techniques. He combined sketches and paper models to imagine the exterior. He further imagined different variations of symmetrical forms blending into a compact shape. Also, it represents musical movement and the dynamism of the City of Angels.

Now, to make it clear, some concepts on generative design are explained to be able to differentiate Parametric design with Generative design.

The Cambridge Dictionary defines generative as the "capacity to produce or create something." Some authors define GD as a design process that mainly refers to evolutionary techniques in both the creation and production processes of design solutions (Fischer and Herr, 2001; Frazer et al., 2002;Zhang and Xu, 2018), whereas others do not restrict GD to evolutionary processes, considering it a design approach based on algorithmic or ruled-based processes that generate multiple and, possibly, complex solutions (Bernal et al.,2015; Bukhari, 2011; Chase, 2005;).Moreover, several authors consider approaches, such as algorithmic generation, cellular automata, evolutionary methods, generative and shape grammars, L-systems, self-organization, agent-based models, and swarm systems, as part of GD.

Algorithmic modelling through softwares such as Dynamo or Grasshopper could be stepping stone for the concept of Generative Design. These methods also use parameters, but they are related to each other, so it is possible to play with them in a different way. If we think about single algorithmic modelling, it still requires human activities, which was called before Passive Design but in case of GD, for example you can insert all data matrix into cloud, so when you run your data, it automatically finds thousands of possible ways to construct your model.

GD must be differentiated from other terms, such as PD. Thus, we define GD as a design paradigm that employs algorithmic descriptions that are more autonomous than PD. In GD approaches, after starting the generative process, the system executes encoded

instructions until the stop criterion is satisfied. Consequently, GD-based methods can generate complex outputs even from simple algorithmic descriptions. In many cases, the algorithm is difficult to correlate with the generated output, thus making the outcome difficult to predict by merely reading the algorithmic description.

The non-traceability between GD programs and the generated designs is one of the main reasons GD methods produce unexpected results, such as the “happy accidents”.

Autodesk's Toronto offices are a prime example of generative design in practice. When Autodesk unveiled their new office building in 2017, it was the first AI-designed offices of that stature. The offices span three floors with a total size of 60,000 square feet.

Figure 3 Autodesk's Toronto office

Individuals who have experience with CAD can easily make the leap to generative design software. In addition to generative design-specific software, many CAD programs now offer integrated generative design tools or plug-ins.

Generative design software, however, offers users more than the traditional functionality of CAD software. These tools enable users to input information on forces, materials, costs, and the like into design profiles as well as prioritize and refine parameters based on graphical representations of design solutions.

By no means an exhaustive list, the following are popular software programs offering generative design capabilities:

Fusion 360 from Autodesk: Fusion 360 offers users a powerful set of modelling tools, including sketching, direct modelling, surface modelling, parametric modelling, mesh modelling, rendering, and much more. Its generative design capabilities enable users to identify design requirements, constraints, materials, and manufacturing options to generate manufacturing-ready designs, all the while enabling users to leverage the power of machine learning and AI to review cloud-generated design outcomes based on visual similarities, plots, and filters. Learn more.

Creo Generative Design from PTC: Leveraging the cloud, this software enables users to create optimized design concepts and simultaneously explore and test numerous design iterations quickly. It highlights the iterations that best match a user’s objectives based on design parameters the user sets. Within the Creo design environment, this software promises

to generate high-quality, lower-cost, and manufacturable designs, all in less time than leading competitors. Find out more.

nTop Platform from nTopology: The nTop Platform software promises users complete control over every aspect of the optimization process and its outputs. Leveraging advanced generative tools, users can create custom, reusable workflows tailored to an application's unique requirements. Featured capabilities of this program include unbreakable modelling and latticing operations, topology optimization, reusable design workflows, field-driven design, and mechanical-thermal finite element analysis simulations. Read more.

NX from Siemens: Beyond generative design, the main feature that NX offers is the digital twin technology, which promises users a flexible, powerful, and integrated solution to help them streamline the design and delivery of better products. NX combines design interoperability, validation, model-based definition, and more to help users move products through research and development faster and at lower costs while improving product quality. Get more information.

MSC Apex Generative Design from MSC Software: This program promises users an end-to-end solution for making high-precision metal components more quickly and with less human intervention than its competitors. MSC Software reports that users experience reductions in initial design and setup time by as much as 80 percent. At a glance, the software combines simplicity, automated design, import and validation, and direct output in one process. Check out the program.

4. Problems in Integration of Optimization with Architectural Design

The contrast between co-evolving architectural design problems and well defined optimization problems points towards a more conceptual reason for this limited application. Architectural designs must fulfil many quantitative and non-quantitative evaluation criteria which one often cannot formulate a priori. Rather, such criteria are continuously redefined during design processes (Dorst and Cross 2001). One possibility is to apply optimization to well-defined subproblems, instead of using it to generate full building designs. But, due to the "wickedness" of co-evolving architectural design problems, this is possible only when problem definitions have stabilized and the potential for efficiency improving design changes has diminished, i.e., towards the end of design processes (Architectural/Engineering Productivity Committee 2004). The integration of optimization into architectural design processes is thus problematic. The scepticism about computational tools in the architecture community is difficult to address directly, but parametric design and performance simulations are likely to become more widespread in near future.

The complexity of architectural design problems is best characterized by Rittel and Webber's oft-cited dictum, "planning problems are wicked problems" (1973). For such wicked problems, "setting up and constraining the solution space and constructing the

measure of performance … is more essential than the remaining steps of searching for a solution." According to Cross and Roozenburg (1992), the view of architectural design problems as ill-defined by definition is one of the key differences between models of architectural design processes and models from engineering design. In other words, architects can define design problems 65 in diverse ways, with different implications for potential solutions. From this perspective, the Co-Evolution of design problems and solution spaces (Dorst and Cross 2001) results from the "wickedness" of architectural design problems. This "wickedness" also implies a severe difficulty in applying optimization methods, which demand explicitly defined problems, to architectural design problems. According to Lawson (2006), another aspect of the complexity of architectural design problems is that their solution requires a "holistic response" based on "skilled judgement." Unsurprisingly, given his skeptical stance on the use of computers in architectural design processes, he states: Rarely can the designer simply optimise one requirement without suffering some losses elsewhere. Kotnik (2010) has a more positive view of computational design approaches but, fears that performance-based design and ADO detract from the potential of these approaches for a "systematization of knowledge and methods on design." In his view, the application of ADO hinders the designer from understanding architectural design problems: For architectural design, the output-driven perspective onto the computational function of a performative design strategy is a pitfall because it encourages a tendency towards optimisation and with it, an economisation and closing up of architectural thinking towards parametric manipulation. The importance of an algorithmic description of the computational function does not lie in the possibility of computing an optimal solution, but rather in the ability to control precisely the geometric relation between architectural elements under consideration. Speaking from a theoretical perspective, Kotnik sees the rule-based definition of an architectural design as a contribution to design knowledge in the spirit of Mitchell's "architectural grammars."
Thesis acknowledges the difficulties highlighted by the theorists above and does not assume that optimization methods can solve architectural design problems outright. Rather, it contends that optimization methods contribute to architectural design processes by offering good solutions for bounded sub-problems and by providing a medium for reflection. In other words, this thesis aims to enhance designer's capabilities through optimization, instead of replacing them with it.

i. State of the art on Optimization Driven Architectural Design of Structures

Architectural design is increasingly influenced by systematic, computational methods such as parametric design (Woodbury 2010), performance simulation (Malkawi 2005), performance-based design (Kolarevic 2005; Oxman 2006; Hensel 2013), and building information modelling (BIM) (Kensek 2014). Such methods allow designers to develop geometrically complex designs, while more accurately predicting their future performance (e.g., Luebkeman and Shea 2005; Lin and Gerber 2014; Wortmann and Tuncer 2017). When designers parametrically define design spaces and numerically simulate the performance of

individual design candidates, mathematical optimization methods can identify well-performing designs.

In the ADO literature, examples that develop a full architectural design through optimization are rare. Dillenburger and Lemmerzahl (2011) and Lin and Gerber (2014) present exceptions that employ genetic algorithms (GAs) to synthesize building designs from criteria such as fulfilment of the building's program, energy efficiency, and cost. But the constrained design space considered by these examples appears to confirm Rittel's insight that the definition of a design space is more decisive than the optimization result. For example, the designs in (Figure 4) exhibit the same overall density, floor height, number of floors, circulation core locations, and similar room shapes.

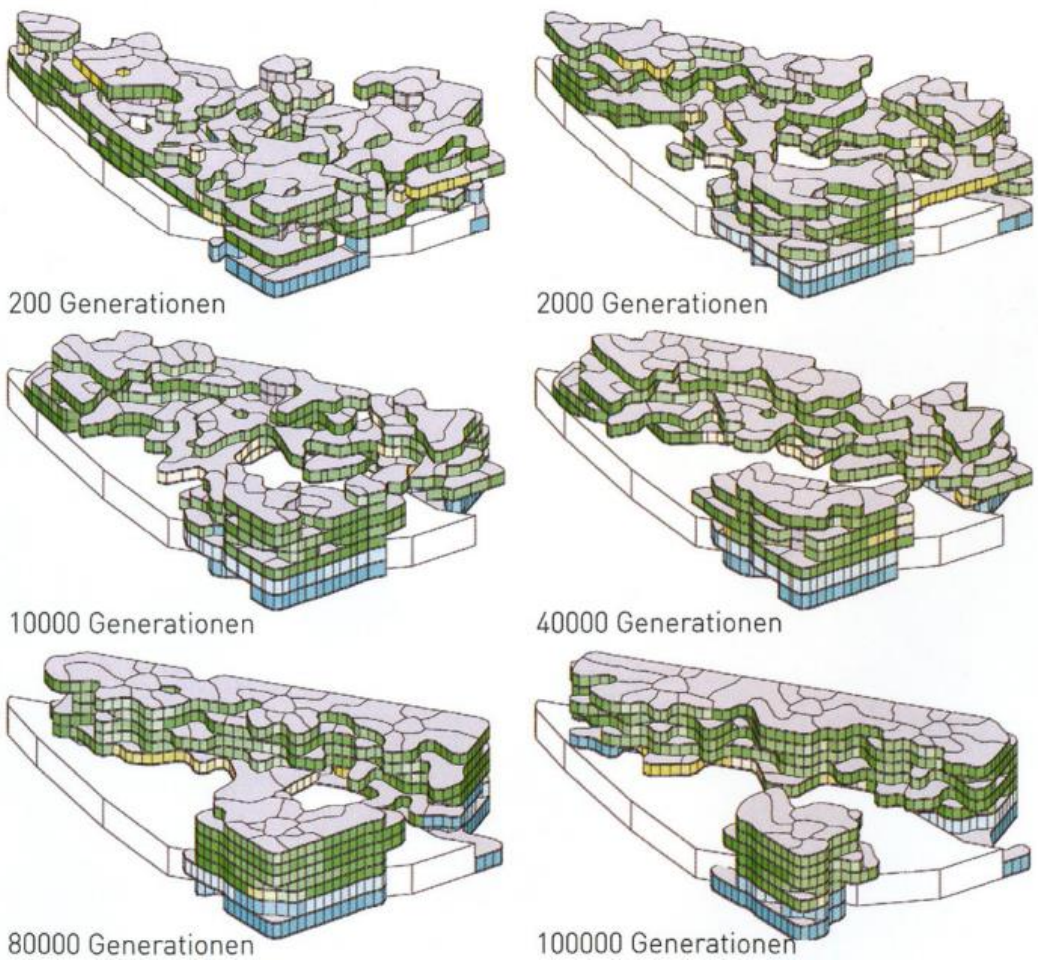


Figure 4 Optimization exapmple, in terms of density

Examples such as these, where optimization methods choose well-performing candidates from a highly constrained set of solutions, raise the question if optimization is necessary or meaningful as a design method for full-fledged building designs.

More often, researchers and practitioners apply optimization methods to one or more specific dimensions of architectural design problems, such as energy consumption or structural weight, or to specific building components, such as the building envelope. Evins

(2013) identifies six areas of applications of ADO to sustainable building design: envelope, form, HVAC systems, renewable energy, controls, and lighting

The architects interviewed by (Cichocka et al. 2017) were interested in optimizing structure (21%), daylight availability (19%), massing in terms of building codes and regulations (19%), circulation (12%), layouts (12%), and views (11%). The ADO literature contains several examples of floorplan and site layout optimization (Damski and Gero 1997; Jagielski and Gero 1997; Yeh 2006; El Ansary and Shalaby 2014) and at least on example of maximizing building volume relative to daylighting regulations (Pasternak 2016).

In structural design, optimization can be applied to the topology of a structure, the shape of a structure with a fixed topology, and the sizing of the individual elements of a structure with a fixed shape (Kicinger et al. 2005; Hare et al. 2013).

ii. Practical Applications of Optimization in Architectural Design

Lui (2015) documents an early example from the 1960s: the development and application of a "Building Optimization Program" at architecture and engineering firm Skidmore, Owings & Merrill LLP (SOM). Lui's account presents a textbook example of the pitfalls of formulating the holistic design of buildings as optimization problems: The program resulted in "cost-effective commercial architecture" that, in one case, "resulted in almost a caricature of the mundane office building." Most likely, this program employed linear programming, a commonly used optimization technique for problems that are formulated as linear functions. Luebkeman and Shea (2005) describe practical and experimental applications of black-box optimization at the Foresight Innovation and Incubation group of multidisciplinary design consultancy ARUP. They describe the minimization of the number of bracing elements for the Bishopsgate Tower in London and the number of members in a stadium roof. Other examples consider the panelization and rationalization of curved surfaces, with the objective of using only flat and ideally repeating panels, and the Pareto optimization of a building envelope in terms of energy and daylight. Also at ARUP, Hladik and Lewis (2010) document the optimization of the angles of the large louvers of Singapore's National Stadium in terms of shading and view. Binkley et al. (2014) provide a similar example: the application of a GA to the design of the roof of a multipurpose 70 sports hall and athletics stadium in Saudi Arabia. The algorithm optimized the clear height below the roof, as well as overall steel tonnage. Rüdenauer and Dohmen (2007) describe their use of a GA to optimize the weight of the timber structure of a mountain shelter on Switzerland's highest mountain, which reduced cost, waste, transport, and assembly time in a difficult to reach location. Scheurer (2007) describes a "proof of concept" developed in collaboration with structural engineering firm Bollinger+Grohmann that revolved around using a GA to optimize the shape of a large roof structure. In addition to the efficiency of the optimized solutions, Scheurer emphasizes their novelty: "Not one of the engineers on the project, with an impressive amount of experience between them, would have come up with the same engineering concepts that evolved from the [genetic] algorithm." Currently, Bollinger+Grohmann regularly employ multi-objective, Pareto-based GAs to generate efficient structures "with an aesthetic logic between order and disorder" (Heimrath 2017). Similarly, structural engineering firm Web Structures employs both single- and multi-

objective GAs to optimize architectural forms in terms of structural performance (Bamford 2018). Besserud et al. (2013) document the use of gradient-based and black-box optimization algorithms for integrated structural and architectural design at SOM. Besserud (2015) discusses the use of GAs at SOM for structural, solar, and daylighting optimization. Imbert et al. (2013) mention the "novel iterative approach to structural optimization" developed for the design of the Louvre Abu Dhabi that achieved "a fine balance of … structure self-weight, aesthetics, cost and buildability" (Figure 5)

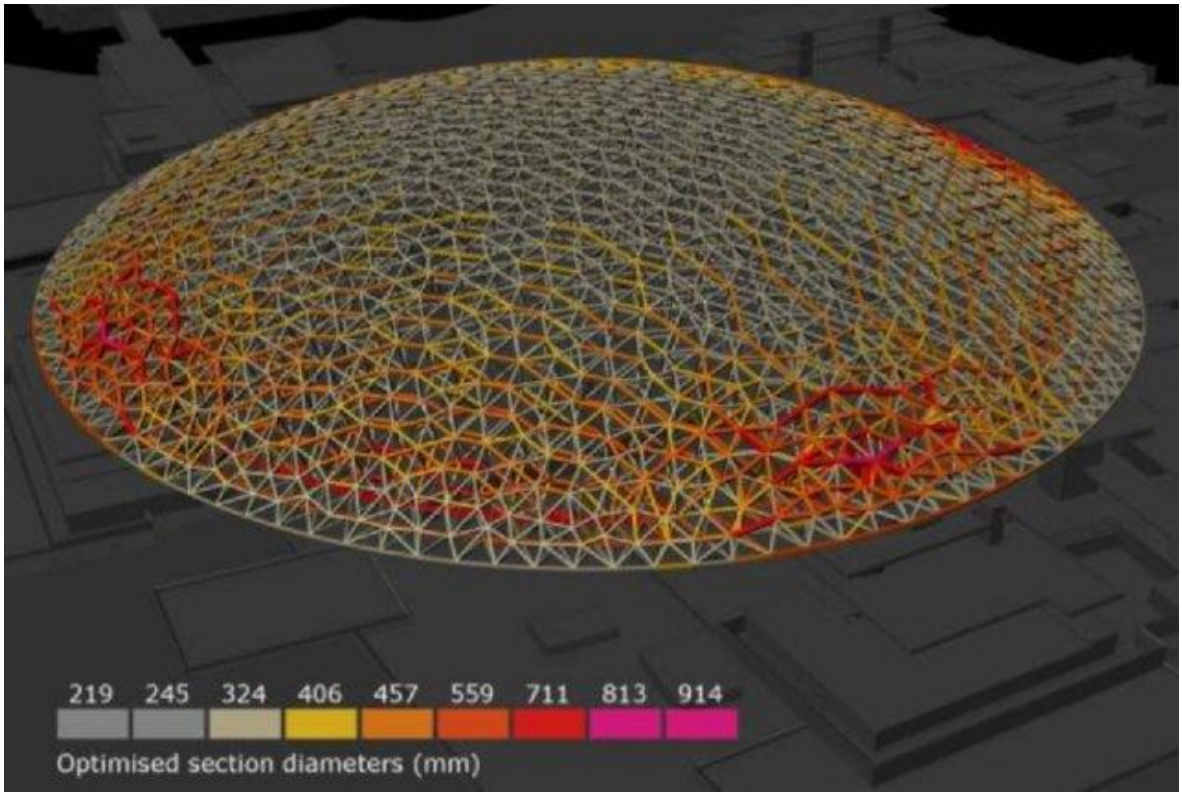


Figure 5 Louvre Abu Dhabi

The above examples illustrate the popularity of GAs for ADO and clarify that performance criteria in architectural design practice are often multiple and cover several disciplines, including a building's geometry, structure, and environmental design. This multidisciplinarity motivates studies that consider multidisciplinary, multi-objective optimization in the context of ADO.

5. Overview on classic method of structural opt. (Size, Shape, topology)

Structural optimization has over the past decades qualified as an important tool in the design process. The method can be grouped into topology, size and shape optimization. The objective of the optimization can be to minimize the stresses weight or compliance for a given amount of material and boundary conditions. The method can be utilized to design engineering structures but it can also be used to tailor mircostructures.
The most widely used numerical scheme for topology optimization is the Solid Isotropic Material with Penalization (SIMP) scheme where the density is approximated as constant within each element. The objective of the optimization is in the present work to find a design that maximizes the stiffness for a given amount of material. The advantage of using the stiffness as the objective, or rather its complement the compliance, is that it is a global measure and thus can be represented by a scalar value. Moreover, the constraint on the volume is also particular simple since it is linear and monotone which in most cases gives rise to a robust numerical algorithm. The SIMP procedure is based on a sequence of convex approximations and the algorithm is simple to implement and at the same time numerically efficient.

- *Sizing optimization*: This is when x is some type of structural thickness, cross-sectional areas of truss members, or the thickness distribution of a sheet. A sizing optimization problem for a truss structure is shown in Figure 6.

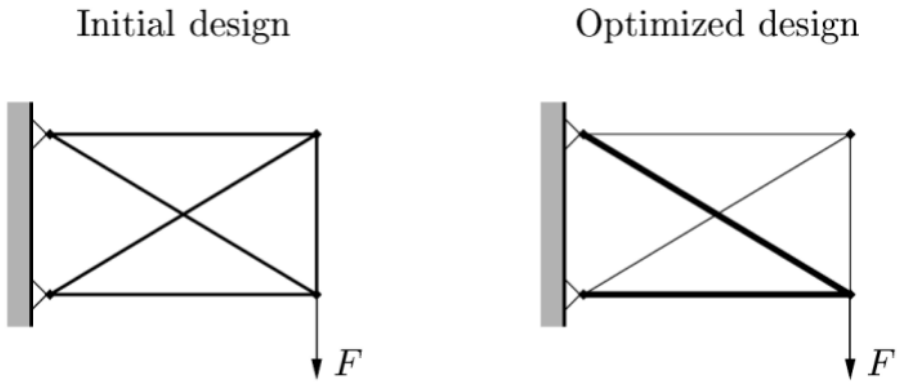


Figure 6 Sizing optmization of truss elements

- *Shape optimization*: In this case x represents the form or contour of some part of the boundary of the structural domain. Think of a solid body, the sate of which is described by a set of a partial differential equations. The optimization consists in choosing the integration domain for the differential equations in a an optimal way. Note that the connectivity of the structure is not changed by shape optimization: new boundaries are not formed. A two-dimensional shape optimization problem.

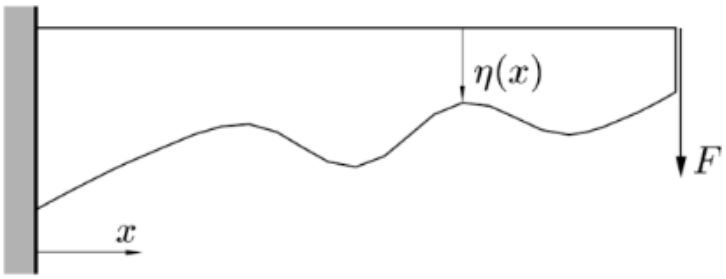


Figure 7 Shape optimization

- *Topology optimization*: This is the most general form of structural optimization. In a discrete case, such as for a truss, it is achieved by taking cross-sectional areas of truss members as design variables, and then allowing these variables to take the values zero, bars are removed from the truss. In this way the connectivity of nodes is variable so we may say that the topology of the truss changes, see (figure 8) If instead of a discrete structure we think of a continuum-type structure such as a two dimensional sheet, then topology changes can be achieved by letting the thickness of the sheet take the value zero. If pure topological features are optimized, the optimal thickness should take only two values: 0 and a fixed maximum sheet thickness. In a three-dimensional case the same effect can be achieved by letting x be a density-like variable that can only take the values 0 and 1. (Figure 8) shows an example of topology optimization.

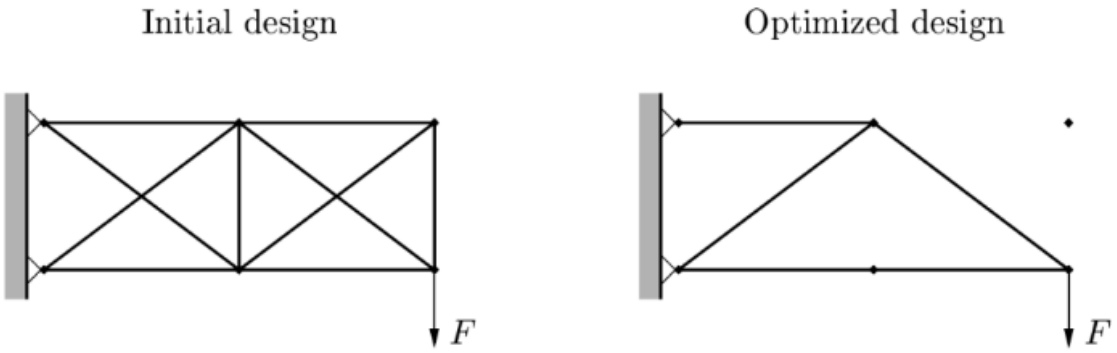


Figure 8 Topology optimization

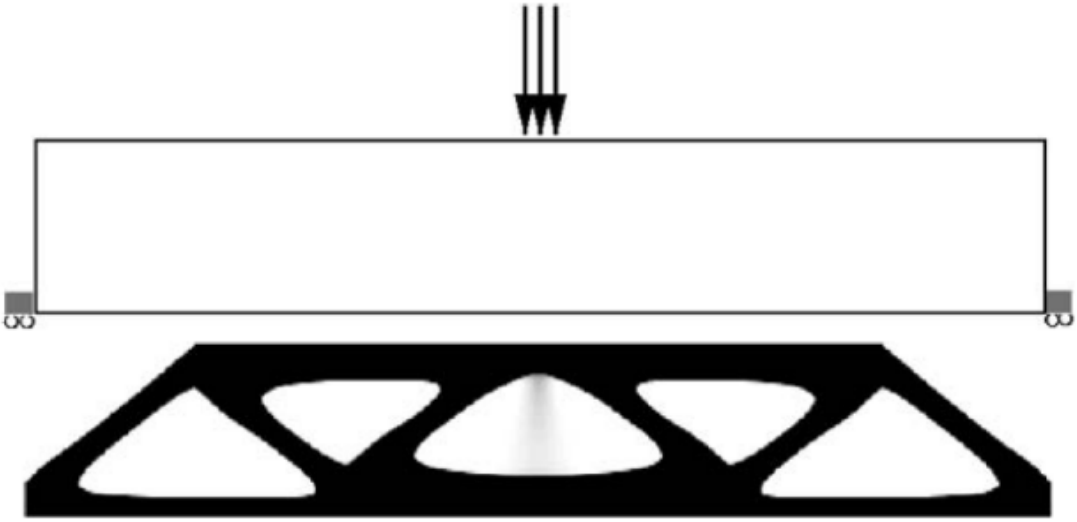


Figure 9 Topology optimization of the element in two supports

Ideally, shape optimization is a subclass of topology optimization, but practical implementations are based on very different techniques, so the two types are treated separately in this text and elsewhere. Concerning the relation between topology and sizing optimization, the situation is the opposite: from a fundamental point of view they are very different, but they are closely related from practical considerations.

When the state problem is a differential equation, we can say that shape optimization concerns control of the domain of the equation, while sizing and topology optimization concern control of its parameters.

I. Connecting structural optimization with generative design

Parametric design refers to a form of modelling in which relationships between various parameters, such as the shape, dimensions, and positioning of objects, are specified, with the advantage that the designer can quickly adjust some, and the rest of the model will act accordingly. The readjustment generated in the model from the user's changes is carried out by the software itself based on the rules previously established by the designer. These permutations must be subsequently evaluated to verify that they meet their objective. Unfortunately, the software traditionally used in the BIM methodology offers little flexibility in terms of exploring design alternatives at the preliminary project stage, since the parameterization they offer is reduced to changing the dimensions and characteristics of pre-established elements in the program's libraries, such as walls, windows, columns, and stairs.

This is how that generative design arises, which allows designers and engineers to define parameters such as materials, spatial constraints, manufacturing methods or cost limitations, to create rule sets or algorithms and thus automatically explore various permutations of the model, where the software generates the best design alternatives

according to the previously proposed objectives. Therefore, in parametric design, it is the user who can easily modify the geometry of the model (or the desired variable) to evaluate these variations later. In contrast, in generative design, it is the software that takes the inputs, evaluates them, and thus creates alternatives that best meet the requirements proposed by the user.

One of the disadvantages of developing a generative design code is that it involves investing time and work on the part of the company or the user. Although on the other hand, it should be considered that the more precise and complete this tool becomes, the greater the time savings in future operational processes that can be solved with such a code. Generative processes then emerge to accelerate the early stages of design.

In general, the use of these tools is associated only with geometry, although in engineering, a generative model comprises a set of rules and physical characteristics, given for example, by the materials, which must be characterized by their mechanical properties. These parameters serve to describe specific ranges, limits, and dispositions. Then, depending on the problem to be solved, one or other parameters (or combinations of them) will be used. This is an interesting tool to combine variables that at first sight are not so clearly related to each other (such as the limits of a building and solar radiation) and thus build a variety of approximate solutions.

Since civil engineering is responsible for feasibility studies, design, management, inspection and construction of works, operation, and maintenance of structures; frequently works with many of the parameters mentioned above. These depend on the branch or subdiscipline being studied since, in each one, different characteristics, behaviours, and properties of certain elements are analysed.

In order to bring generative design closer to civil engineering, this study was delimited to structural engineering (where it has had limited use), as it is more in touch with architecture, an area which, as it has been deepened, has varied experiences with this process. Therefore, the objective of this work is to compile experiences of the application of generative design in structural engineering, together with an analysis of its respective advantages and feasibility of implementation. Further, it is clearly shown how this variables inside Generative Design method are used for Structural Optimization.

Problem formulation refers to determining the three fundamental components of an optimization problem in the problem search space, namely the design variables which is described above through GD, objective function(s), and constraints. When conducting structural optimization, it is presupposed some freedom to change the attributes of the structure. The parameters used to represent the change of these attributes are usually called design variables and these variables can only be introduced in PD and GD. Design variable can be divided into two categories according to its value, namely continuous design variable and discrete design variable. The values of continuous design variables fluctuate within a certain range, while discrete design variables only have isolated values. Objective function

refers to a function or a set of functions that can be used as a measure of the optimization result. Constraints refer to the safety and serviceability requirements that must be satisfied during the optimization process. According to the form of the expression, constraints can be divided into two categories, namely equality constraints and inequality constraints. They can be interconverted to satisfy the requirements of different optimization methods. For example, an equality constraint h(X) = 0 can be replaced by two inequality constraints h1(X) ≥ 0 and h2(X) ≤ 0. In addition, constraints could be combined into the objective function as penalty functions to convert the constrained objective function to an unconstrained one. The range of design variables is called search space or design space, which could be further divided into feasible domain and infeasible domain. Feasible domain contains design points that satisfy all of the constraints, while design points that violate at least one constraint constitute infeasible domain. The general form of an optimization problem can be defined as follows:

$$\frac{Minimize}{Maximize}: f(X);$$

$$Subject\ to: gi(X) \leq 0, i = 0,1,2,3 \dots \dots, m;$$

$$hj(X) = 0, j = 1,2,3, \dots \dots, p;$$

$$X \in S.$$

where X is usually a vector X = [x1, x2, x3, ..., xn] and represents the set of design variables, in which n is the number of design variables; f(X) is the objective function; $g_i(X)$ and $h_j(X)$ refer to inequality and equality constraints; m and p are the number of constraints; and S is the search space of the optimization problem.

As mentioned above, there are four different types of objectives in structural optimization. Therefore, the problem formulation must be discussed according to the type of objective. Defining the objective function refers to finding a quantification of the desired result for an optimization problem while satisfying some requirements. Therefore, the parameter representing the objective function is sometimes different from the optimization objective. For example, cost minimization, which is the most commonly adopted objective in structural optimization, is usually quantified as the total weight of the structure to set up the objective function. As a result, the optimal design is achieved by minimizing the total weight of the structure. However, using weight to represent cost is often criticized by structural designers because a structure design with minimum weight does not necessarily lead to the minimum cost. Therefore, some objective functions are proposed to deal with the minimization of the cost, but only a small fraction of articles in the field of civil engineering structural optimization focus on this topic because of the uncertainties and fuzziness encountered. In terms of size optimization, the structure system is usually divided into several structural elements, and the cross-sectional areas are chosen as the design variables because the total weight of the structure is directly relevant to the cross-sectional properties of each structural element. Since the distribution of different materials is not considered in these studies, the objective function can be defined as:

$$Minimize: W = \sum_{i=1}^{n} \gamma g Ai Li$$

where W is the total weight of the structure; γ is the density of the material; g is the acceleration of gravity; the set of design variables X = {A1, A2, A3, ... , An} represents the cross-sectional areas of structural elements; and Li is the length of each structural member. For shape optimization, the nodal coordinates are used as the design variables. This type of structural optimization is often combined with size optimization for weight minimization. In terms of topology optimization, this type of structural optimization focuses on finding the optimal connectivity among nodes (joints), that is, determining whether there should be structural elements between the nodes or not. Topology optimization generally starts from a predefined dense structure with a lot of structural members, which is called the ground structure. In the optimization process, unnecessary elements are progressively eliminated and eventually the optimal design with minimized weight is obtained. Similarly, a vector is used as the set of topology variables. There are two values of these variables, namely 1 and 0. If the value of a topology variable is 1, the structural element represented by this variable can be removed, while 0 means the element cannot be removed. Structural topology optimization is also commonly combined with size optimization for structure weight minimization because the structural elements with very small cross-sectional areas are regarded as unnecessary and can be removed. In structural optimization with the goal of cost minimization, stress and displacement constraints are usually adopted, and the specific design requirements depend on the regional specifications applied instead of the type of optimization. Some most commonly used regional specifications include the ACI Codes for Concrete, Eurocodes 2, AASHTO, and British Standards.

Another commonly adopted objective for structural optimization is improving structural performance. However, there is not a uniform parameter to quantify the structural performance. Many performance indexes such as stiffness, compliance, strain energy, and static displacement are used to construct the objective function in the collected literature. The reason may be that topology optimization leads to optimal structural size in principle, and can be further refined by size and/or shape optimization methods. In this type of structural optimization, compliance minimization is generally set as the objective function to maximize the stiffness of structures. The objective function can be expressed as:

$$Minimize: C = F^T \times u(x)$$

where C is the compliance of the structure; F represents the load vector applied on the structure; and u refers to the displacement vector. The constraints of structural optimization for structural performance improvement are more diverse than that for weight minimization because of the various design requirements of structural properties. For example, when considering the dynamic response of structures, natural frequency is always constrained to avoid destructive effects of dynamic loads. Based on the design requirements, many mechanical constraints such as displacement, stiffness, stress, and buckling loads are adopted

in this type of structural optimization. In addition, material weight or volume is often constrained to control the structural cost.

II. Optimization tool Galapagos

Galapagos is a component inside of grasshopper that can optimize a shape so that it best achieves a user defined goal. For this to work, Galapagos needs a series of options or genes to try out, and a defined goal or fitness value.

Galapagos is a tool created by David Rutten. What makes Galapagos special is that it does not try every single possible combination of these options in order to arrive at the optimum solution. It can do this accurately in much less time by "learning" from each successive round of experiments or generations and progressively closing in on the best answer. In practice this means the difference between a week long calculation and one that can be completed overnight. Up until now, all of our optimizations use Galapagos as the solver, however there are plenty of other tools out there. Goat and Octopus are two others that are available as components inside the Grasshopper environment.

An important part of parametric design is not only setting up a script, but also optimizing the model for different goals. For example, the position of a house can be optimized to receive as much daylight as possible, or the thickness of a construction can be reduced to the utmost.

The Design Problem

Imagine the following scenario: after years of experimenting and designing, you have found the ideal toy for architecture students. Now you want to send a package including your prototype to several friends. Unfortunately, the costs of sending the product, is based on the size of the box. In the following chapters we will discuss a method to optimize the volume of the box, based on the orientation of the toy. Furthermore, we will also discuss most settings of Galapagos.

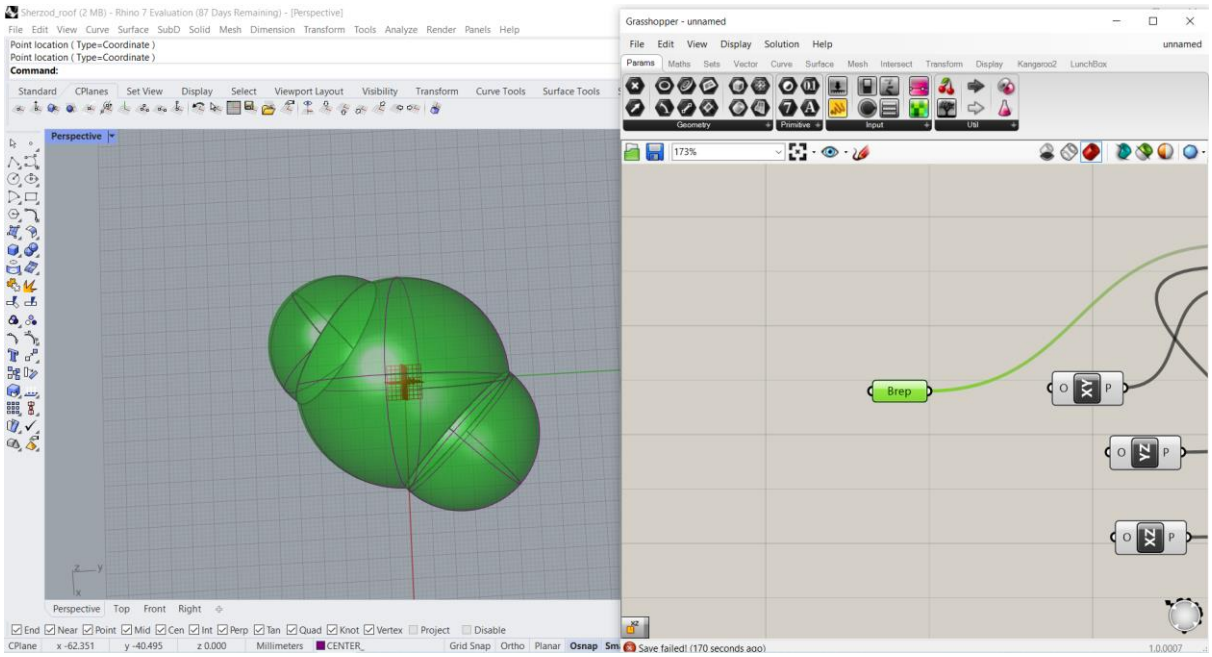


Figure 10 Example of optimization through Galapagos

Setting up the script:

First we create a model in Rhino. Then we set this model to a geometry parameter in Grasshopper.

Design a model in Rhino
Decide which transformations of the model in the box are allowed. In the case of this example, we will assume it is allowed to rotate the model in three directions: X, Y and Z. By connecting a XY, XZ and YZ plane to a Move component, we can do these translations.

Model must be rotated the in three directions:

Transform » Euclidian » Rotate

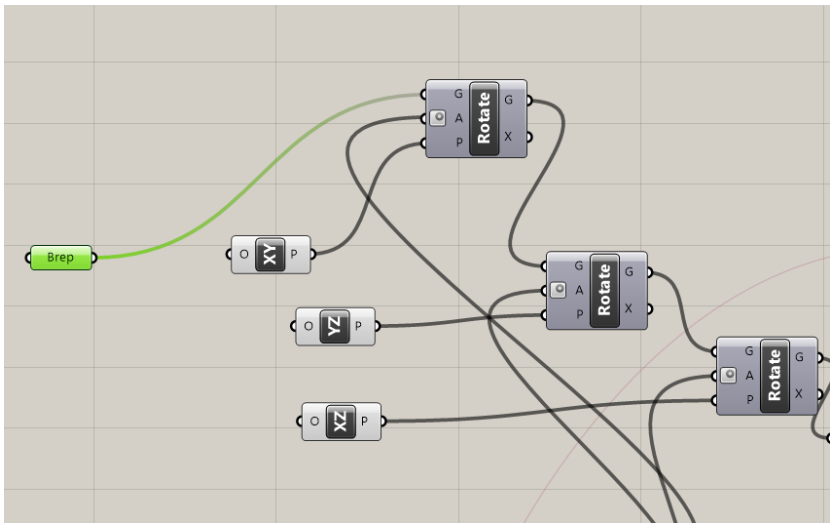


Figure 11 Initiation of procedure

The available values will be defined by a Gene Pool. By double-clicking on the Gene Pool, you can setup the sliders. This process works similar to a standard Number Slider. Set the Gene Count to 3 and the decimals to 0. Set the maximum to 359, since that is the maximum amount of degrees to rotate.

- Add a Gene Pool Params » Input » Gene Pool
- Change the settings of the Gene Pool RMB on Gene Pool » Edit

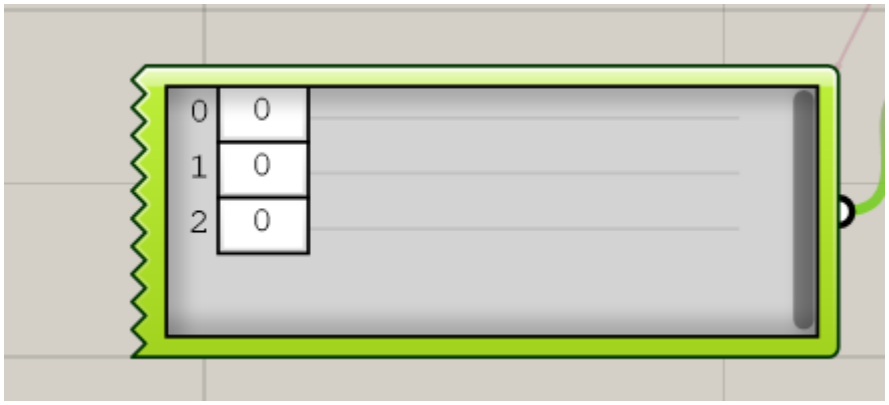


Figure 12 Gene pool

Extract the values using a List Item component. Zoom in on the List Item Component and click on the plus icon to extract item +1 and +2.

- Extract the Gene Pool parameters Sets » List » List Item
- Click on the + icon twice

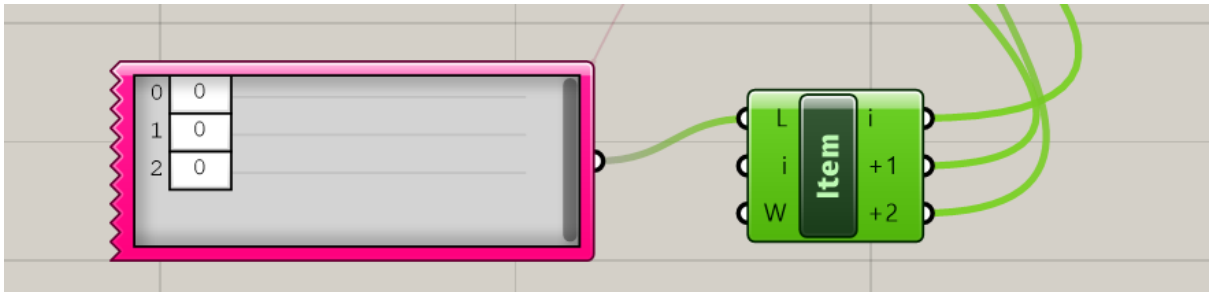


Figure 13 Extraction of items

Connect the outputs to the different Move components. Also set the Angle to degrees by right-clicking on the Angle input.

- Connect the outputs to the Angle inputs
- Set the Angle inputs to degrees RMB on Angle » Degrees

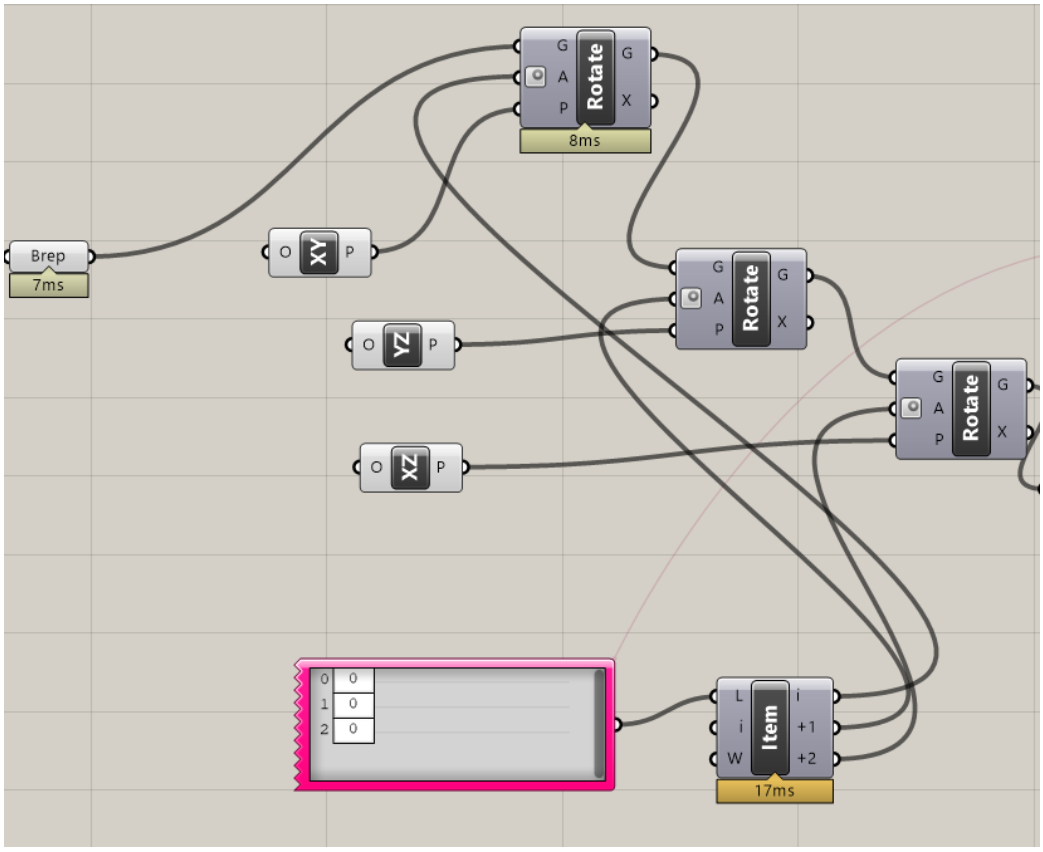


Figure 14 Rotation of model

Finish the script by adding a Geometry Parameter. This node indicates that this is the final result.

- Add a Geometry parameter Params » Geometry » Geometry

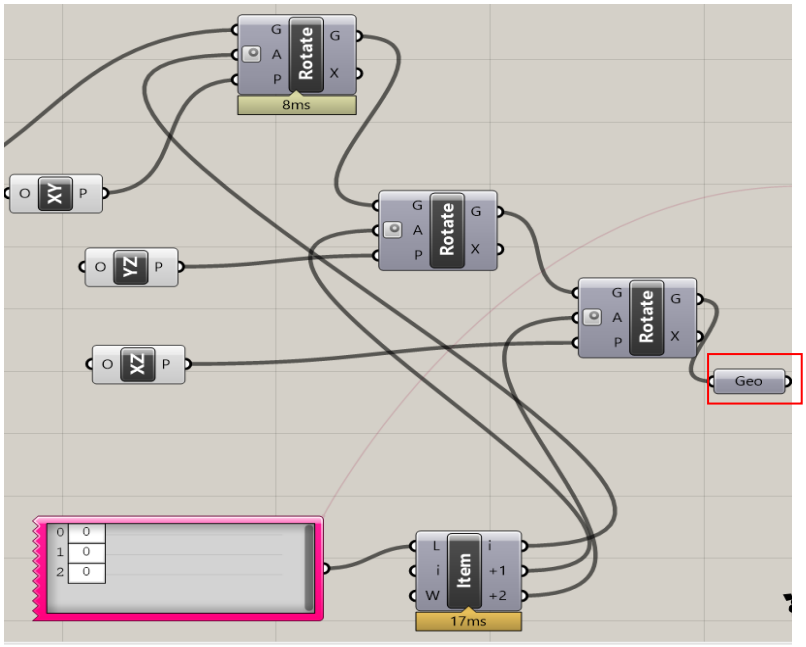


Figure 15 Links between new and old geometry

Now we need to create a box around the object. Use a Bounding box to find the smallest possible box in a certain orientation. If you are using multiple separated objects. Right-click on the Bounding Box component and click on Union Box.

- Create a bounding box around the model Surface » Primitive » Bounding Box

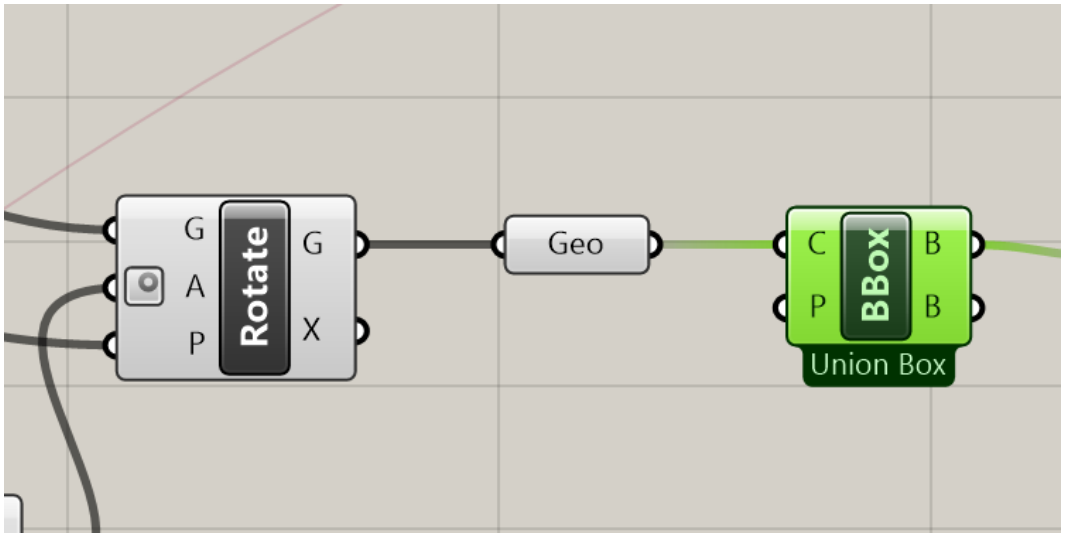


Figure 16 Creation of box around model

Finally we have to calculate the volume of the box. Add a volume component and connect a panel to the Volume output to see the details.

- Calculate the volume of the bounding box Surface » Analysis » Volume
- Connect a panel to the Volume output Params » Input » Panel

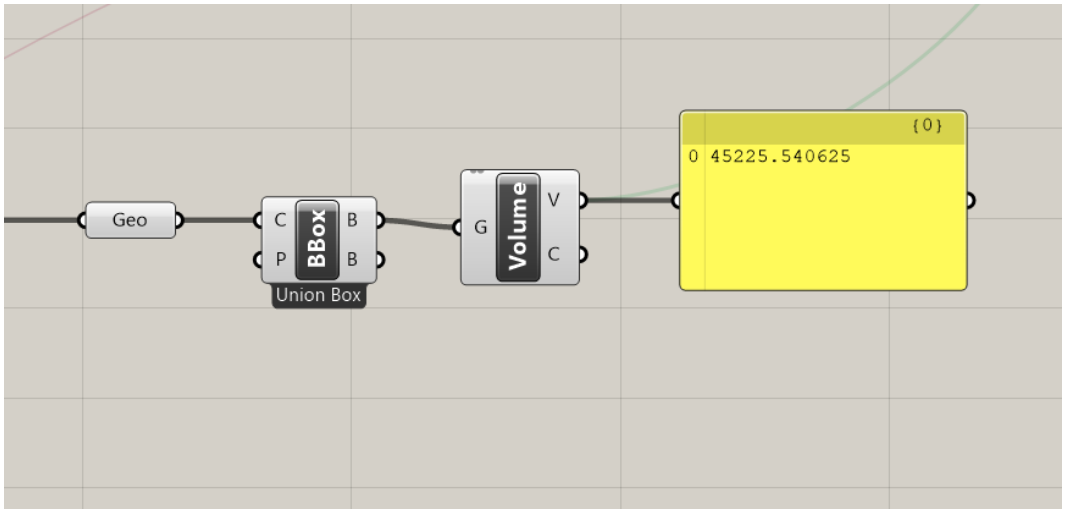


Figure 17 Creation of box

Initiating Galapagos

Just like other components, add the Galapagos node to the canvas.

- Add Galapagos to the canvas Params » Util » Galapagos

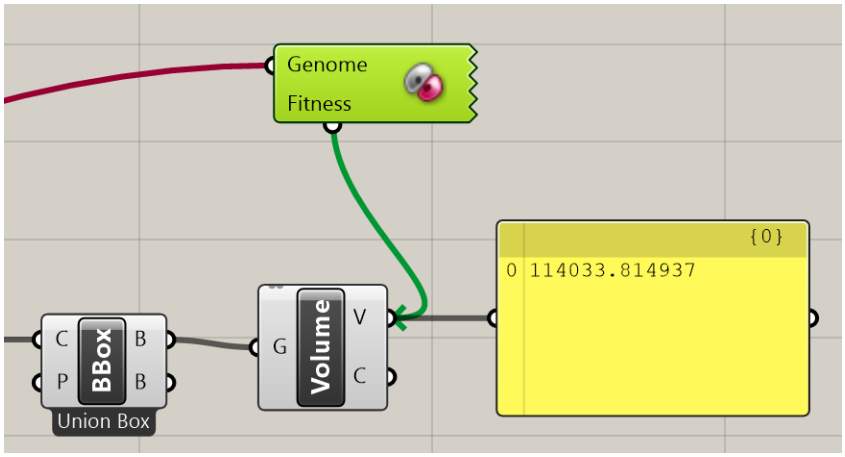


Figure 18 Galapagos links

Galapagos has two inputs: the Genome and the Fitness. A fitness value can be described as the value you want to optimize, in this case the volume. The Genome is a collection of parameters that influence the Fitness. For this tutorial: the rotation in the three different directions.

Since we have set our Gene Pool to 0 digits, each slider has 360 degrees of freeform. In total, this results in 360 * 360 * 360 = 46.656.000 possible outcomes to our script. It would be impossible to try all options ourselves. Therefore, we use Galapagos to find an approximation to the best solution fairly efficiently.

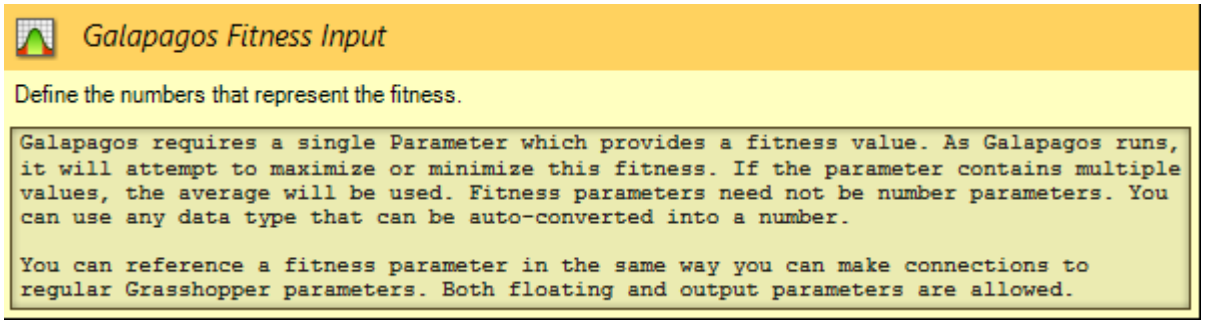


Figure 19 Fitness input

Connecting the wires to Galapagos works a bit different than normal components. You always have to connect the wire *from* the Galapagos input *to* the parameters. Connect the Genome to the Gene Pool and the Fitness to the Volume output.

- Connect the fitness to the Volume output
- Connect the Genome tot the Gene Pool

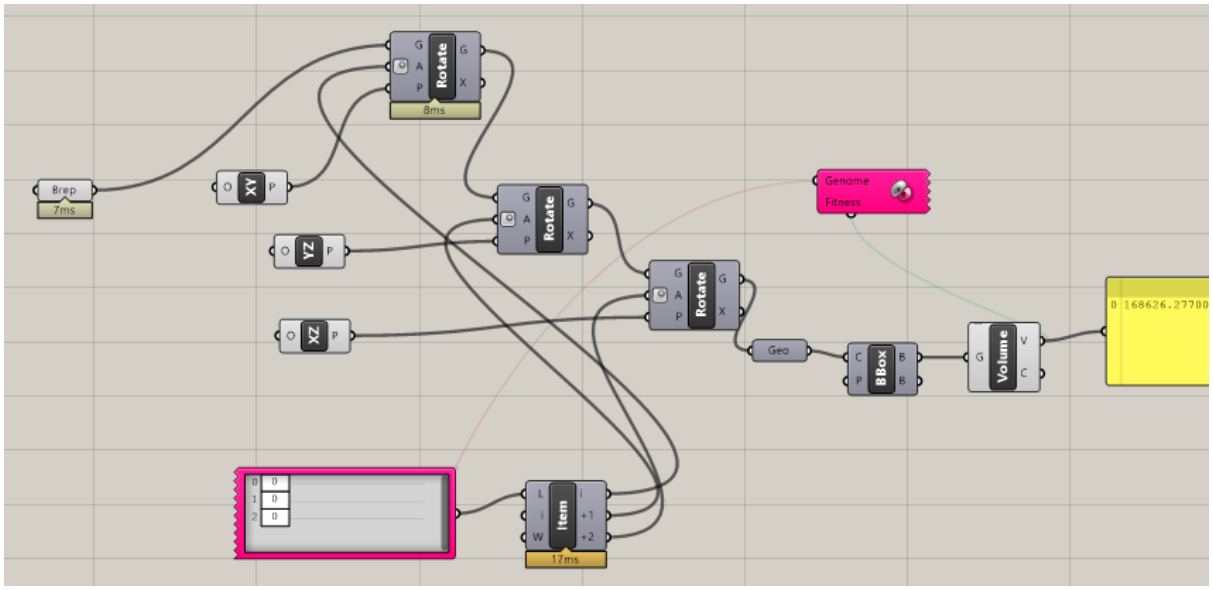


Figure 20 Overall links between commands

Double-Click on the Galapagos component. The Galapagos Editor is now opened.

- Open the Galapagos editor Select Galapagos node » Double Click on icon

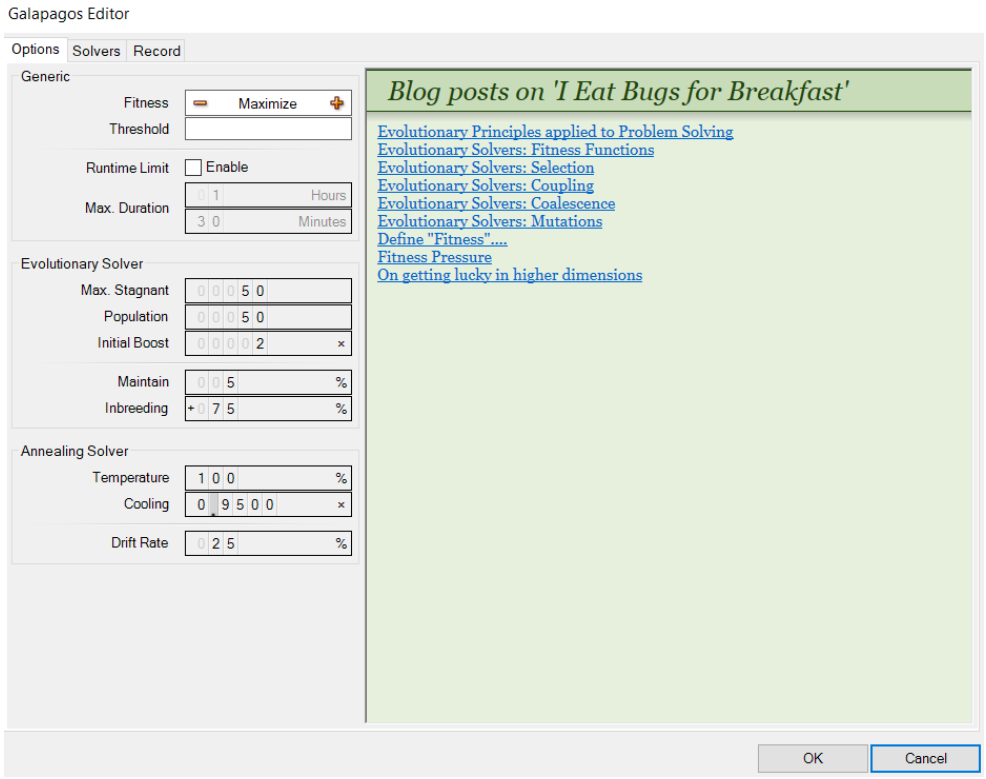


Figure 21 Optimization toll Galapagos options

Galapagos Initial Settings

Now, we will discuss different initial settings in Galapagos before running.

Fitness/threshold Defines if you want to maximize or minimize the value. By setting a threshold, you can define the fitness that should be found. If the algorithm finds a solution with this threshold, it will stop. If you leave the threshold empty, the algorithm will continue indefinitely, until it has tried all options, or reaches a time limit.

- Set the Fitness to minimize

Runtime Limit By setting a time limit, you can define the maximum time the model is allowed to run.

- Do not enable the Runtime Limit

Galapagos has two types of optimization algorithms: evolutionary and annealing. Generally speaking, it is advised to use the evolutionary solver. The evolutionary solver searches for a good result and then optimizes it making small changes to the parameter. However, in some cases, a script is built with a great degree of freedom which may lead to unexpected bad results in the evolutionary solver. Therefore you can use the annealing solver.

The following settings only apply to the evolutionary solver:

Max. stagnant The maximum amount of generations that do not lead to a more optimized result, before the solver should stop.

Population How many options the algorithm should try before it progresses to the next optimized result.

Initial Boost The initial boost can lead to better results in cases where:

- The model only has specific local parameter value combination that lead to great results
- The model has high degrees of freedom; and lots of possible combinations
- Leave the Initial Boost to 2 for now. Feel free to experiment with different values.

Maintain The amount of results that should be combined every generation to find new better results.

Inbreeding The freedom factor of the algorithm to use very similar or very different genes to breed with. A high positive factor means that the algorithm will only use genes with very similar results. A high negative factor states that the algorithm is allowed to use very different genes to combine.

The following settings only apply to the annealing solver:

Temperature The chance that an iteration jumps to a completely different combination of parameter values to check if it works better.

Cooling The factor to lower the temperature with at each jump to another combination of parameter values.

Drift rate The chance that a parameter value is changed. If the drift rate is 0%, only one parameter will be changed every iteration

Since we are using the evolutionary algorithm, we can leave the annealing settings as they are. Click on OK.

- Click on OK

Before starting the algorithm, make sure you turned off all previews of nodes that are not relevant. In our case, only enable the Geometry parameter node and the Bounding Box component. After that, open the Galapagos editor again and click on the solvers.

- Open the Galapagos Editor Select Galapagos node » Doubleclick on icon
- Click on the solvers tab

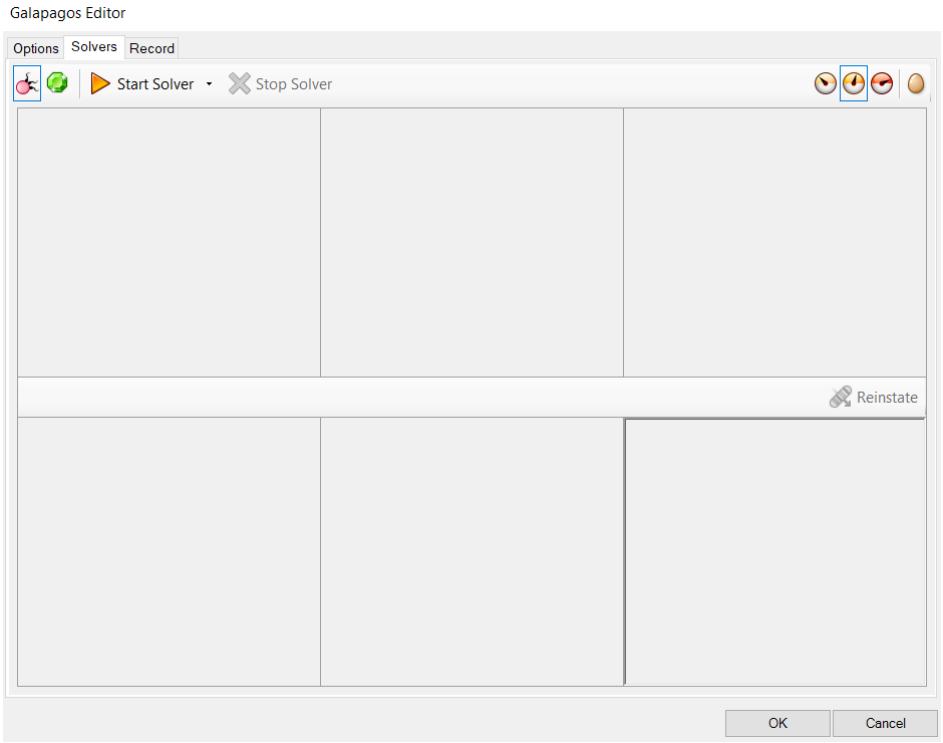


Figure 22 Window of solver in Galapagos

In the top shelf, you can specify several settings. The first two icons define the algorithm type you are using. Next to that, you can start the solver by clicking on Start Solver. At the end, there are some options to display intermediate results in the Rhino viewport. For fast and efficient calculations, it makes sense to disable the display. However, for this example, we will turn on the display of all genes, by clicking on the first clock icon. Finally click on OK.

- Turn on the display of all intermediate genes

Now click on Start Solver. You will see that Galapagos changes the parameter settings of your model and displays different results.

- Click on Start Solver
- Check your Rhino viewport

Interpreting the results

After running the algorithm for a while, you can Stop the Solver.

- Run the algorithm until the changes between iterations are minimized

The bottom right diagram shows some of the best solutions. Select the top result and click on reinstate. The result should now be visible in the viewport. If you are interested, you can also check other solutions.

- Select the top Gene
- Click on reinstate
- Open your Rhino viewport

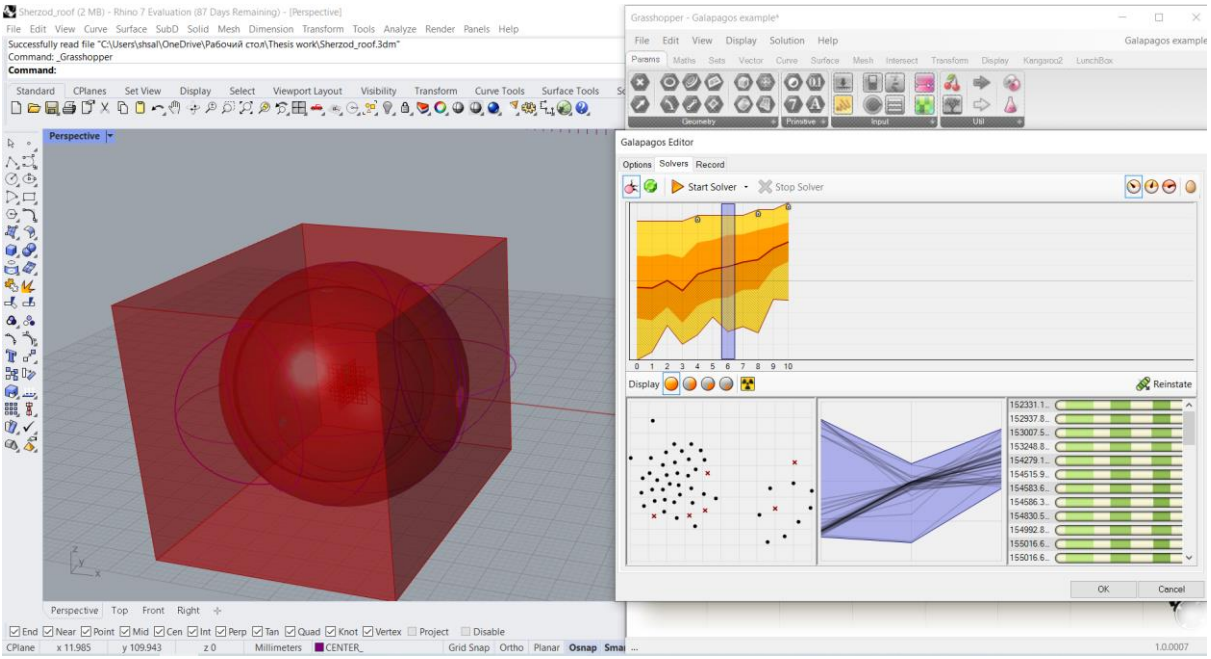


Figure 23 Optimization process

If you are still not satisfied with the solution. Select one of the genes and click on the little triangle next to the Start Solver button. By clicking on Start from selected Genome, you can continue the algorithm using your preferred gene.

III. Multi Objective Optimization tool Octopus

Octopus by Robert Vierlinger was originally made for Multi-Objective Evolutionary Optimization. It allows the search for many goals at once, producing a range of optimized trade-off solutions between the extremes of each goal. It is used and works similar to David Rutten's Galapagos, but introduces the Pareto-Principle for Multiple Goals.

Octopus now also includes:

- Evolutionary Breeding of Artificial Neural Networks with extended Basis Functions, based on CPPN-HyperNEAT

- Interactive Evolution – Selector Component
- When running a genetic evolutionary optimization, human decisions can be added as a decision- maker.
- Simple Supervised Learning with Backpropagation and Artificial Neural Networks
  To make a component map N numeric inputs to M numeric outputs, based on examples it was shown before.
- Supervised Learning with a Support Vector Machine (SVM)
  To make a component map N numeric inputs to 1 numeric output, based on examples it was shown before.
- Octopus Explicit Components
  To build a genetic algorithm from its basic functions; allowing many different flavors of the way things are handled in the optimization.

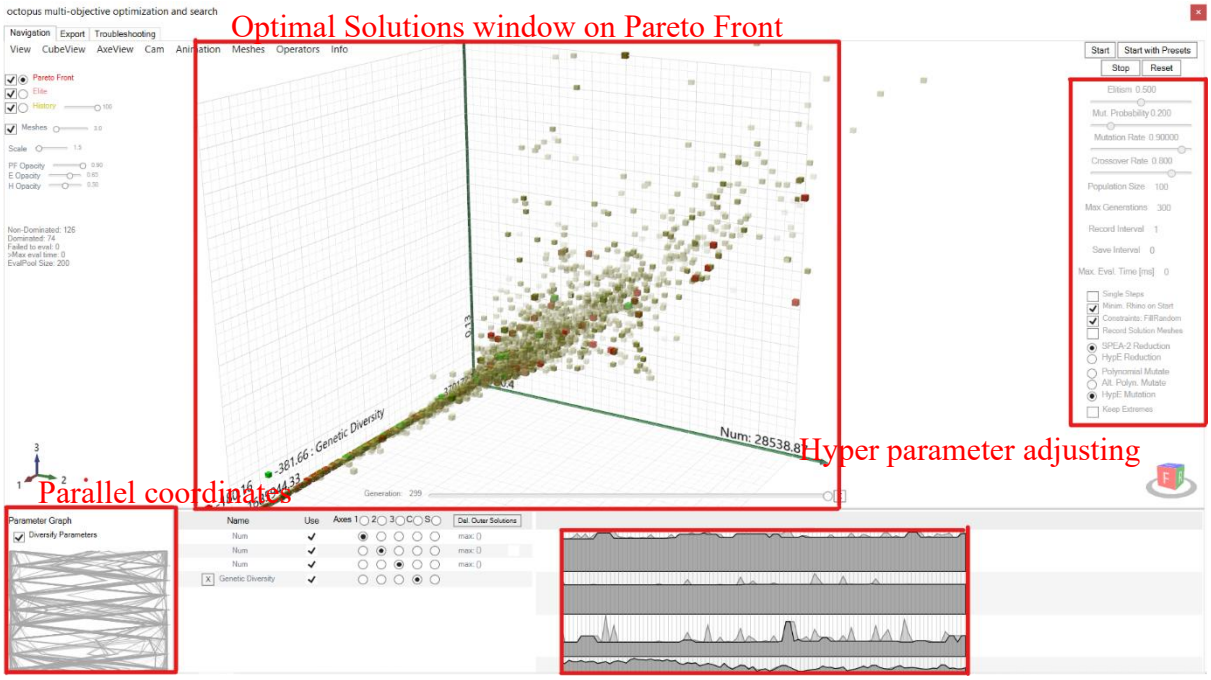


Figure 24 Octopus user Interface

At the left-bottom corner, the parallel coordinate diagram shows different combinations of variable values. As the # Generations increases, the paths will change from diverse to more concentrated to the optimal solutions. At the right-bottom corner, the minimum objective values from each generation are recorded. They will tend to be lower as the # Generations increases.

IV. FEA tool Karamba3D

Karamba3D is a parametric structural engineering tool which provides accurate analysis of spatial trusses, frames and shells.
Karamba3D is fully embedded in the parametric environment of Grasshopper which is a plug-in for the 3d modelling tool Rhinoceros (Rhino3d). This makes it easy to combine parameterized geometric models, finite element calculations and optimization algorithms like Octopus or Galapagos.

Karamba3D is a Finite Element program like many others. However it has advantages over these in several important respects: It is easy to use for non-experts, has been tailored to the needs of architects and engineers in the early design phase, works interactively and costs slightly less than the rest.

6. Literature review on Gridshell

Gridshells have an interesting history. The name "gridshell," particularly in Europe, became a way to describe free form combinations of dome-type curvature and inverse curvature in a single lattice. But, at first, gridshells had either positive curvature, as geodesic domes, or negative curvature as hyperboloids. In the closing decade of the 19th century, Russian engineer Vladimir Shukhov built tall towers with criss-crossing straight line generators that formed hyperboloids of revolution (figure 10). In the late 1920s, Walther Bauersfeld created the first geodesic dome as formwork for a planetarium in Germany, and subsequently, Buckminster Fuller started popularizing this form in the 1950s.

Figure 25 Tower by Vladimir Shukhov (one of the first gridshell structure)

In February of 1965, Dr. Douglas Wright (Founding director of Geometrica) published a seminal paper on the design of gridshells, *Membrane Forces and Buckling in Reticulated Shells* in the Journal of the Structural Division of the American Society of Civil Engineers. It explained how these beautiful structures could be engineered based on sound principles of mechanics, even before computers were sufficiently powerful or available. It became the Structural Division's most discussed paper to its date, and enabled the rational use of this structural form.

Grid shells are basically shells where material has been removed to create a grid pattern. Where in a plain shell an infinite number of load paths were available, in a grid shell the internal forces are carried by members and therefore have to follow a restricted number of paths.

Apart from only being discrete shell structures that advantageously benefit from their geometry to become self standing, the grid shells are characterized by their innovative erection scheme. By tacking only the exact definition of a grid shell, any 3 geodesic dome or reticulated surface could be called a grid shell. The powerful concept that lies behind structures referred to as grid shells, is that the construction starts from a flat surface. All the members of the structure can be assembled flat on the ground so as to form a two-dimensional articulated mat. The final structure will then be obtained by pushing, pulling and deforming the mat, and this being done without introducing any additional connection or structural member. Once in place, the surface having quite a small radius of curvature, the continuous members will act as arches. The final structure will thus benefit from the efficiency of both shell and arch scheme. In the same way as in a shell, the members of the grid shell experience only tension or compression.

There are several examples of pavilions or artistic installation created with this kind of structure, thanks to its lightweight and resistance. An interesting light structure is the Observation Tower in the Helsinki Zoo, that has a particular curved shape similar to a “Bubble”.

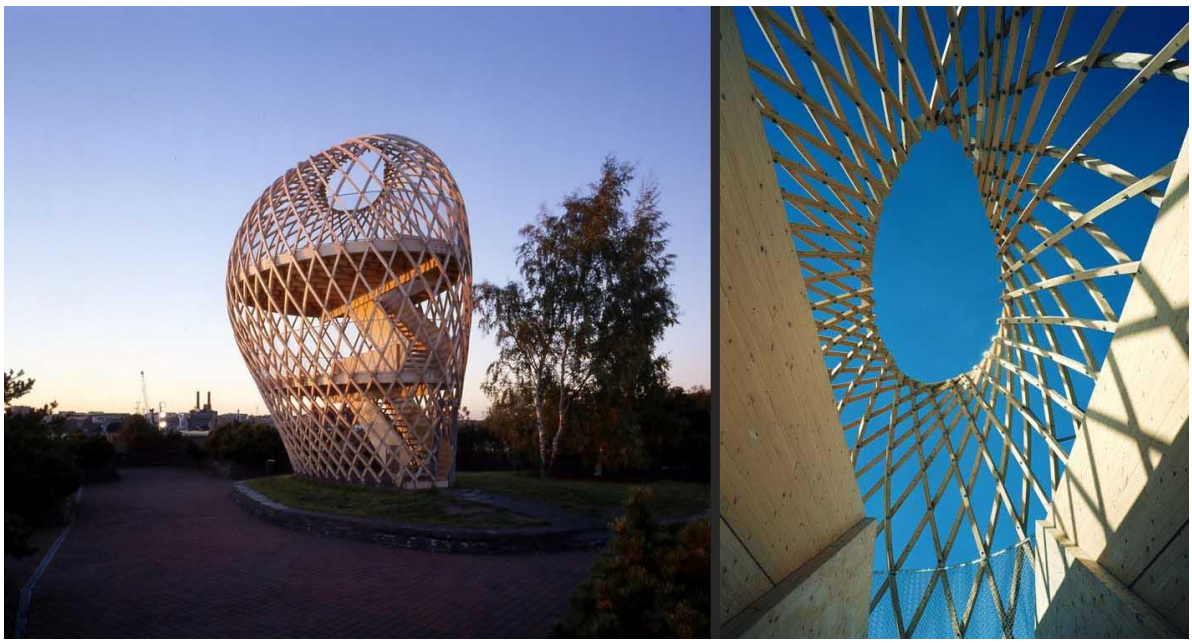


Figure 26 Gridshell structure in Helsinki Zoo

The Centre Pompidou-Metz by Shigeru Ban Architects is a perfect example of how gridshell can be used in a complex building, creating indoors spaces with different volumes. In this project the roof and the interiors are organized in a dynamic composition, with different sections and glazed walls.

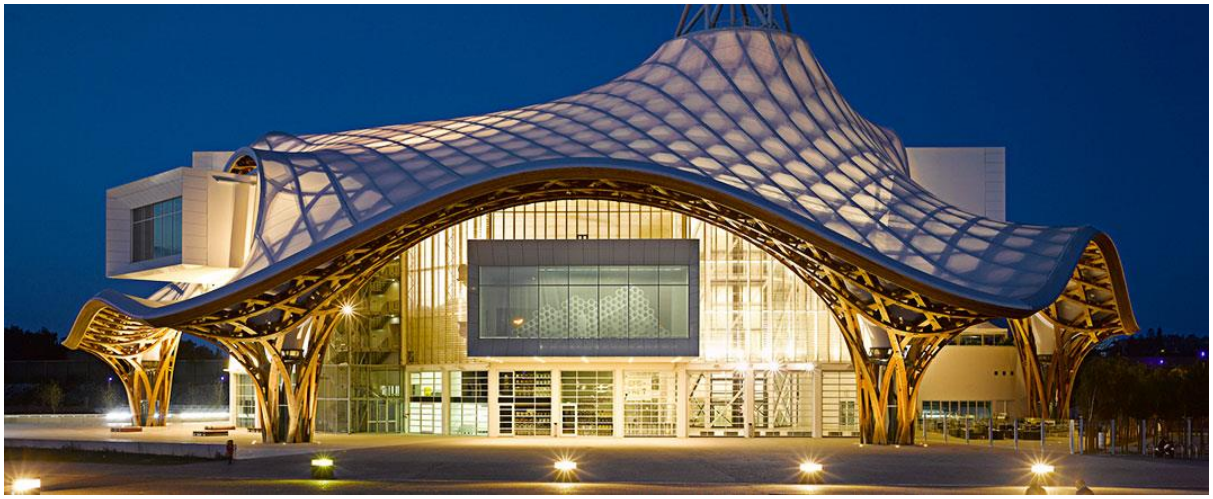


Figure 27 Centre Pompidou-Metz by Shigeru Ban Architects

How a gridshell works

Gridshells mainly sustain the loads and got good strength capabilities due to it's double curvature. A shell is a three dimensional structure that resists applied loads through its inherent shape. If regular holes are made in the shell, with the removed material concentrated into the remaining strips, the resulting structure is a gridshell. The three dimensional structural stability is maintained by shear stiffness in the plane of the shell, achieved by preventing rotation at the nodes or by introducing bracing. For the construction of gridshells, it's most often used wood or steel.

Timber has low torsional stiffness and timber gridshells can be made by laying out a lattice as a flat mat and then pushed into shape. The very long timbers needed to make timber gridshells are fabricated by splicing shorter, defect-free pieces together. The word "lath" has been coined for these long timbers. During forming, the timber lattice must allow rotation at the nodes and bending and twisting of its constituent laths. Once formed, shell action is accomplished by bracing, which triangulates the structure and provides in-plane shear strength.

As mentioned above, gridshells are shell structures consisting of one dimensional elements of continuous surface. These elements can mainly be of two kinds. The first is when the structure consists of relatively short (discrete) beams or bars which start and end between the nodes of the grid. This is the most common strategy when steel is chosen as the structural material. The other way of constructing a gridshell is to use continuous members spanning between the supports. These are initially flat and bent into their three dimensional desired shape by utilizing the material's bending capacity.

In terms of production, the elements of gridshell structures with discrete members between nodes (such as steel gridshells), must be fabricated in correct lengths and mounted between their two corresponding nodes. The geometry of these connections can be very complex, with various number of connecting members in different angles in 3D space making the production costs high. Also the assembly process usually requires some sort of false-work to support the shell while cantilevering until completion. Gridshells with continuous members on the other hand are especially interesting if the manufacturing and construction process is considered. When the shell is constructed by slender continuous members that are elastically bent into position, a network of flat elements can be assembled on ground prior to the erection of the structure. The connections do not need to be uniquely produced, but can be mass manufactured and added to the lattice. Mechanically, a number of criteria for the nodes connecting the members must be fulfilled to make the flat grid deformable:

• The nodes lock the connecting members in translation.
• The continuous members must be able to rotate relative to each other (an extra rotational degree of freedom in each node is required).
• The connection must not substantially reduce the members load bearing capacity (by penetrating the material for instance).

To turn a directional flat grid into a doubly curved surface without changing the node position along the members is therefore only possible if the joints are rotatable. For a bidirectional grid with members arranged orthogonally, the initial flat configuration forms squares between the laths. When a Gaussian curvature κ 6= 0 is introduced these squares shear into rhombuses. The material of choice must have a low ratio between its Young's modulus and bending capacity. That means it must be possible to deform the material extensively without yielding or breaking. Since the laths seldom follow the geodesic curves on the surface, another requirement of the material is low torsional stiffness since they must rotate along their longitudinal axis.

When the lattice has been erected to its desired shape, the freely rotating nodes must be constrained in order to make transmission of forces between the members possible. In its initial pinned condition, the grid can transmit forces in the direction of the laths and by out of plane bending, but not in-plane shear. Diagonal stiffness must be added, which can be modelled and achieved in various ways:

• By stiffening the nodes by making them moment resistant.
• Provide diagonal bracing, either by adding cable elements or struts.
• Applying a continuous shear stiff cladding onto the grid.

The preferred bracing method is a choice influenced by a number of factors such as design intent, load conditions and given erection method.

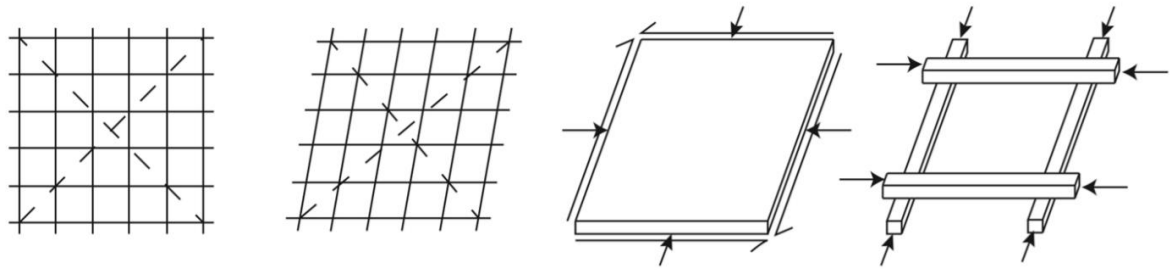


Figure 28 To the left: lattice distortions. To the right: Shear stiffness is provided in a continuous shell element but not for a set of grid shell elements with rotatable joints

7. Case study: Long span roof gridshell

The object under study is the steel structure with long span roof. Aim of this test case is to perform Finite Element analysis to find optimum topology for the roof. All the necessary data about the structure were given in .dwg and in Revit files. Preliminary dimensions of the structural elements were extracted from the provided data. The roof is composed of curved longitudinal steel sections and straight transverse beams. Gridshell structure was used as a main structural element for the roof. The optimization process was carried out to find best topology for Gridshell roof, because as it is obvious, the position of columns, number of divisions of grid in two directions can vary thousands of times. To perform accurate optimization process, model was created in computer-aided design application software Rhino, because later using the Grasshopper plugin for Rhino, it was possible to achieve the optimization.

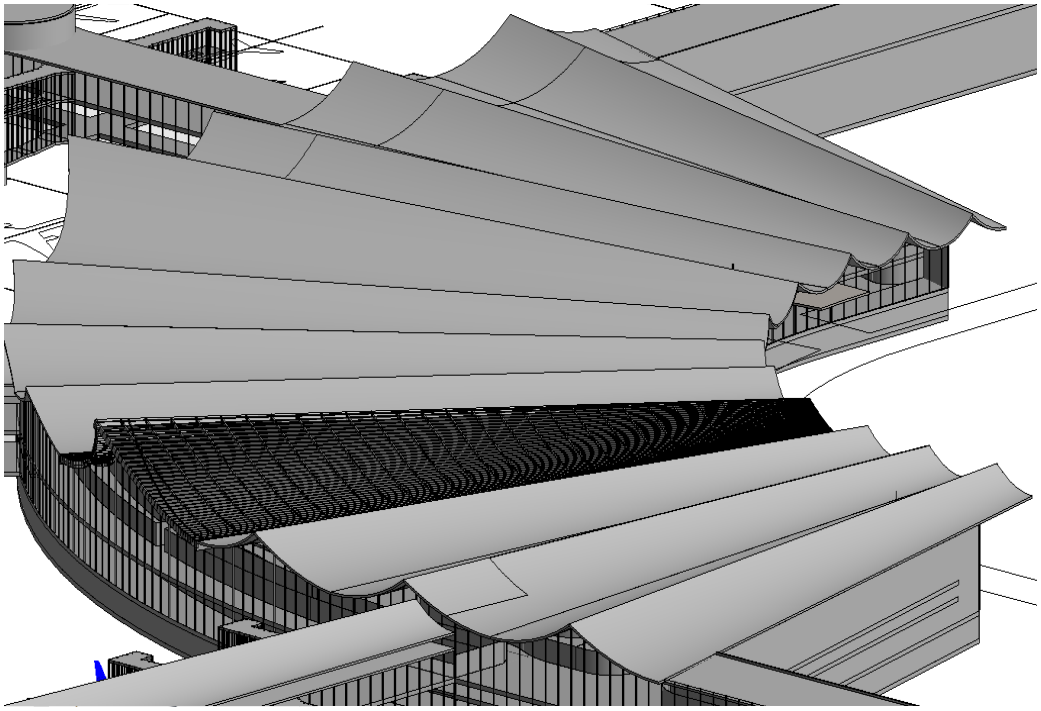


Figure 29 Model extracted from Revit data

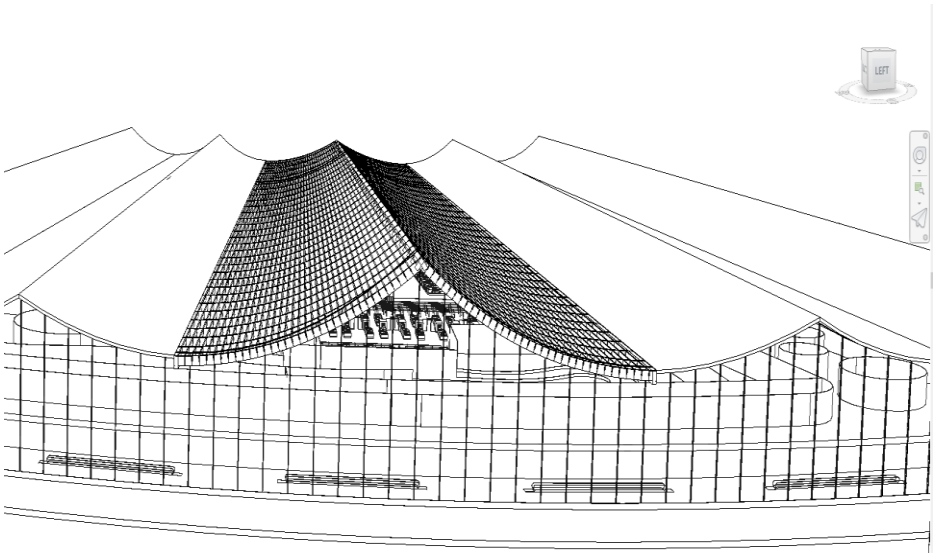


Figure 30 Left view

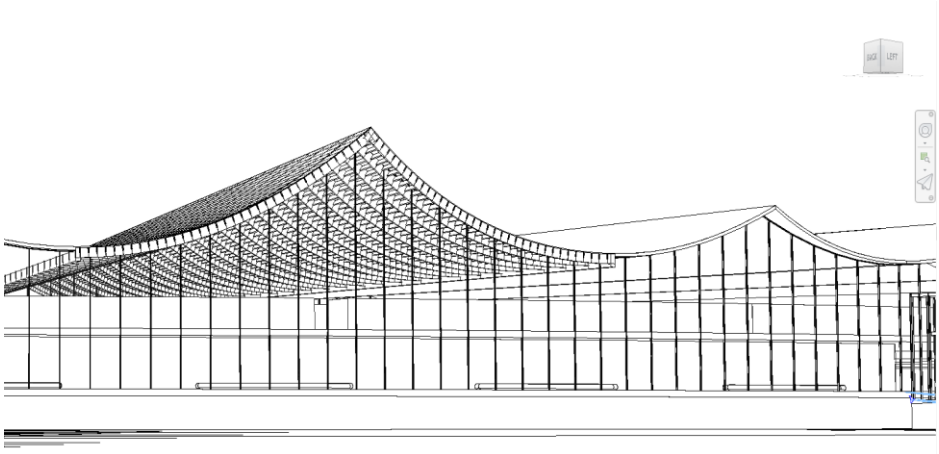


Figure 31 Left view

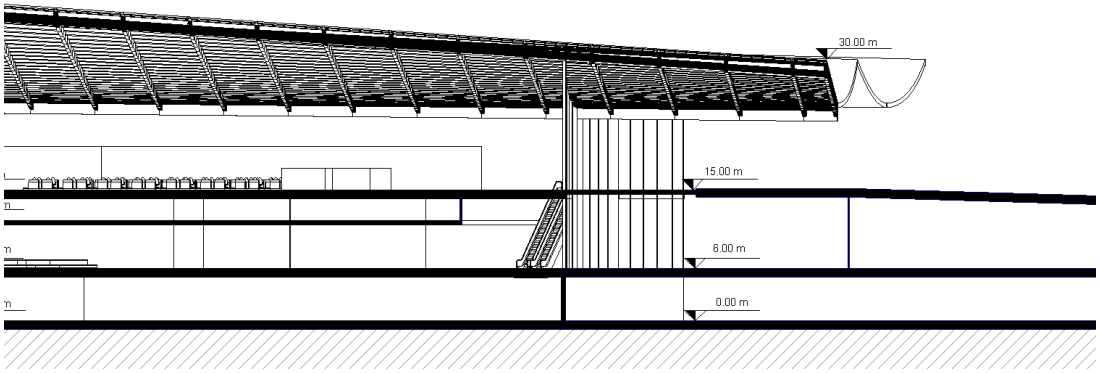


Figure 32 Section view

In the following figure, you can visualize the 3D model of the structure in Rhino.

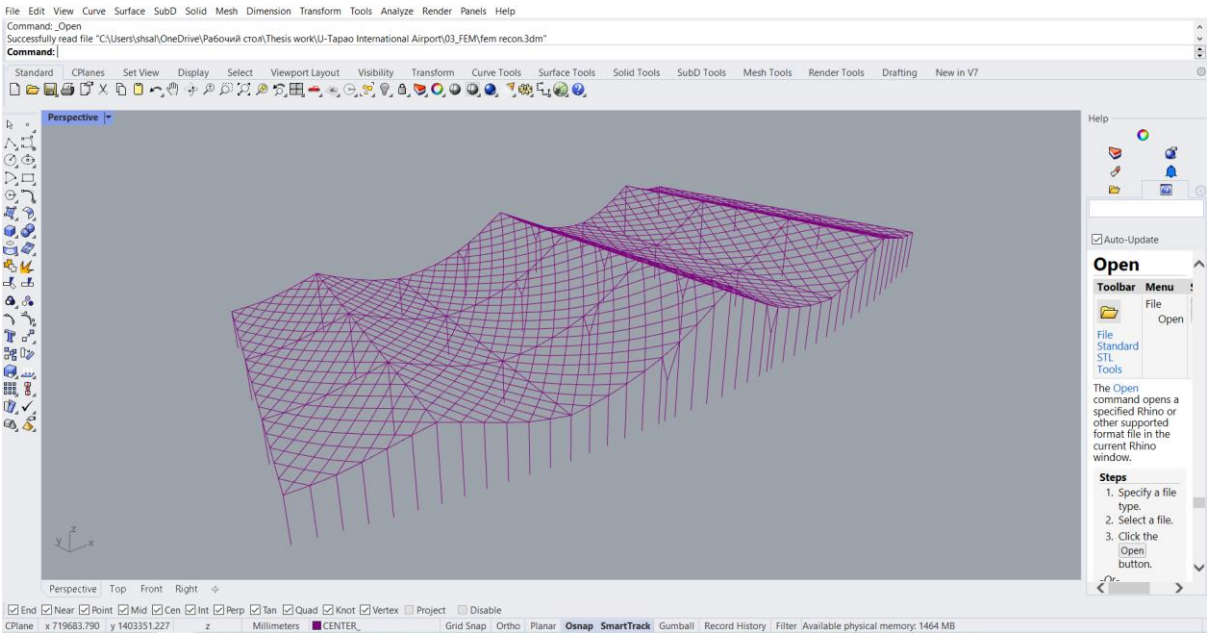


Figure 33 Long span roof model in Rhino

By clicking on the button launch Grasshopper, you can see the additional window, which is plugin for Rhino

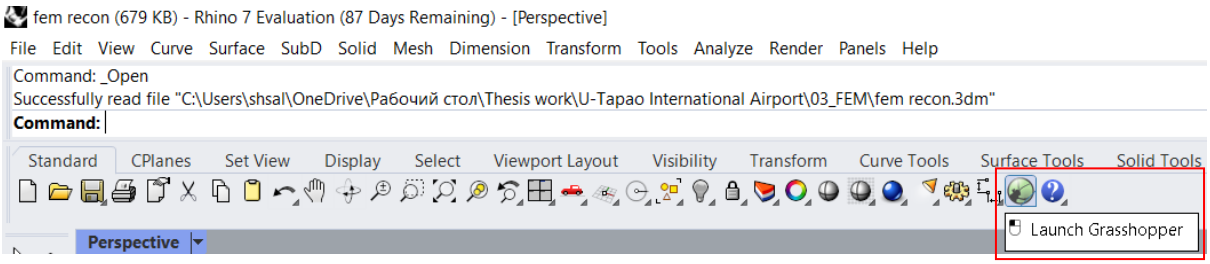


Figure 34 Grasshopper location

I.Development of a parametric model

Preliminary model

The parametric model of the structure was developed using the Grasshopper, as we have preliminary geometry of the model, we are able to think about different strategies to start modelling process. The first strategy used to model was fixing the nodes on the corners of the structure.

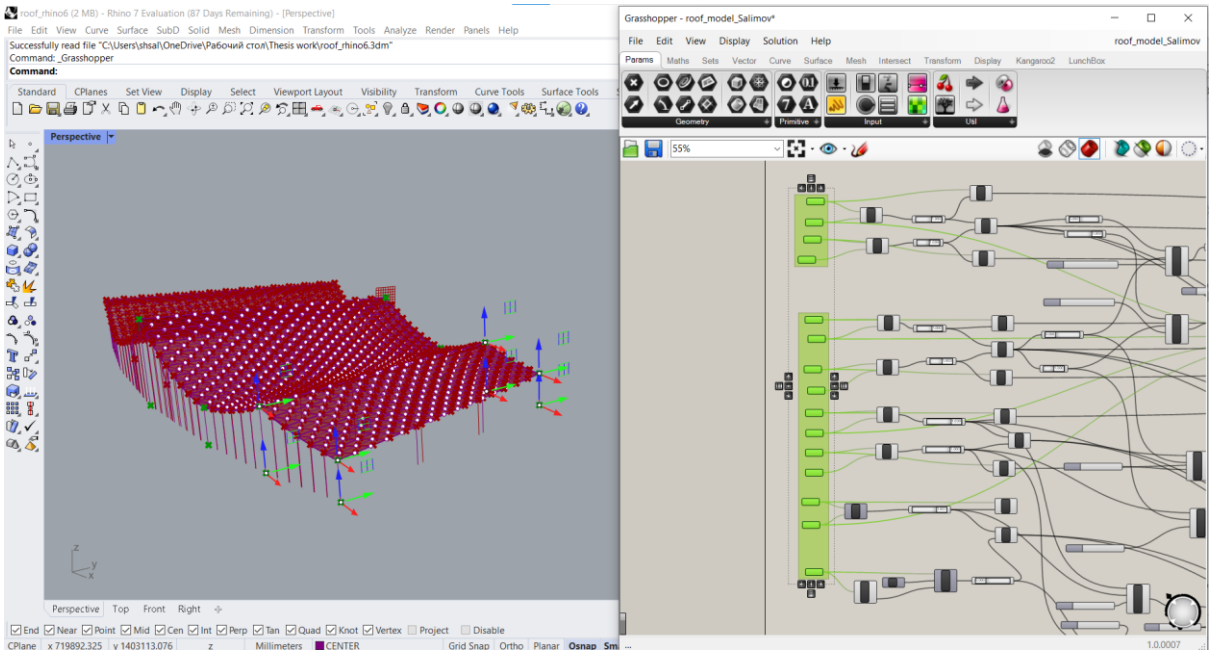


Figure 35 Parametric design of roof, initial stage

In this case, variables are constructed between those fixed nodes, which gives opportunity to save a time. The next operation was creating variable columns in the corners. In the following figure it can be clear the concept of variation. There is a function called point on curve, so node of the roof was connected to column using this command. It resulted, in possibility of playing with that node with respect to vertical axis.

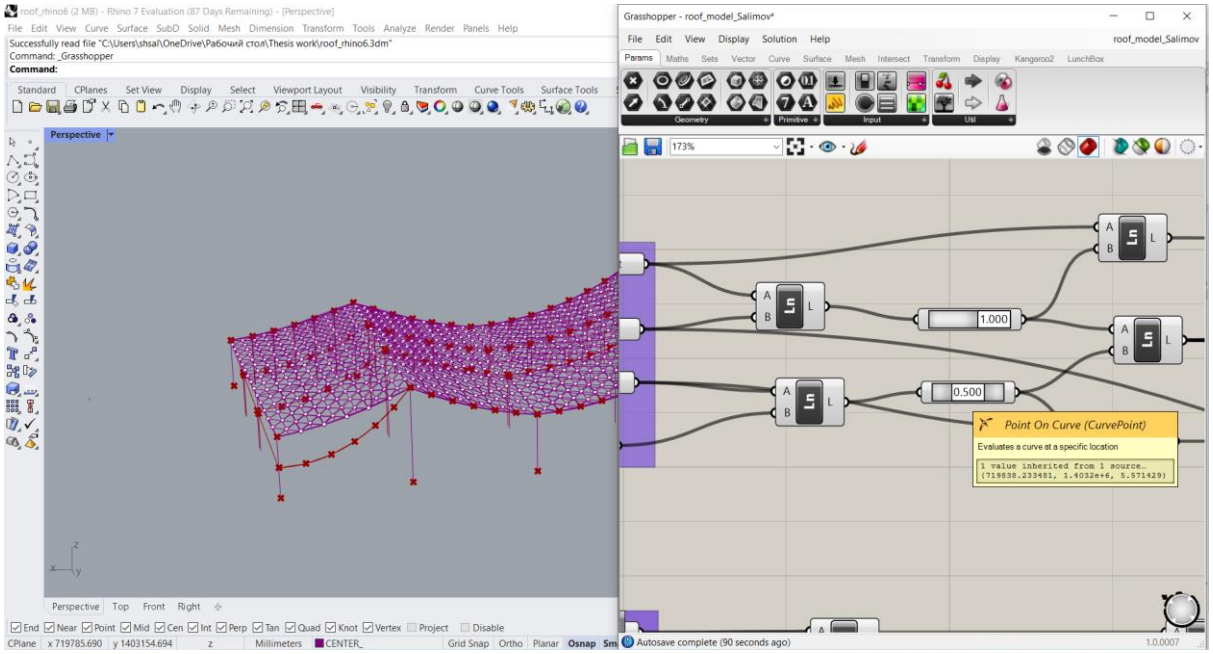


Figure 36 Parametric design of roof, commands regarding curves

Then, for the curves of the roof, catenaries were constructed and they are also made variable.

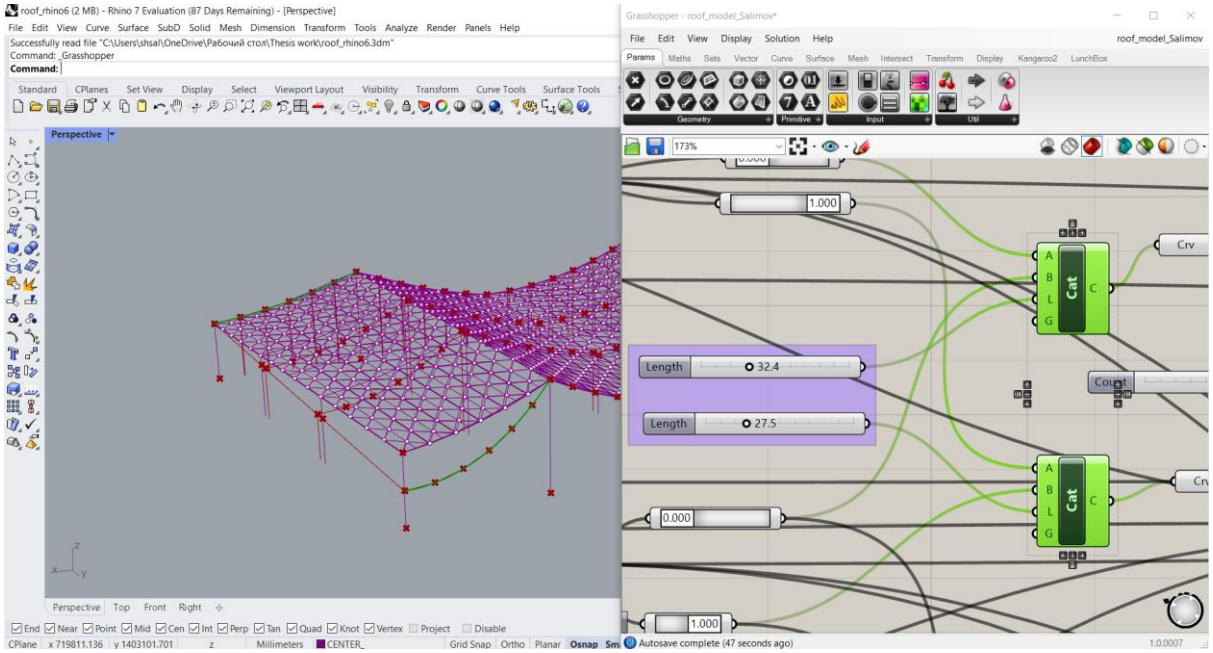


Figure 37 Parametric design of roof, commands regarding division of curves

Then these curves were divided, because we needed to have a columns on the nodes of grid, so if we divide the curves, we can consider division points as nodes.

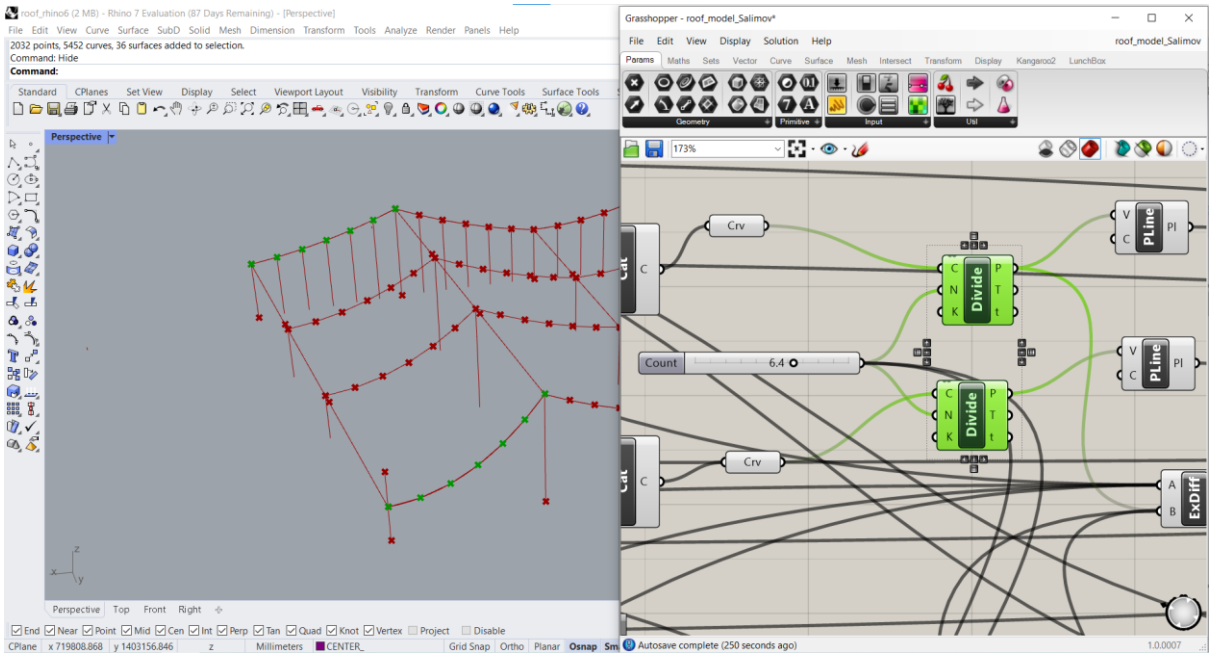


Figure 38 Parametric design of roof, commands regarding division of curves

After division, it is needed to construct a point on that division points and to create a column which will be depended on the curve.

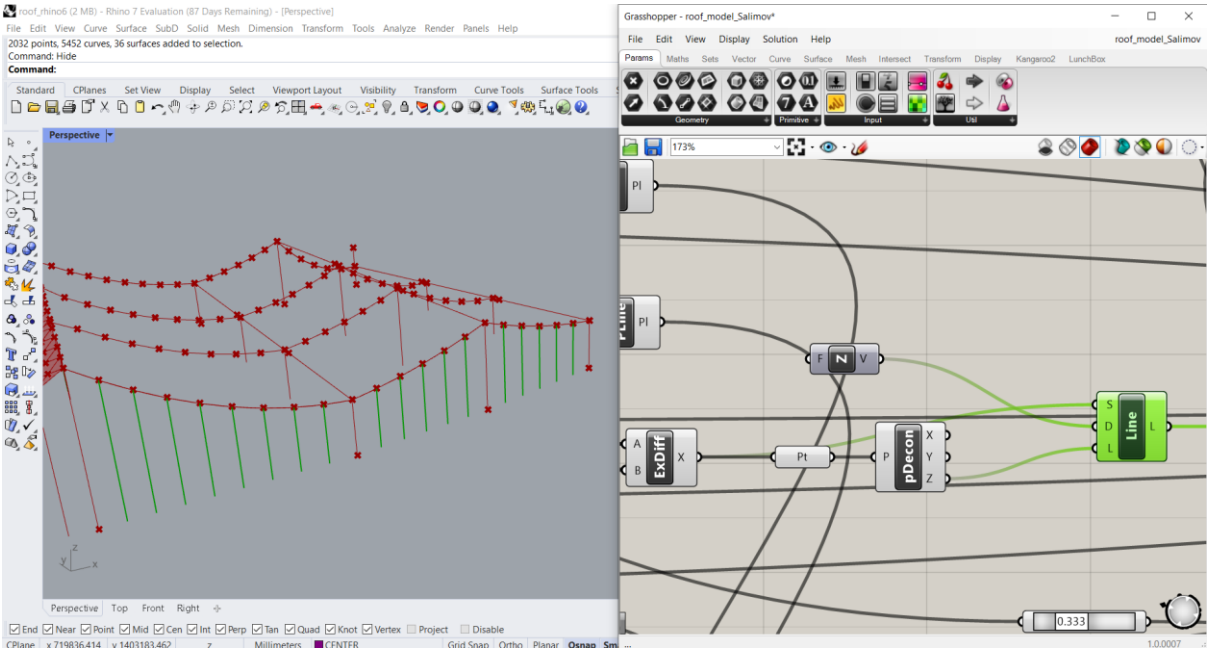


Figure 39 Creation parametric columns

The model was created by dividing the roof into portions, as can be seen from the figures, we have 3 portions in one half of the structure. As we four columns in one line, it was quite problematic to properly construct a column in the middle of the structure, because we don't have a certain node to consider as a starting point or end point of node as the nodes are variable. So, in a such situations, advantage of this softwares comes to scene, because we can use the function called "Closest point" and connect the column to that node.

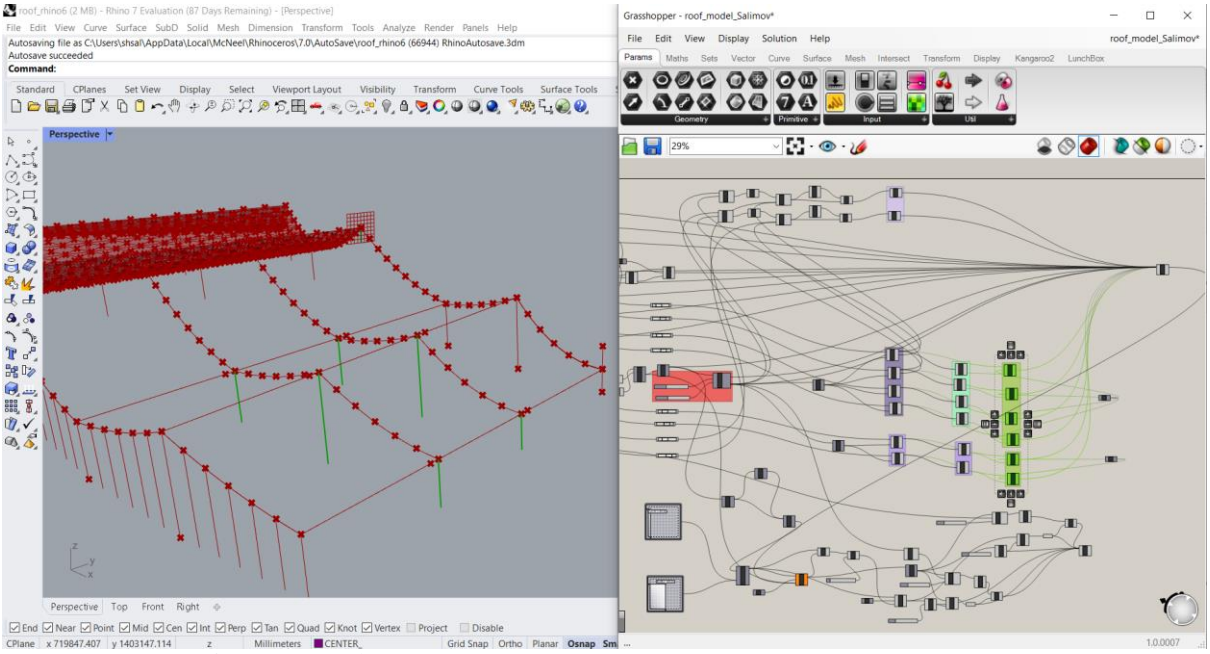


Figure 40 Creation of parametric columns

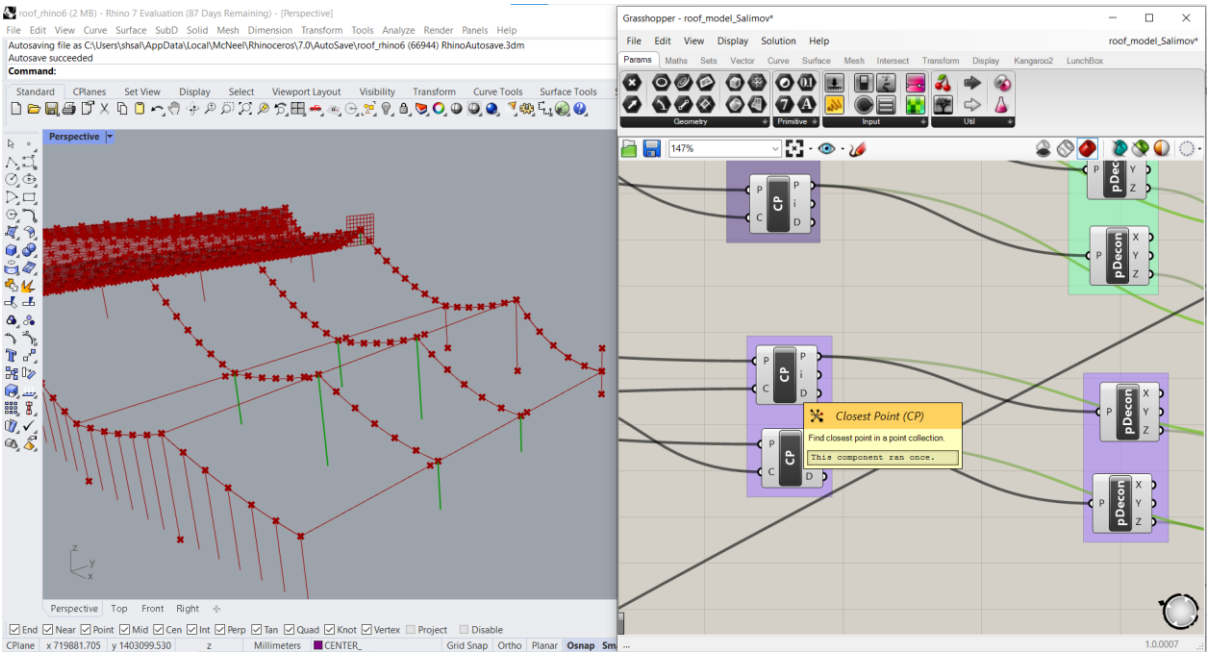


Figure 41 Clarfication of parametrisation

So, next step was to union all the curves to create a unique surface for the roof, which is called Diagrid.

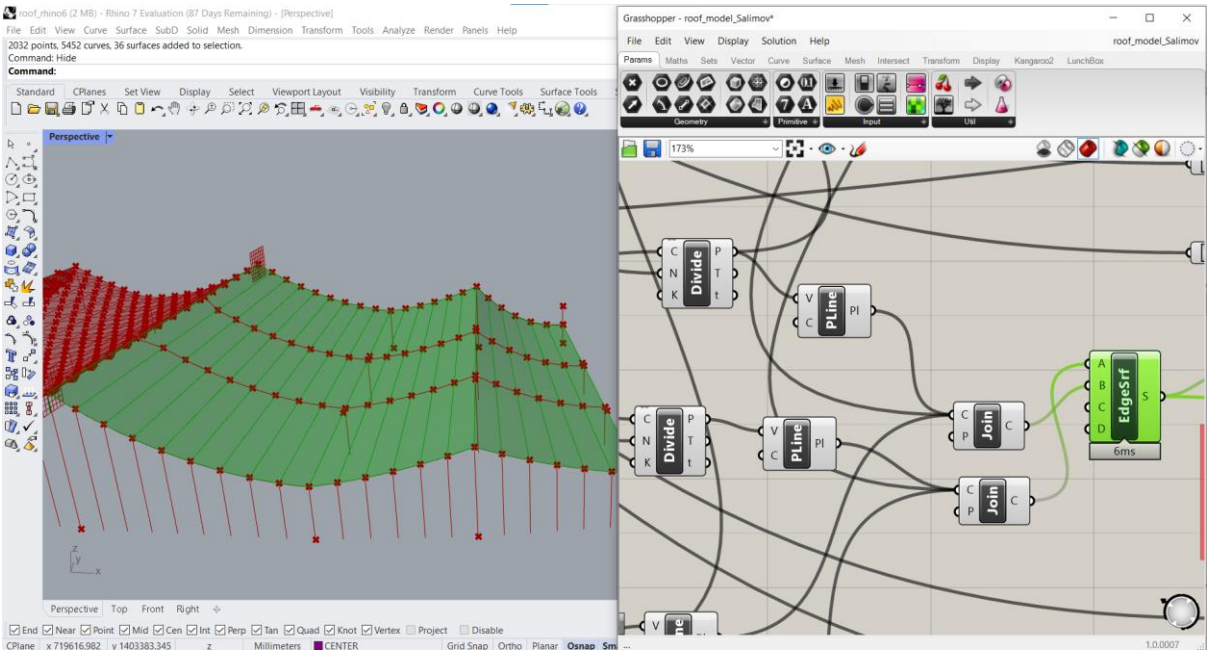


Figure 42 Creation of roof

In this picture, you can see the variable of the diagrid which are division in U and V directions.

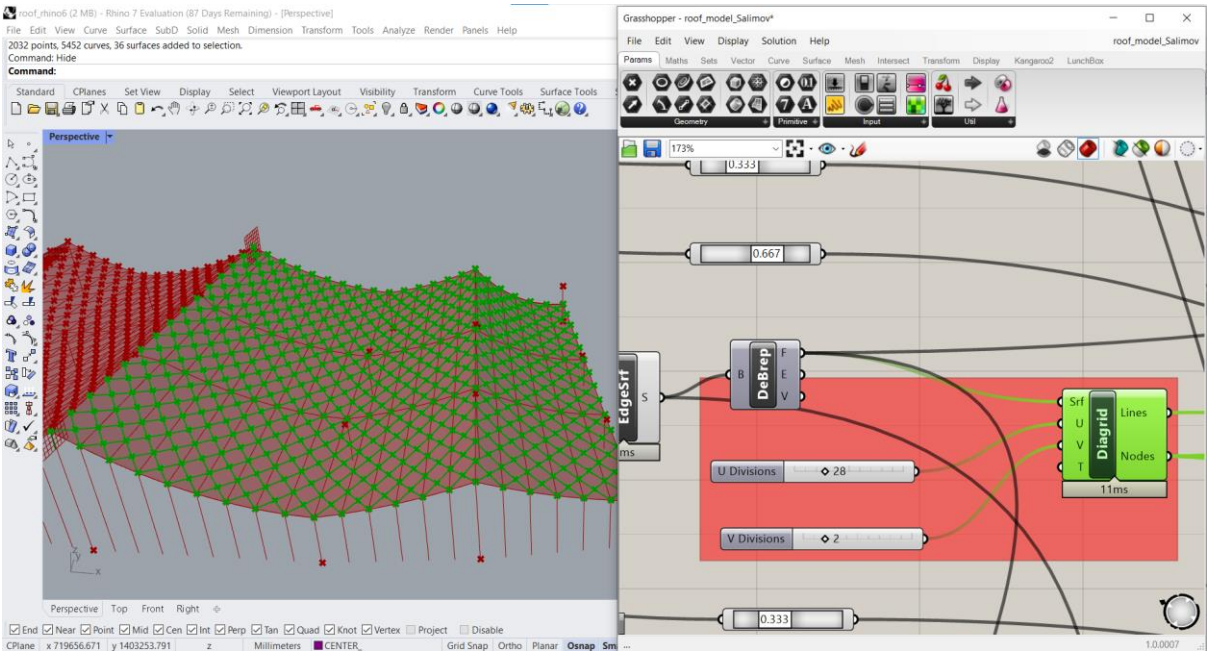


Figure 43 Creation of Diagrdid structure

We have our half model representing the long span roof structure in a parametric way. So, simplest way to complete the model is to make it mirror, because our structure is symmetric with respect to vertical zy plane.

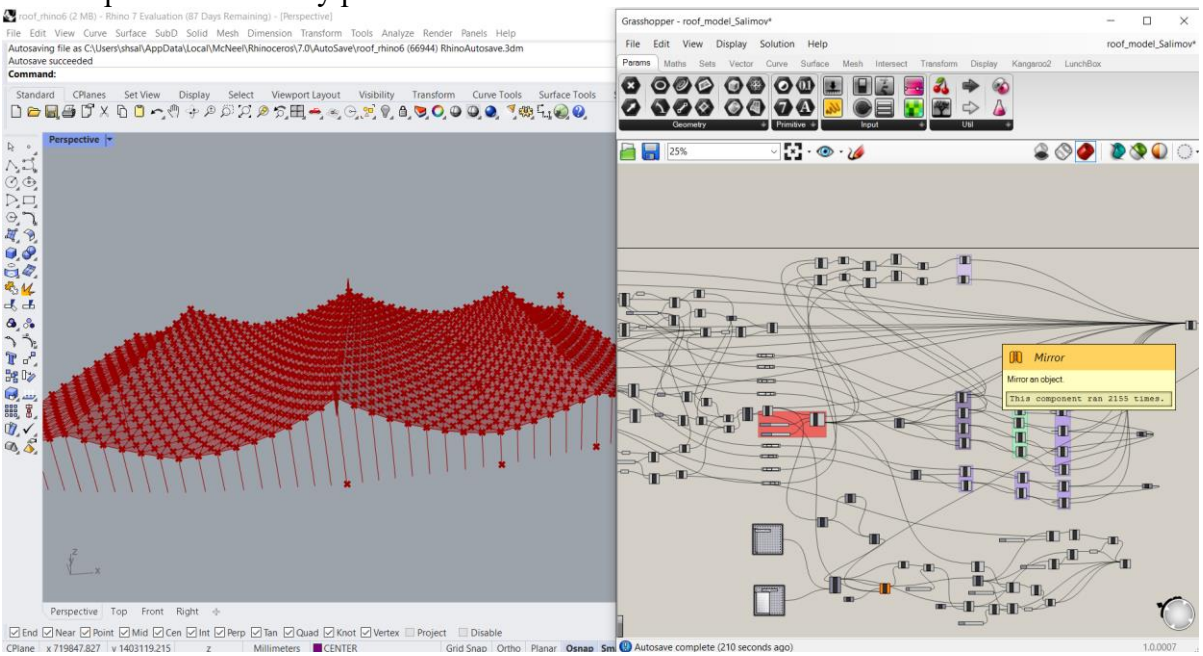


Figure 44 Mirroring the objects

In this case model was completed, but due to high number of variables, and complexity of the model it was impossible to perform FEA using this model. One of the main reason is that parametric model must be constrained well and should be as simple as possible to correctly obtain optimization results. For this reasons, another model was created using the same application but using different logical sequence, which will be the topic of next section.

Final parametric model

In this final development of model, firstly instead of creating some points on the nodes, it was decided to use curves, because it is then possible to directly create variable lines for transversal beams.

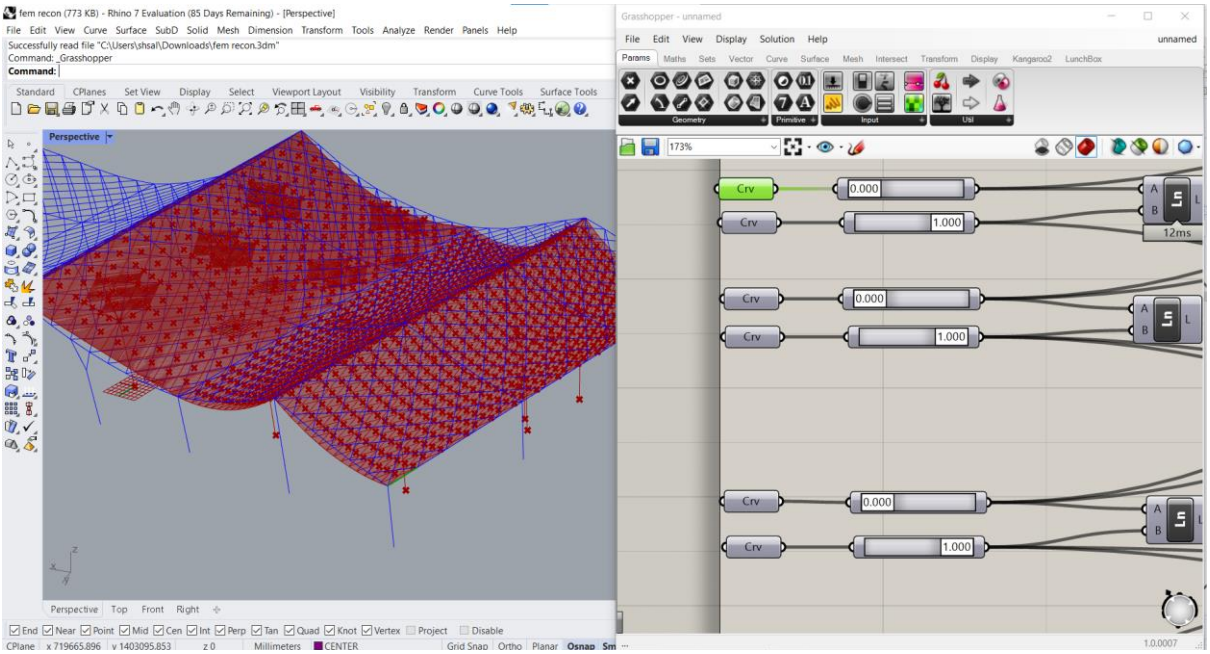


Figure 45 Parametric design process, initial stage

In Figure 46 you can realize that catenary curves were used in this model as well. But in this case we have simpler version due to limited commands regarding the nodes.

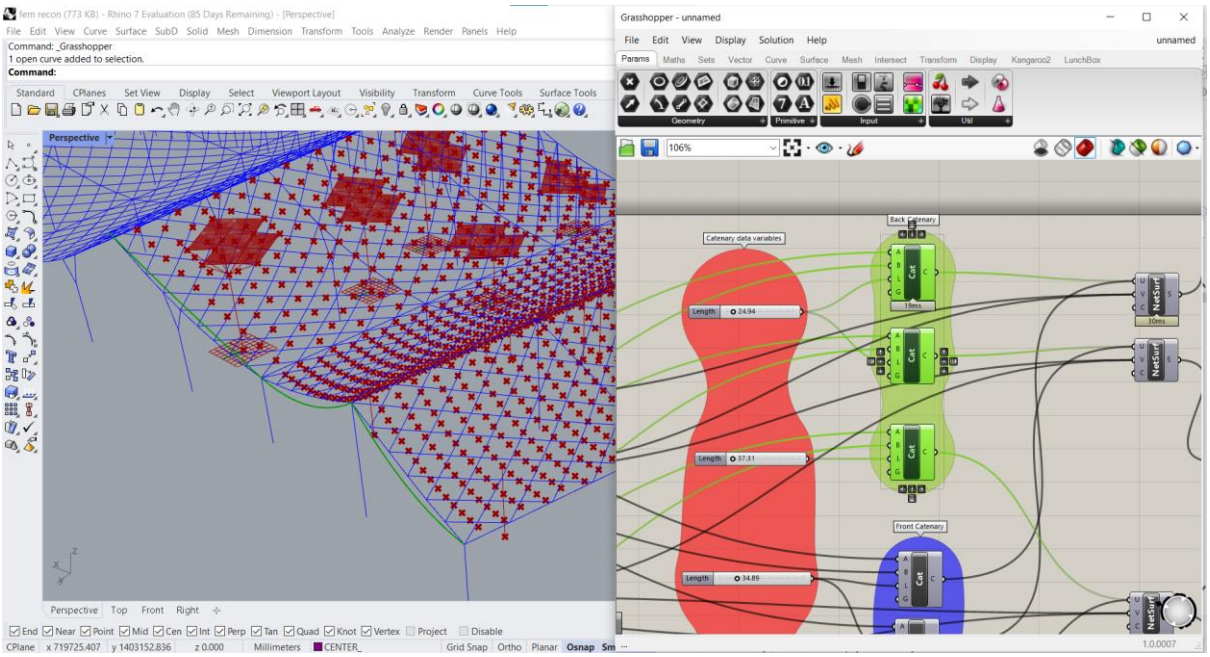


Figure 46 Parametric design process, curves creation

The parametric equation of catenary is given by:

$$x(t) = t$$

$$y(t) = \frac{1}{2}a(e^{\frac{t}{a}} + e^{-\frac{t}{a}}) = \text{a cosh}(\frac{t}{h})$$

Where t=0 corresponds to the vertex and a is a parameter describing the how quickly the catenary "opens up".

- Arch length:

$$s(t) = \text{asinh}\,(\frac{t}{a})$$

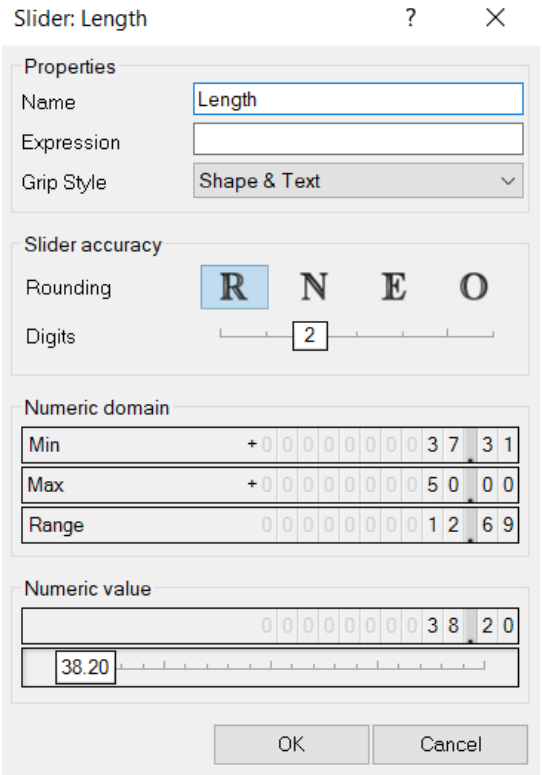
- Curvature:

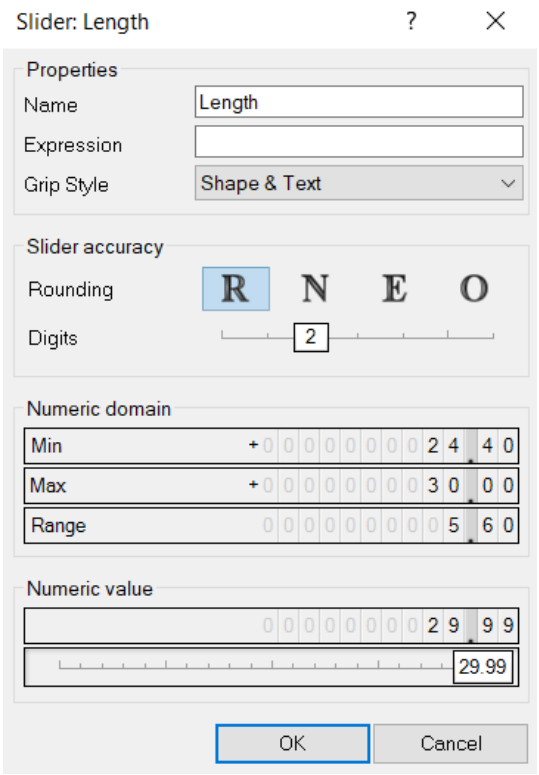
$$k(t) = \frac{1}{a}sech^2(\frac{t}{a})$$

- Tangential angle:

$$\varphi(t) = 2tan^{-1}[tang\left(\frac{t}{2a}\right)]$$

Length of catenary was set as a variable, as a consequence s(t) is our function, where parameter that is being varied is "a".





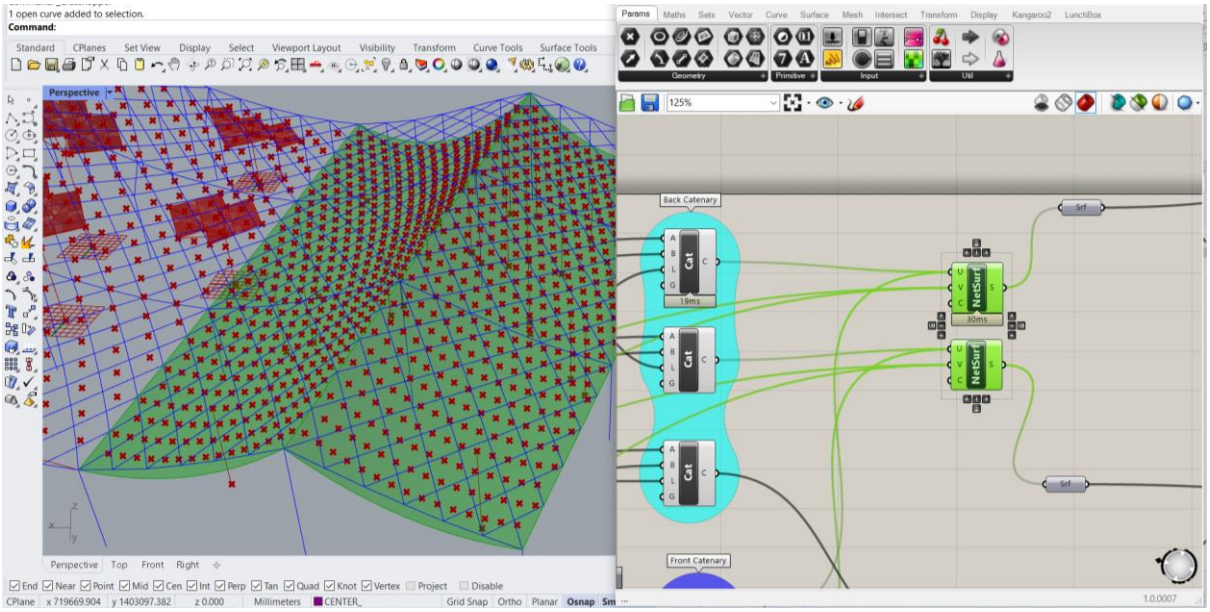


Figure 47 Parametric design process, evaluation of surface for grid

For this case, instead of creating a diagrid with all the nodes inside, it was decided to achieve grid structure with the help of some basic commands. Firstly, it was created network surface which is represented in the Figure 47 above.

Now, it must be clear why before we needed surface, because command “Diamond” requires a surface to perform a action. In this case, we have a grid structure without nodes at the corners. The points in the middle of the grid were created by purpose to place columns and make them variable.

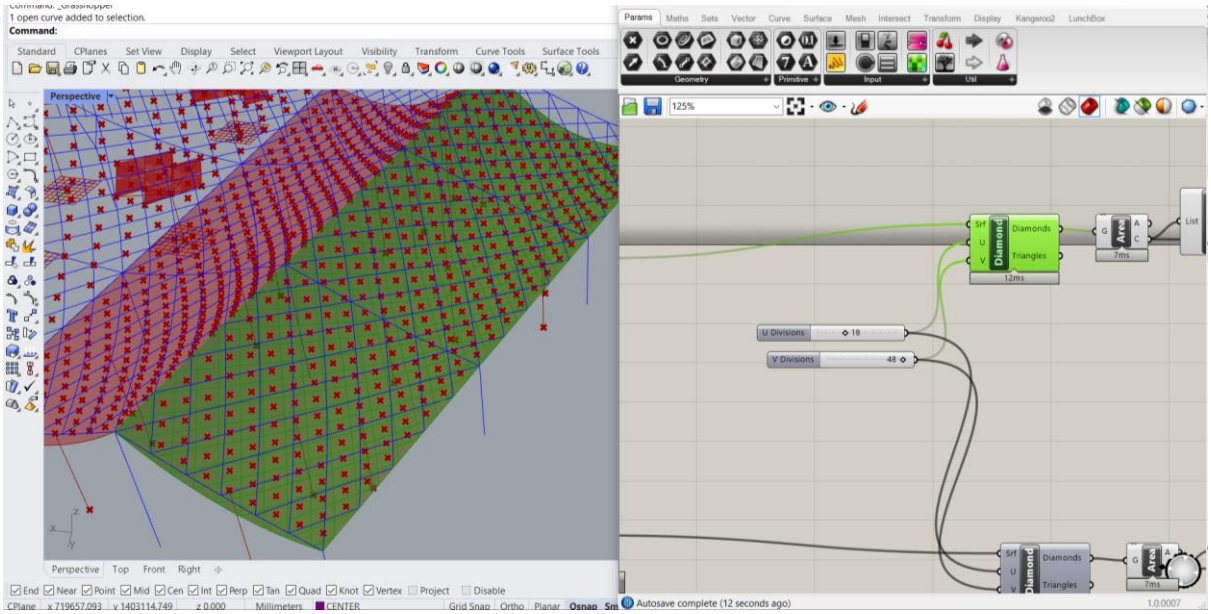


Figure 48 Parametric design process, creation of Diamond

After creation of grid surface at the top with nodes in the middle of the grids, columns were created. But most important thing in this figure, is that after creation of columns, their positions were realized by connecting to nodes. So, there are a lot of nodes and they depend on the divisions of a diamond, question may arise, how we can choose optimal position of

columns? Optimal position was chosen by putting gene pools for extraction of points from surface to connect the column. This gene pools can be played in thousands of ways, so idea was to use other plug-in also for this variation, to determine optimal “topology”.

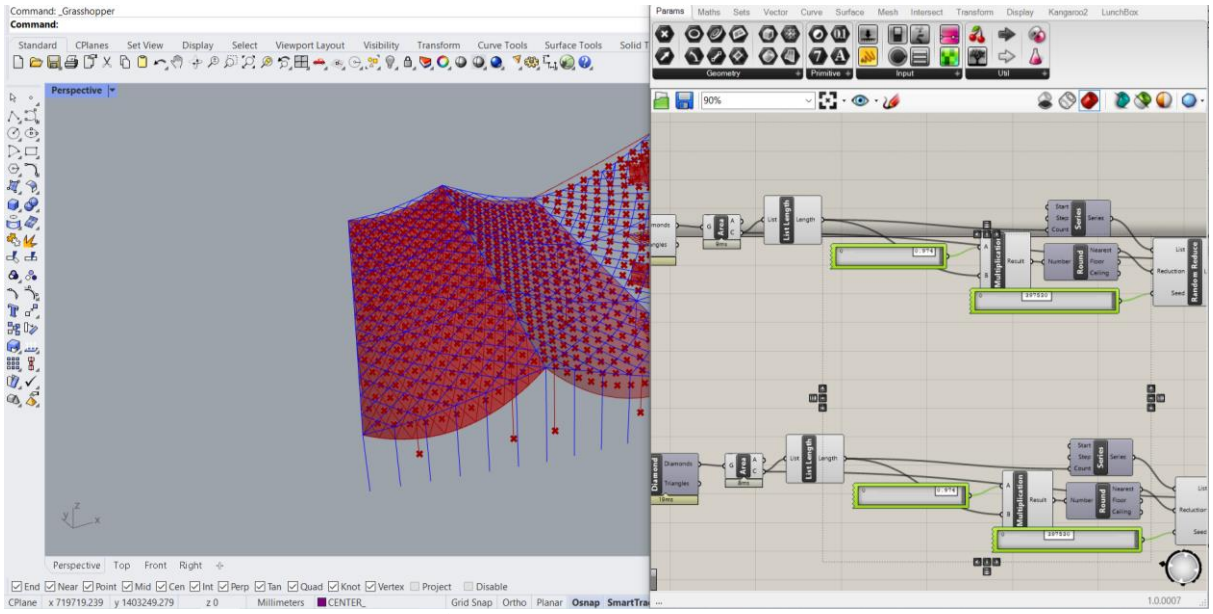


Figure 49 Parametrisation process, gene pools

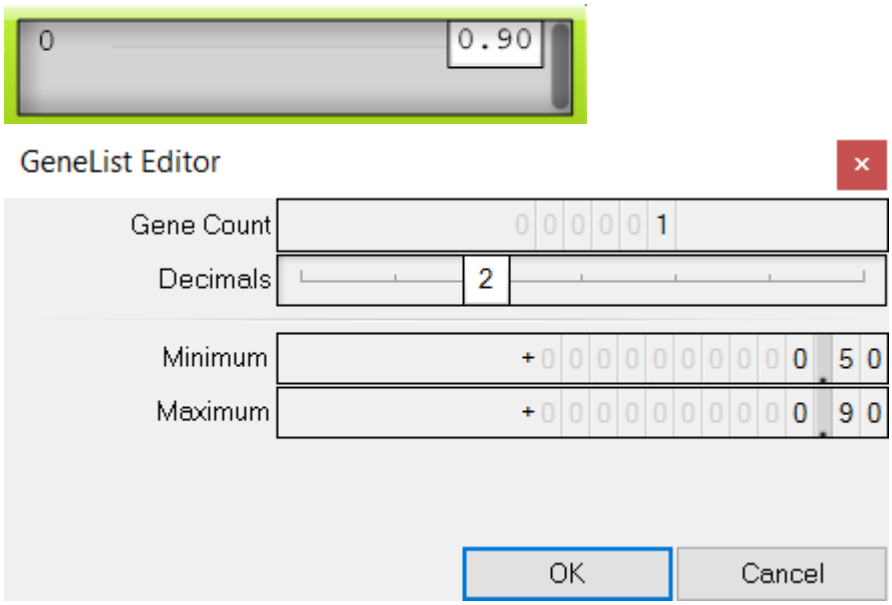


Figure 50 Gene pool for column's quantity

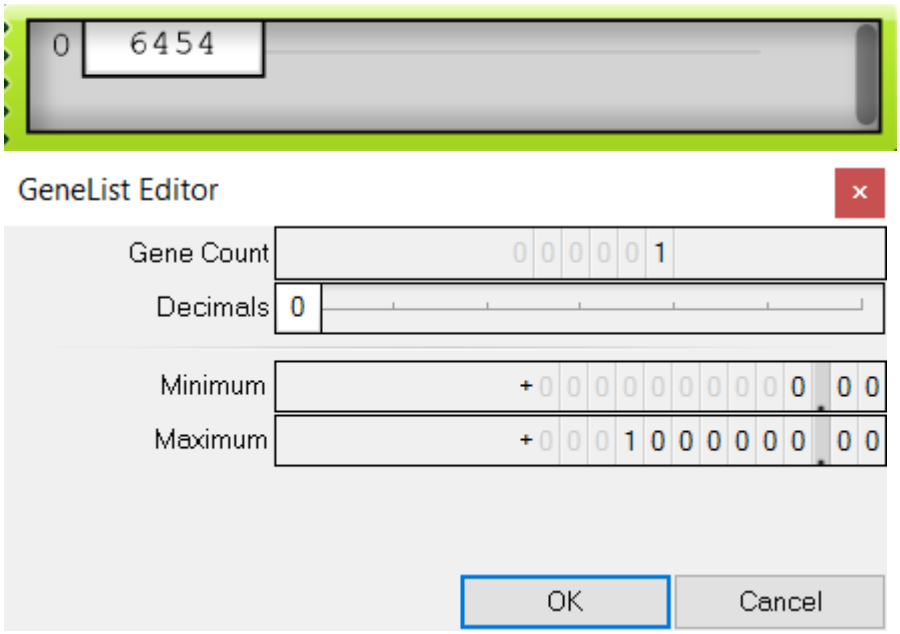


Figure 51 Gene pool for Column's position arrangement

Columns that are divided into parts in their heights must be realized also using “gene pool”, because if we make the number of beams in the upper part of column variable, it can be included in the optimization process.

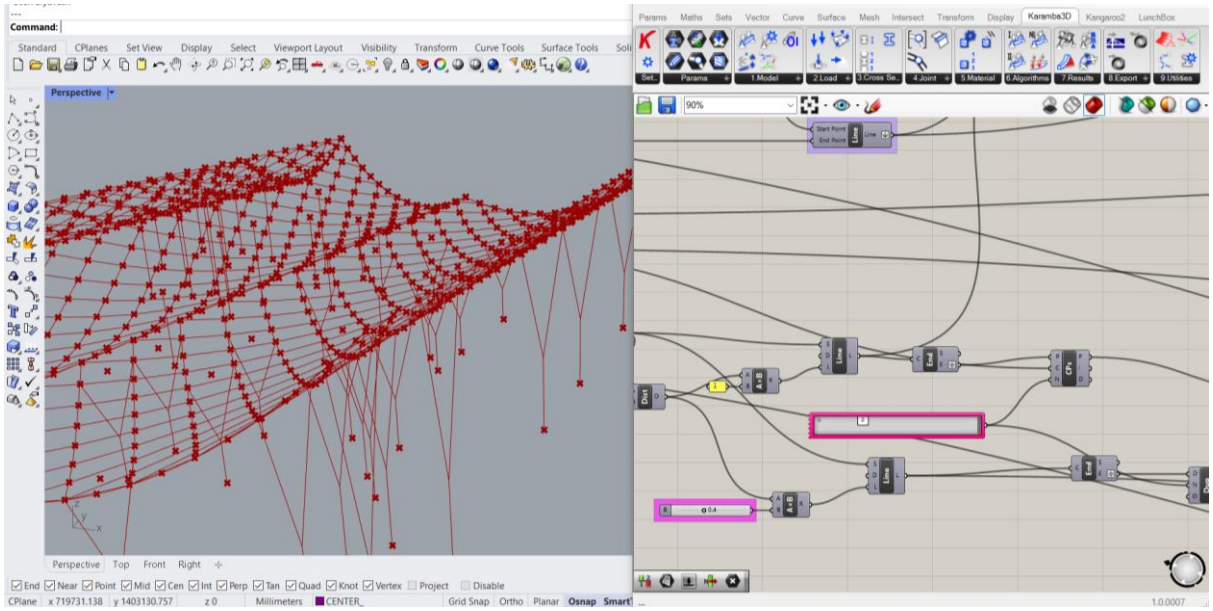


Figure 52 Column branches

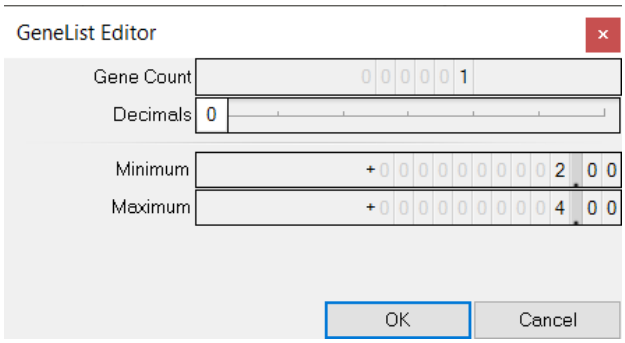


Figure 53 Gene pool for Branches

Now, before starting assembling the model, it is very important to separate the columns, beams of roof into discrete segments, so that in process of assembling, software can specify them as beams.
This procedure was performed in two steps, first segmentation for curves, then for transverse beams of the roof.

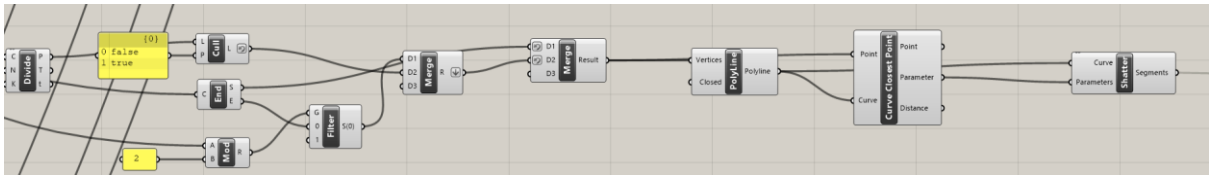


Figure 54 Dividing into segments, for curve

II.Assembling the model

Now, as soon as we completed the creation of parametric model, it is possible to move on assembling the model in Karamba3D. Assembling the model is the creation of Finite Element model using parametrically defined lines, points, planes, etc…Karamba3D is a interactive, parametric Finite Element program. It is easy to use for non-experts and has been tailored to the needs of architects and engineers in the early design phase. Karamba3D is embedded in the parametric environment of Grasshopper which is a plug-in for the 3d modelling tool Rhinoceros. This makes it easy to combine parameterized geometric models, finite element calculations and optimization algorithms like Octopus or Galapagos.

In order to calculate the behavior of a real world structure one needs to define its geometry, loads and supports. The component **"Assemble"** gathers all the necessary information and creates a structural model from it.

In case that some beams were defined by node indexes then these will refer to the list of points given at the **"Pt"**-input-plug: the first node in the list has index zero in the model, the next one index one, and so on. The **"Pt"**-input can also be used to give the model nodes a specific order.
The value at the input-plug **"LDist"** defines the distance of points below which they will be merged to one. This helps in dealing with inaccurate geometry. The limit distance default value is 5mm.

Cross sections of elements and materials can be defined either upon creating an element or at the **"Assemble"**-component. The latter option overrides the former and assigns cross sections and materials via element identifiers. Using regular expressions for selecting identifiers of elements provides a flexible means of attaching cross sections and materials to different parts of a model.
The output-plug **"Mass"** renders the mass of the structure in kilogram and includes user specified point-masses. **"COG"** represents the position of the center of gravity. When being plugged into a panel the model prints basic information about itself: number of nodes, elements, and so on. At the start of the list the characteristic length of the model is given, which is calculated as the distance between opposing corners of its bounding box.

Name	ID	Description	Type
Point	Pt	Input nodes of the model that may be connected by elements. May contain duplicate points. The elimination of duplicate nodes closer than 'LDist' does not apply to them. In case of duplicate points, elements are attached to them alternating: first element to first duplicate point, second element to second duplicate point, third element to first duplicate point in case there are only two points in one position. The same mechanism applies to point-loads and supports.	Point
Elem	Elem	Input beams/shells of the model	Element
Support	Support	input supports	Support
Load	Load	input loads	Load
Cross section	CroSec	Input cross sections of the model	CrossSection
Material	Material	Input material of the model	FemMaterial
Joint	Joint	Input element joints	Joint
Beam set	Set	Input sets of beams	ElemSet
Limit distance [m]	LDist	Limit Distance [m] for coincident points (for point-loads, supports given with coordinates).	Number

Figure 55 Input parameters

Name	ID	Description	Type
Model	Model	Model based on inputs	Model
Info	Info	information regarding the assembly of the model	String
Mass [kg]	Mass	Mass of structure in [kg]	Number
Center of Gravity	COG	Center of gravity of the model.	Point

Figure 56 Output

As we are working with only half of the structure, we have three Diagrid systems on the roof which was defined by the code. In the assembling, it was decided to choose line to beam command to transfer lines to beams.

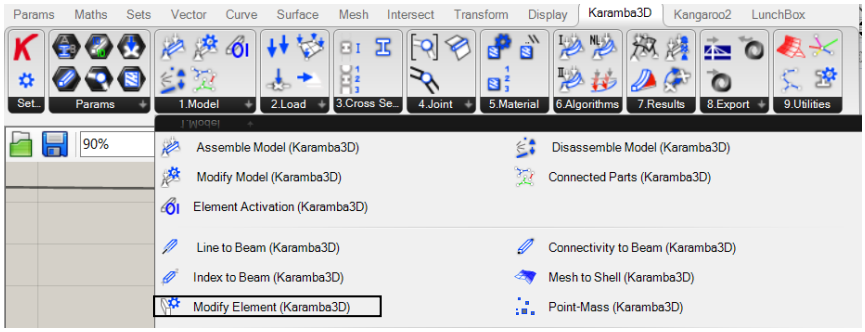


Figure 57 Options of Karamba

Line to Beam

The **"LineToBeam"**-component accepts straight lines, polylines and splines as geometric input. Polylines get exploded into segments. Splines are intersected according to the parameters 'ToPAng', 'ToPTol' and 'ToPMinL' (see below for their meaning) . All coordinates are in meters (or feet in case of Imperial units).

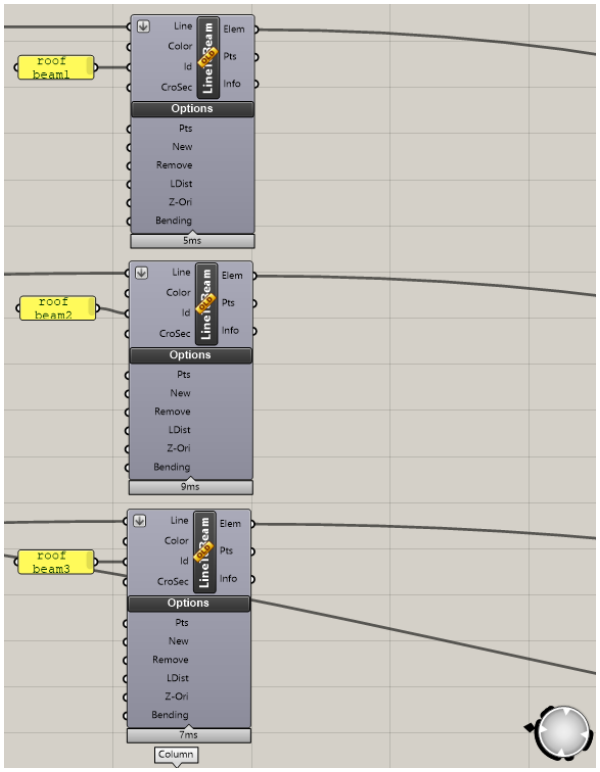


Figure 58 Line to beam transformation

Those beam elements which were transferred from line, now are connected through the component “assemble” as in Figure 54.

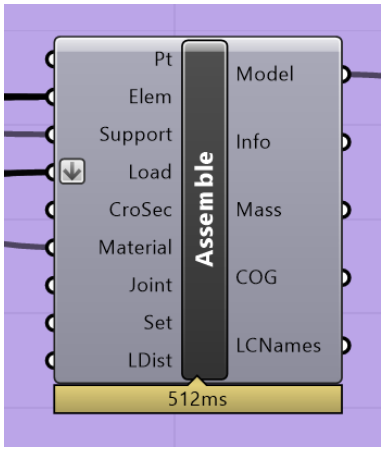


Figure 59 Assembling component

Cross Section

The next step of assembling the model is the assignment of cross section. Cross section can be defined for beams, shells and springs. In our case, structure is composed of beams, so we used Karamba component called “Cross Section Karamba3D”. But in this case cross sections were assigned to elements separately, in other words, component “assemble” wasn’t used.

Figure 60 Cross section component

In buildings, hollow sections are mainly used for columns and lattice girders or space frames for roofs. For our case study, as we are not going to optimize the cross section, optimal prediction for type of cross section is mandatory, so hollow circular cross section for columns and grid structure on roof was selected as the best suitable condition. The unique design of the Hollow tube enables it to be very long without losing form or shape. Use of direct joints reduces construction time as compared to Steel Rods. Hollow sections are safer to use as they don't have sharp edges and its light weight reduces injury.

Why hollow section?

1. Hollow sections have a better finish
2. Hollow sections are much stronger in torsion
3. Hollow sections don’t fail in buckling, meaning that you will be able to design it more easily without having to undertake the buckling check
4. If buckling is the main failure reason for your I beam, it may be that an RHS will be more economical to use (* This may not be the case in your region)
5. Hollow sections have much better minor axis bending resistance, so if you have a beam in bi-axial bending you would be better off with an RHS or SHS
6. If you have a column with high minor axis moments or an offset connection on the minor axis side, a hollow section would be stronger than an I-section
7. You can chamfer cut and weld hollow section into a frame
8. Hollow section takes less paint and is easier to paint
9. Hollow sections (RHS and SHS) have flat, square surfaces, and are precise, so you can use them in engineering and product applications
10. Hollow sections come in several different colours and patterns, so as long as you like a smooth grey (rough browny orange in outdoor applications), you can use them as-is.

In the Figure 57, the parametric relation of dimensions of beams can be visualized clearly.

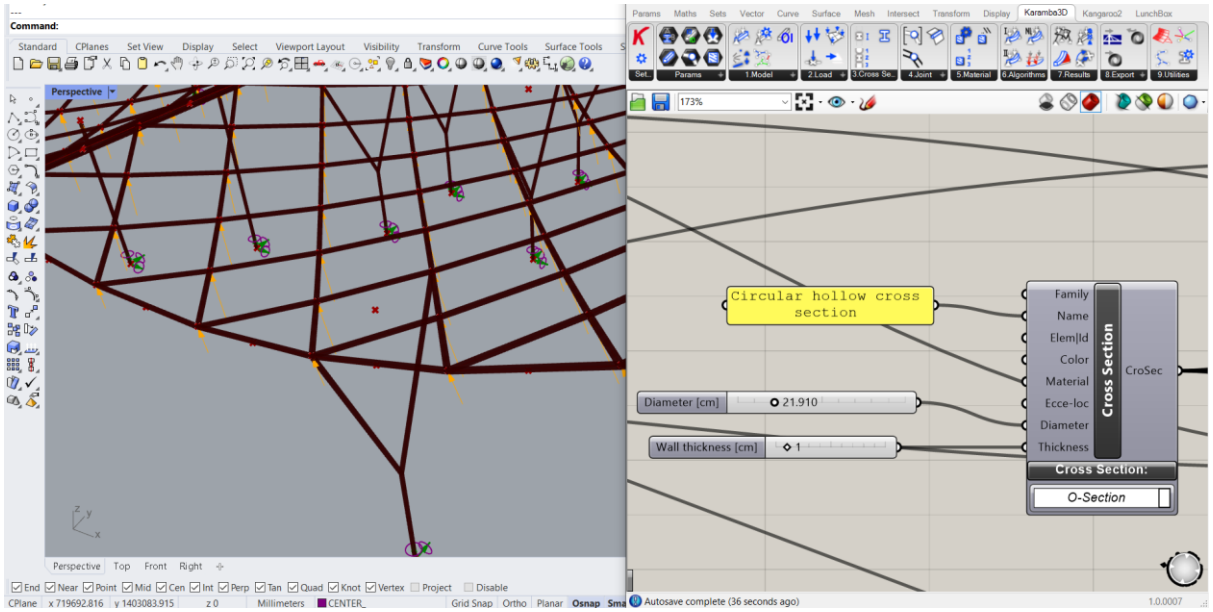


Figure 61 Cross section of columns and grid structure

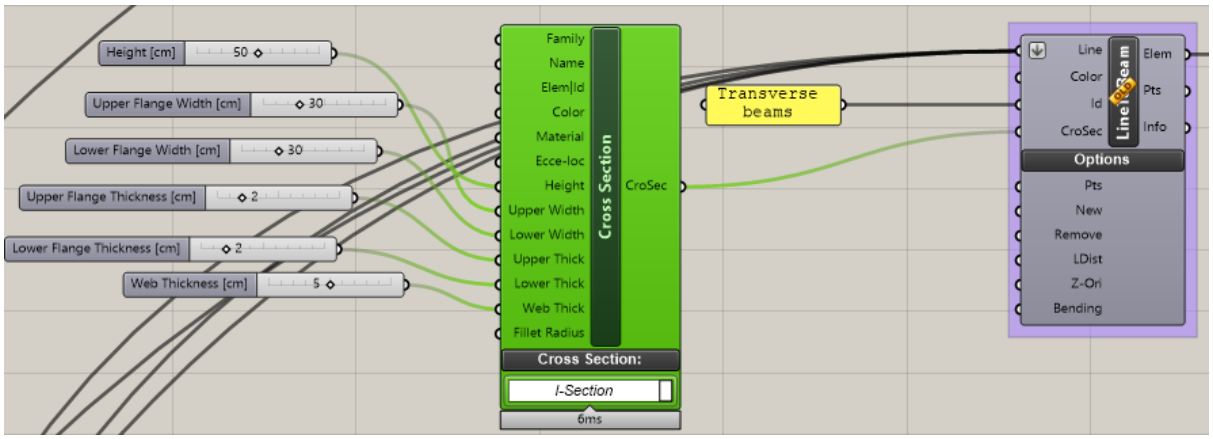
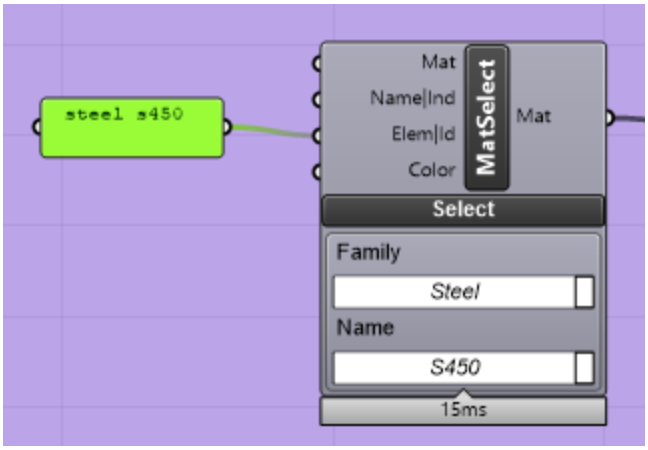


Figure 62 Cross section for Transverse beams

Material types

The Steel Grid Structure is a high-order statically indeterminate space structure composed of many rods according to certain rules. The space is small in force, light in weight, rigid, and has good seismic resistance. In our model, material was selected as S450 grade. Grade 450 stainless steel alloy is a martensitic type stainless steel. It has high ductility and strength. It has a yield strength which is nearly 3 times the strength of grade 304 stainless steel.



Loads

Notes from EUROCODE

(1) The wind pressure acting on the external surfaces, w_e, should be obtained from Expression (5.1).

$$w_e = q_p(z_e) \cdot c_{pe} \tag{5.1}$$

where:

$q_p(z_e)$ is the peak velocity pressure

z_e is the reference height for the external pressure given

c_{pe} is the pressure coefficient for the external pressure

4.5 Peak velocity pressure

(1) The peak velocity pressure $q_p(z)$ at height z, which includes mean and short-term velocity fluctuations, should be determined.

NOTE 1 The National Annex may give rules for the determination of $q_p(z)$. The recommended rule is given in Expression (4.8).

$$q_p(z) = \left[1 + 7 \cdot I_v(z)\right] \cdot \tfrac{1}{2} \cdot \rho \cdot v_m^2(z) = c_e(z) \cdot q_b \tag{4.8}$$

where:

ρ is the air density, which depends on the altitude, temperature and barometric pressure to be expected in the region during wind storms

$c_e(z)$ is the exposure factor given in Expression (4.9)

$$c_e(z) = \frac{q_p(z)}{q_b} \tag{4.9}$$

q_b is the basic velocity pressure given in Expression (4.10)

Now, we need to find q_b and c_e

$$q_b = \frac{1}{2} \cdot \rho \cdot v_b^2 \qquad (4.10)$$

NOTE 2 The values for ρ may be given in the National Annex. The recommended value is 1,25 kg/m^3.

To calculate v_b, we have to do following procedure.

4.2 Basic values

(1)P The fundamental value of the basic wind velocity, $v_{b,0}$, is the characteristic 10 minutes mean wind velocity, irrespective of wind direction and time of year, at 10 m above ground level in open country terrain with low vegetation such as grass and isolated obstacles with separations of at least 20 obstacle heights.

NOTE 1 This terrain corresponds to terrain category II in Table 4.1.

NOTE 2 The fundamental value of the basic wind velocity, $v_{b,0}$, may be given in the National Annex.

(2)P The basic wind velocity shall be calculated from Expression (4.1).

$$v_b = c_{dir} \cdot c_{season} \cdot v_{b,0} \qquad (4.1)$$

where:

v_b is the basic wind velocity, defined as a function of wind direction and time of year at 10 m above ground of terrain category II

$v_{b,0}$ is the fundamental value of the basic wind velocity, see (1)P

c_{dir} is the directional factor, see Note 2.

c_{season} is the season factor, see Note 3.

NOTE 1 Where the influence of altitude on the basic wind velocity v_b is not included in the specified fundamental value $v_{b,0}$ the National Annex may give a procedure to take it into account.

NOTE 2 The value of the directional factor, c_{dir}, for various wind directions may be found in the National Annex. The recommended value is 1,0.

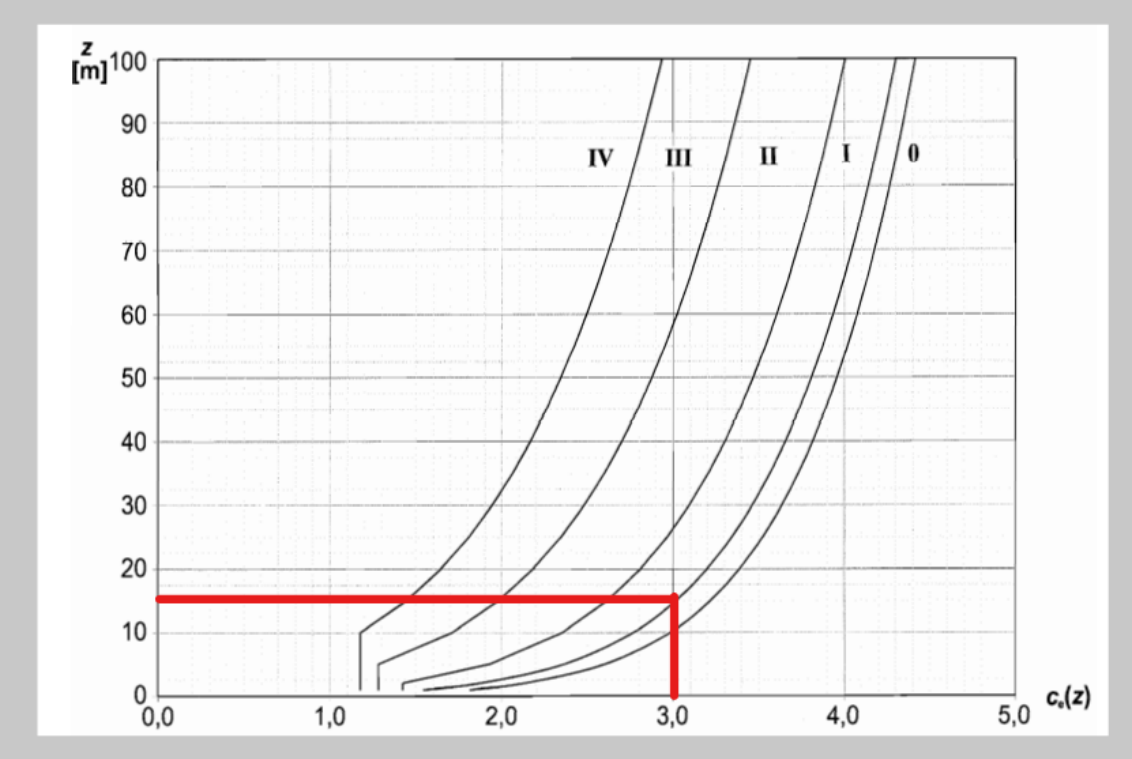
NOTE 3 The value of the season factor, c_{season}, may be given in the National Annex. The recommended value is 1,0.

NOTE 4 The 10 minutes mean wind velocity having the probability p for an annual exceedence is determined by multiplying the basic wind velocity v_b in 4.2 (2)P by the probability factor, c_{prob} given by Expression (4.2). See also EN 1991-1-6.

$\mathbf{c}_{dir}$ and $\mathbf{c}_{season}$ are taken as 1, to be conservative.

1.6.1
fundamental basic wind velocity
the 10 minute mean wind velocity with an annual risk of being exceeded of 0, 02, irrespective of wind direction, at a height of 10 m above flat open country terrain and accounting for altitude effects (if required)

The fundamental value of basic wind velocity was chosen to be 25 m/s
So, $\mathbf{v_{b0}}$**= 25 m/s** and also $\mathbf{v_b}$**=25 m/s**
Now, to use equation 4.9 we need $\mathbf{c_e(z)}$



To use this graph we have to specify our terrain category, and reference height and it was decided to choose as category I, as worst condition. And the elevation is 13m from mean sea level.

Table 4.1 — Terrain categories and terrain parameters

Terrain category		z_0 m	z_{min} m
0	Sea or coastal area exposed to the open sea	0,003	1
I	Lakes or flat and horizontal area with negligible vegetation and without obstacles	0,01	1
II	Area with low vegetation such as grass and isolated obstacles (trees, buildings) with separations of at least 20 obstacle heights	0,05	2
III	Area with regular cover of vegetation or buildings or with isolated obstacles with separations of maximum 20 obstacle heights (such as villages, suburban terrain, permanent forest)	0,3	5
IV	Area in which at least 15 % of the surface is covered with buildings and their average height exceeds 15 m	1,0	10
NOTE: The terrain categories are illustrated in A.1.			

Now, as soon as we know v_b , we are able to compute q_b and q_p

$$q_b = \frac{1}{2} \cdot \rho \cdot v_b^2$$

$$c_e(z) = \frac{q_p(z)}{q_b}$$

$c_e(z)=3$

q_b=390,625 kg/m3*m2/s2=390.625N/m2
q_p=1.17 kN/m2

(1) The wind pressure acting on the external surfaces, w_e, should be obtained from Expression (5.1).

$$w_e = q_p(z_e) \cdot c_{pe} \quad (5.1)$$

where:

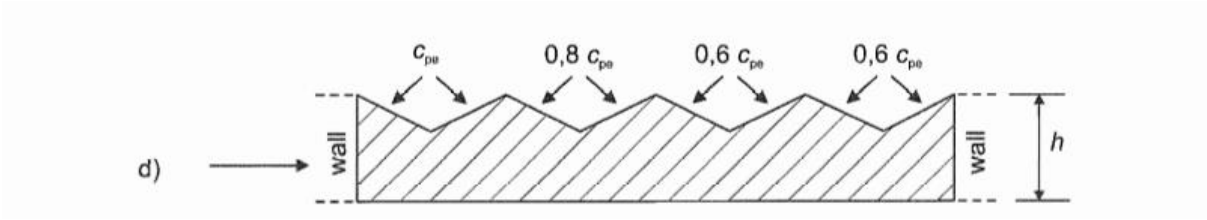
$q_p(z_e)$ is the peak velocity pressure

z_e is the reference height for the external pressure given in

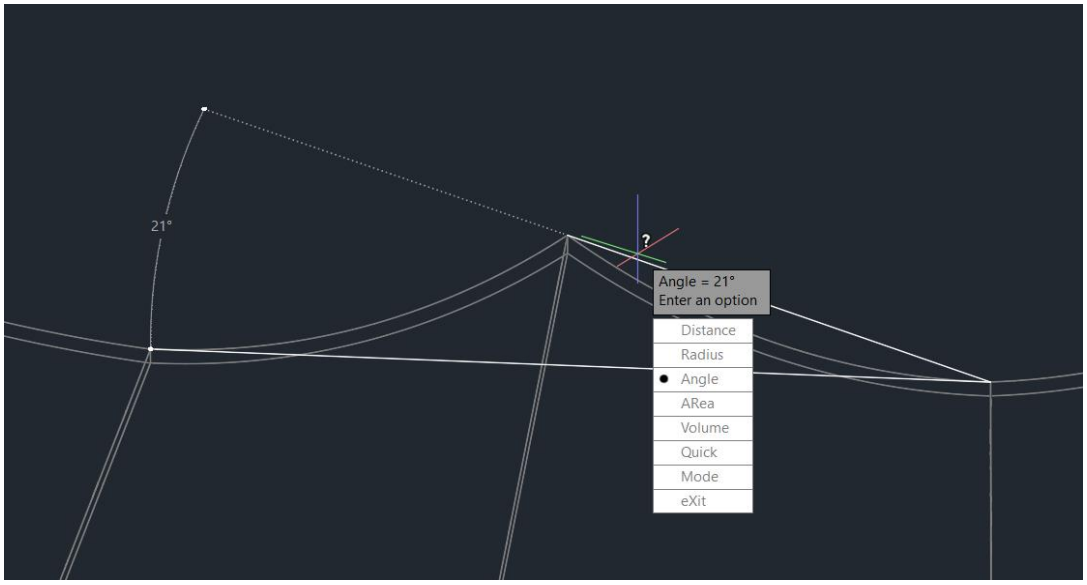
c_{pe} is the pressure coefficient for the external pressure

24

Now, to compute pressure on our surface, we have to find preffusure coefficient. First of all, we shouldn't forget that our roof is multipitch roof.



In this case, coefficients of duopitch roof is used but according to this scheme.



Our angle is 21 degrees.

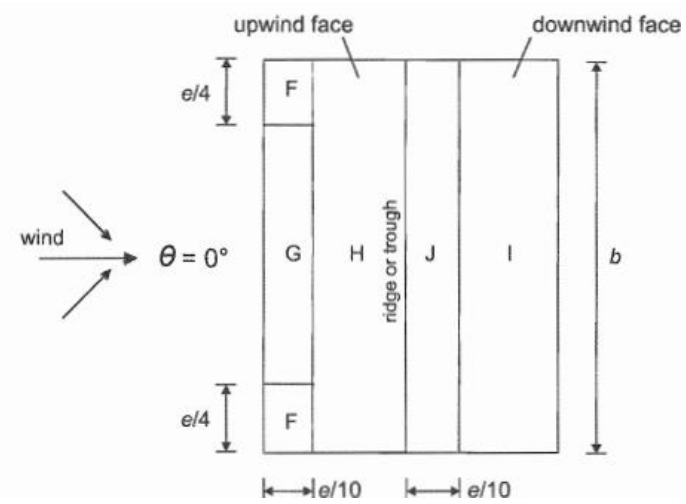


Table 7.4a — Recommended values of external pressure coefficients for duopitch roofs

Pitch Angle α	Zone for wind direction $\theta = 0°$									
	F		G		H		I		J	
	$C_{pe,10}$	$C_{pe,1}$	$C_{pe,10}$	$C_{pe,1}$	$C_{pe,10}$	$C_{pe,1}$	$C_{pe,10}$	$C_{pe,1}$	$C_{pe,10}$	$C_{pe,1}$
-45°	-0,6		-0,6		-0,8		-0,7		-1,0	-1,5
-30°	-1,1	-2,0	-0,8	-1,5	-0,8		-0,6		-0,8	-1,4
-15°	-2,5	-2,8	-1,3	-2,0	-0,9	-1,2	-0,5		-0,7	-1,2
-5°	-2,3	-2,5	-1,2	-2,0	-0,8	-1,2	+0,2		+0,2	
							-0,6		-0,6	
5°	-1,7	-2,5	-1,2	-2,0	-0,6	-1,2	-0,6		+0,2	
	+0,0		+0,0		+0,0				-0,6	
15°	-0,9	-2,0	-0,8	-1,5	-0,3		-0,4		-1,0	-1,5
	+0,2		+0,2		+0,2		+0,0		+0,0	+0,0
30°	-0,5	-1,5	-0,5	-1,5	-0,2		-0,4		-0,5	
	+0,7		+0,7		+0,4		+0,0		+0,0	
45°	-0,0		-0,0		-0,0		-0,2		-0,3	
	+0,7		+0,7		+0,6		+0,0		+0,0	
60°	+0,7		+0,7		+0,7		-0,2		-0,3	
75°	+0,8		+0,8		+0,8		-0,2		-0,3	

NOTE 1 At $\theta = 0°$ the pressure changes rapidly between positive and negative values on the windward face around a pitch angle of $\alpha = -5°$ to $+45°$, so both positive and negative values are given. For those roofs, four cases should be considered where the largest or smallest values of all areas F, G and H are combined with the largest or smallest values in areas I and J. No mixing of positive and negative values is allowed on the same face.

NOTE 2 Linear interpolation for intermediate pitch angles of the same sign may be used between values of the same sign. (Do not interpolate between $\alpha = +5°$ and $\alpha = -5°$, but use the data for flat roofs in 7.2.3). The values equal to 0,0 are given for interpolation purposes

The coefficient that we need are -0.9 and -0.5 for H and I parts of roof. But in some regions of roof there may arise high uplifting forces, so to be conservative, for upwind face -2.5 and for downwind face -1 was taken.

So, at the end we have:

W_e (for upwind face)= -2.93 kN/m2
W_e (for downwind face)=-1.17 kN/m2

To be conservative!
Now, below is the illustration of application of load on our structure.
First of all, load is applied through Karamba3D, and there are different load types.

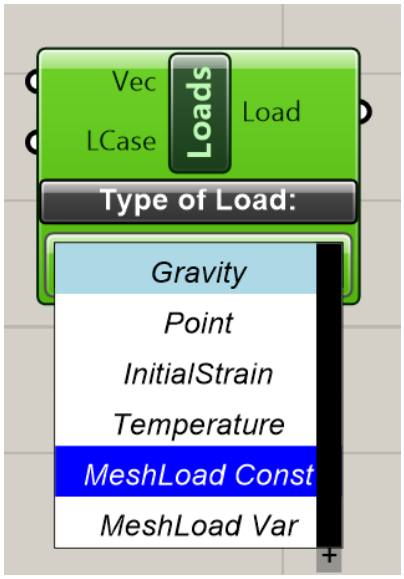


Figure 63 Load types

Besides, we have two types of load, gravity load which is self-weight and wind load. For this reason, load cases must be specified. Load case "0" was assigned to gravity load and load case "1" for wind loads.

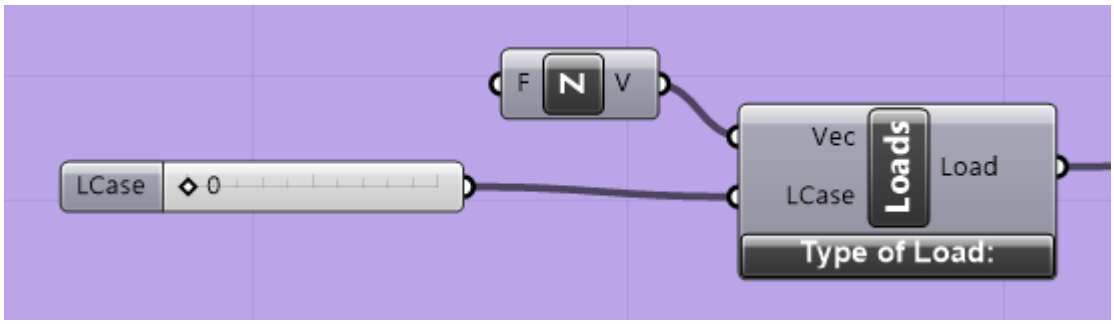


Figure 64 Load case 0, Gravity

For the application of wind load, as it is distributed along area, it was possible to use "Mesh Load Constant" option, we could also use "Mesh Load Variable", but it will not make big difference in the results.

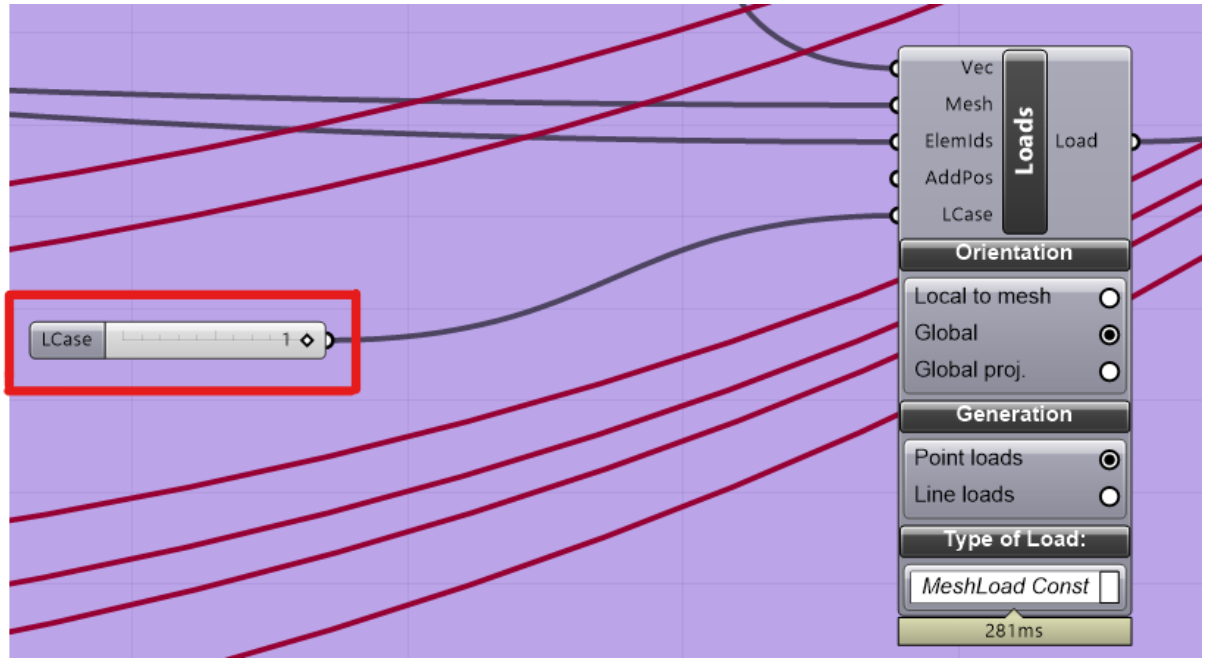


Figure 65 Load case 1, wind load

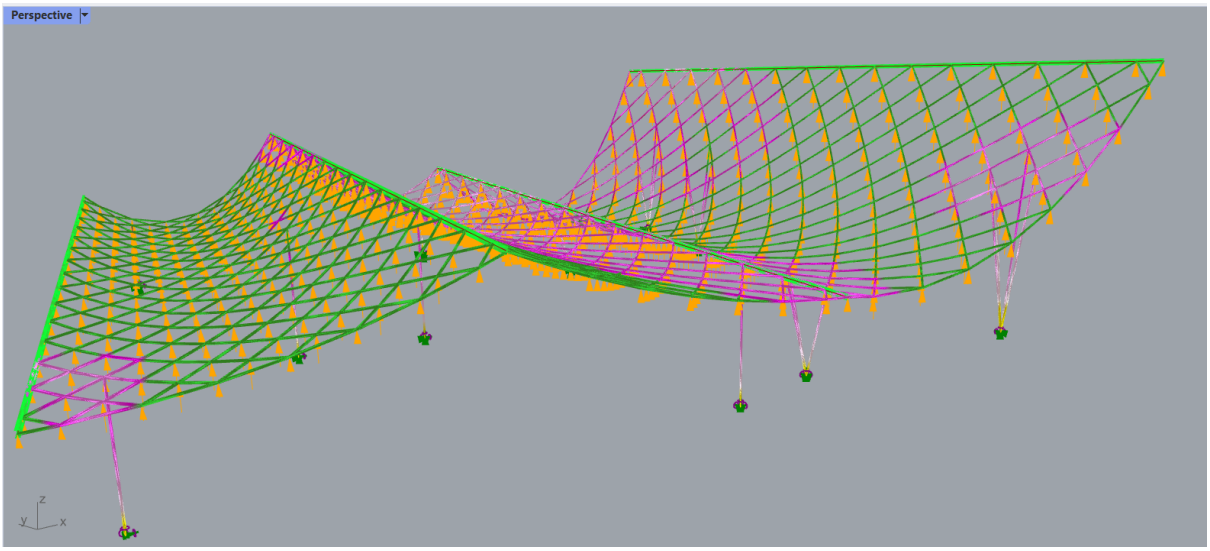


Figure 66 Load sybols in the structure

Supports

Supports of the structure were also defined by using Karamba3D, in this case all the translations and rotations were fixed. This was done by collecting the all nodes on the ground to one list and connecting it to function called “support”.

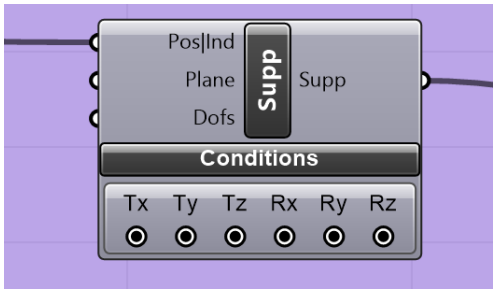


Figure 67 Support conditions

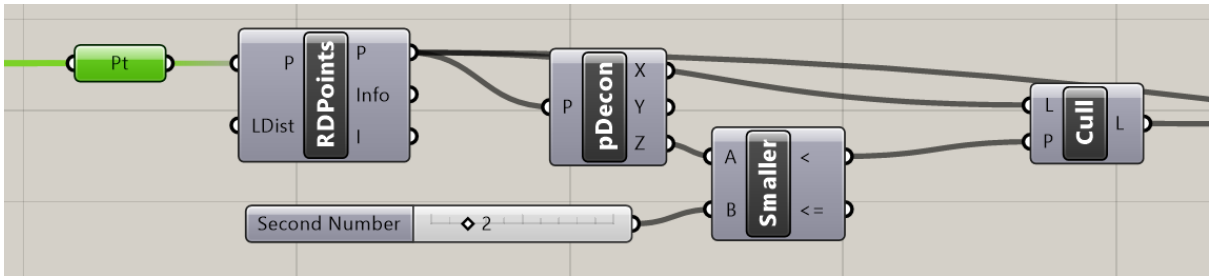


Figure 68 Extraction of nodes on the ground

Results

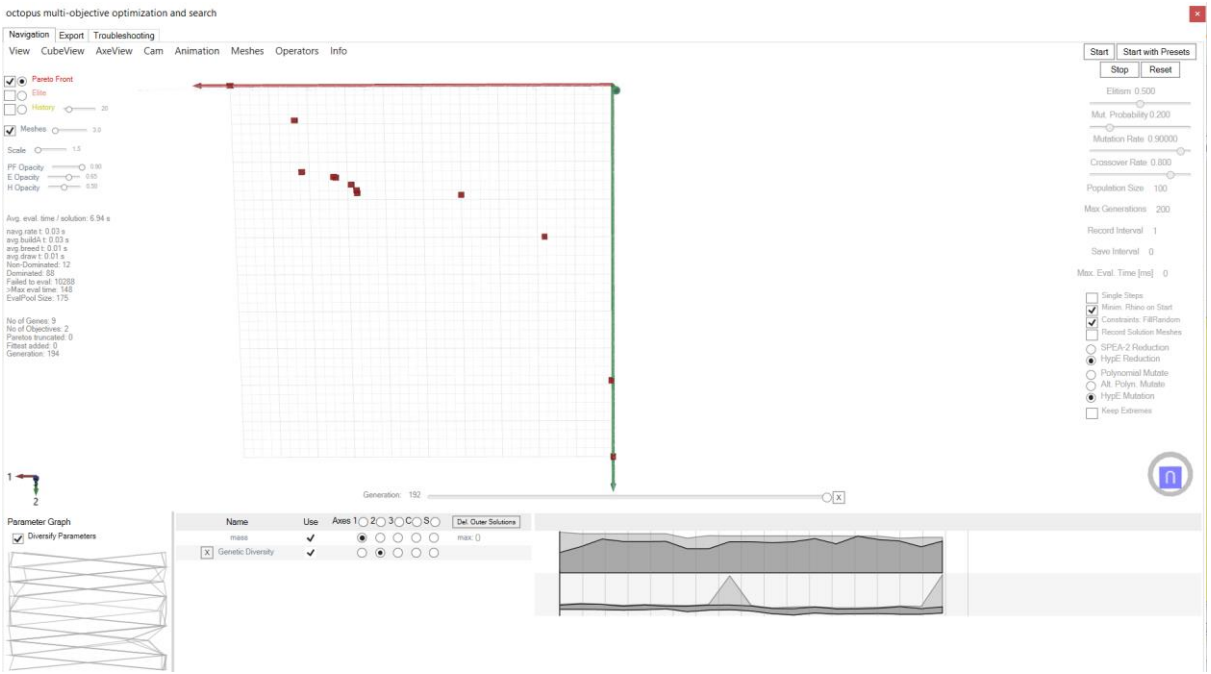


Figure 69 Pareto Front

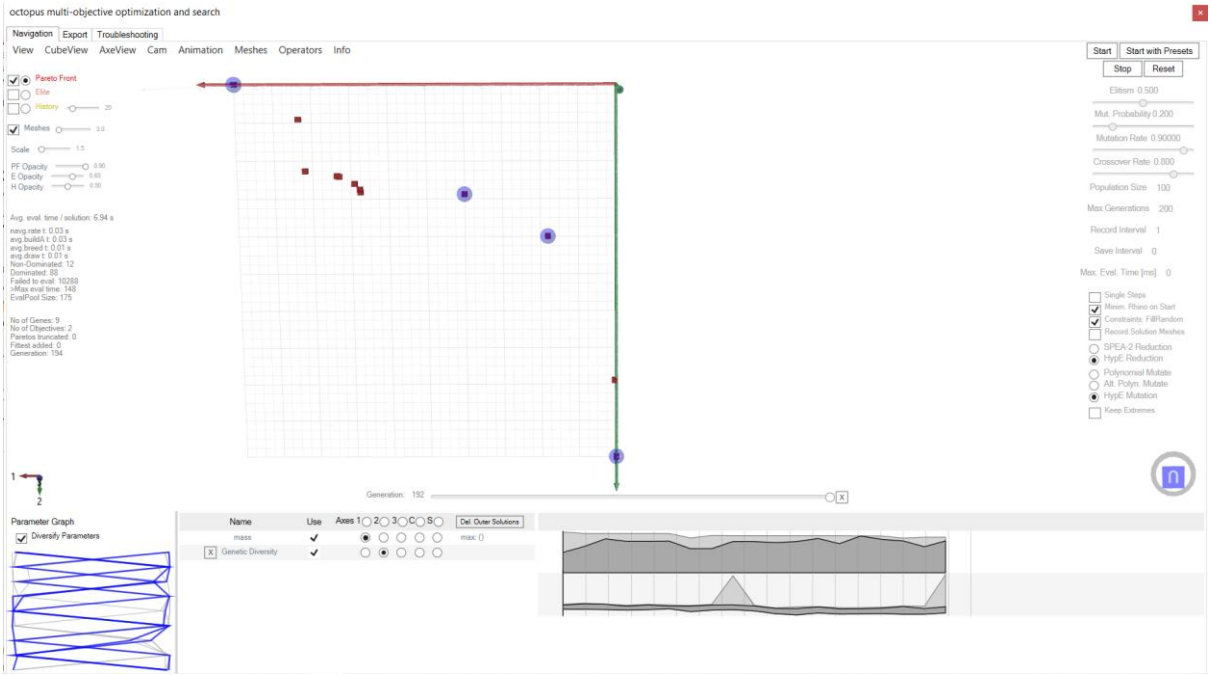


Figure 70 Pareto Front

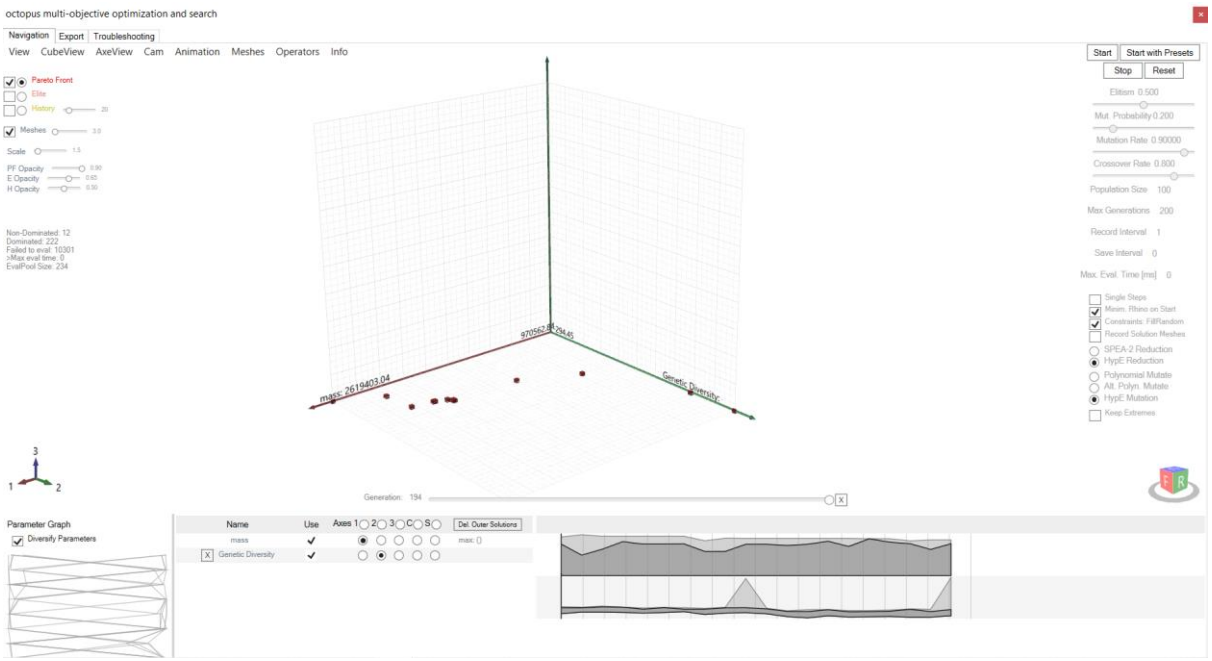


Figure 71 3D view

After the optimization process, four solutions are illustrated below.

1st Solution:

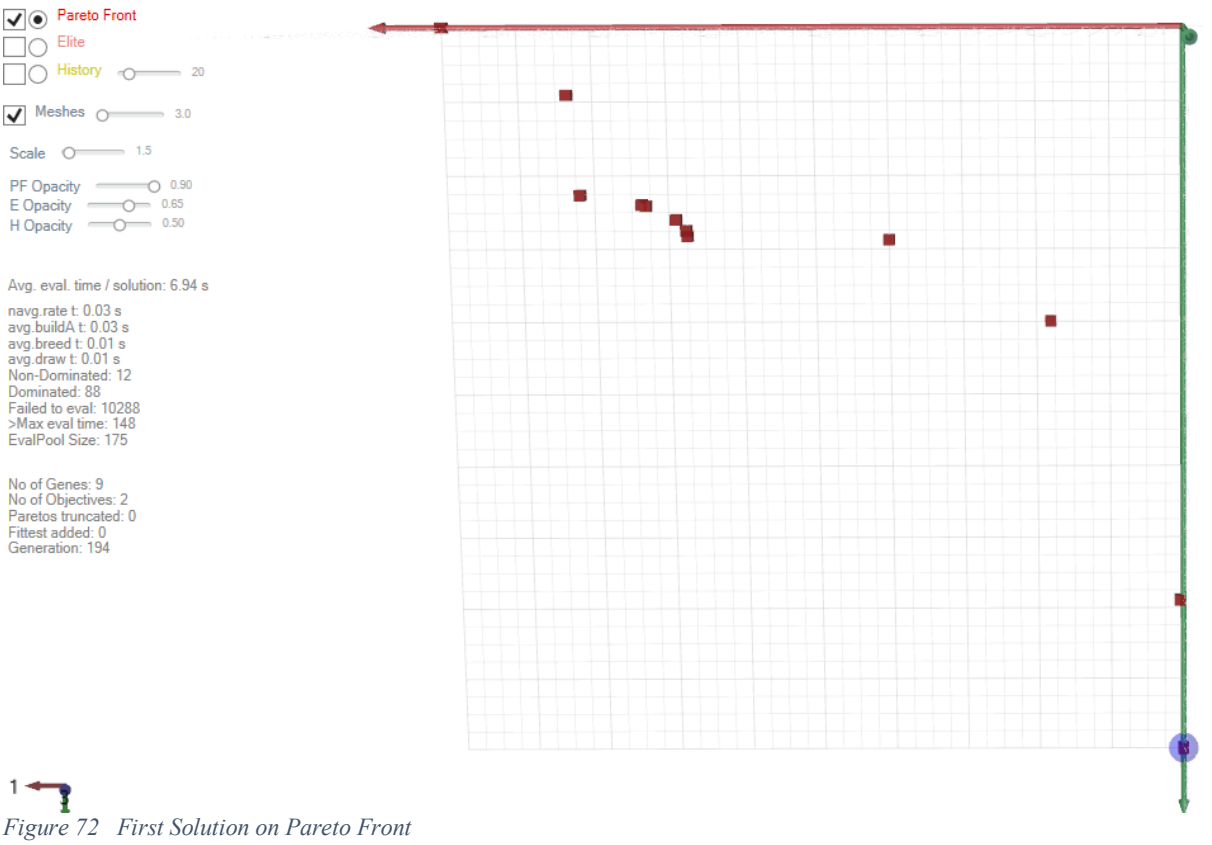


Figure 72 First Solution on Pareto Front

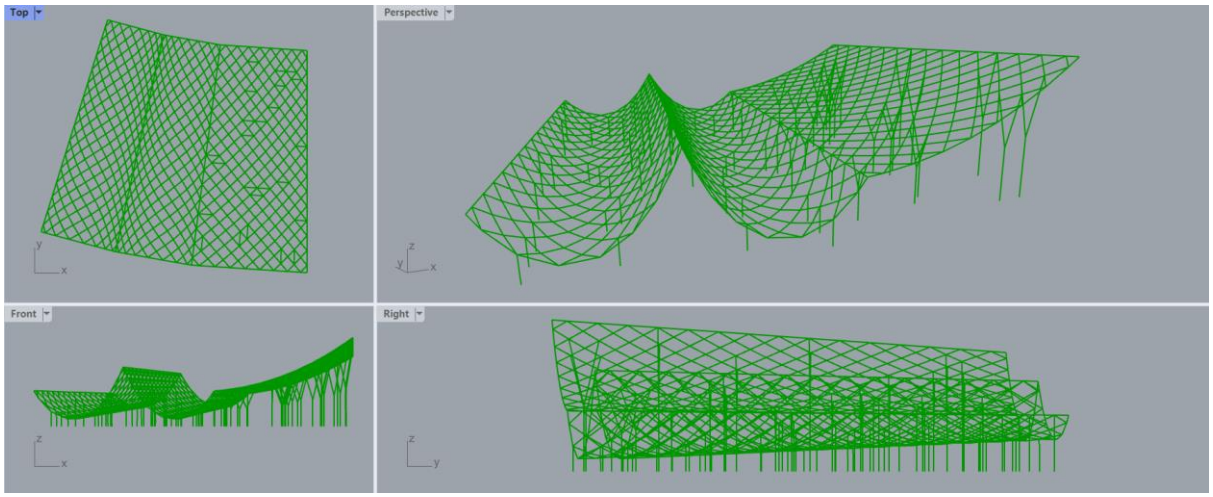


Figure 73 First solution

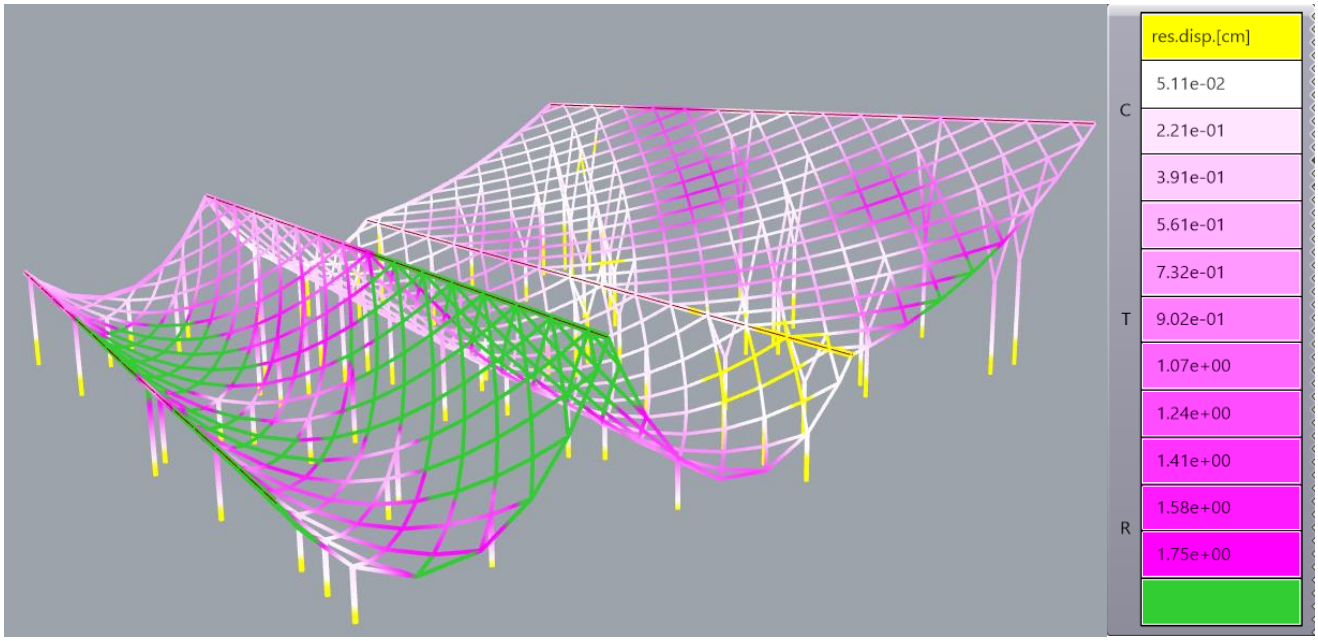


Figure 74 Residual displacement

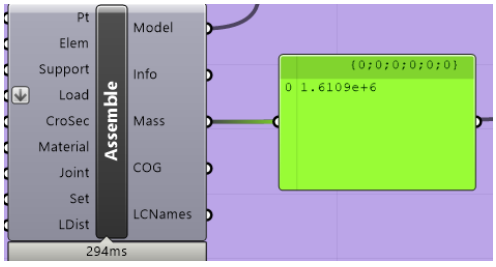


Figure 76 Mass

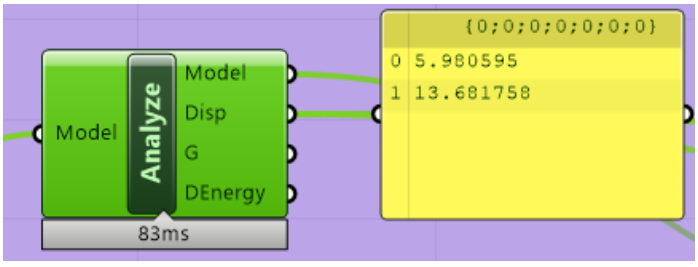


Figure 75 Displacement

Displacement: 13.7 [sm]
Total mass: 1.6e+6 [kg]

2nd Solution:

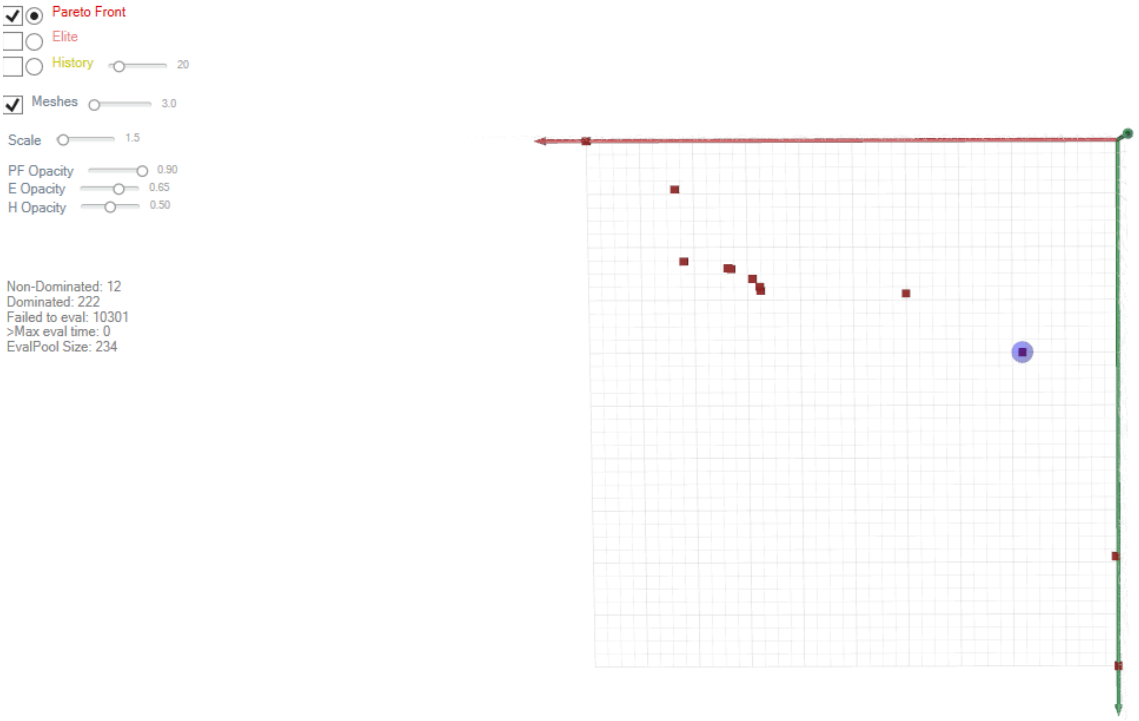


Figure 77 Second solution on Pareto Front

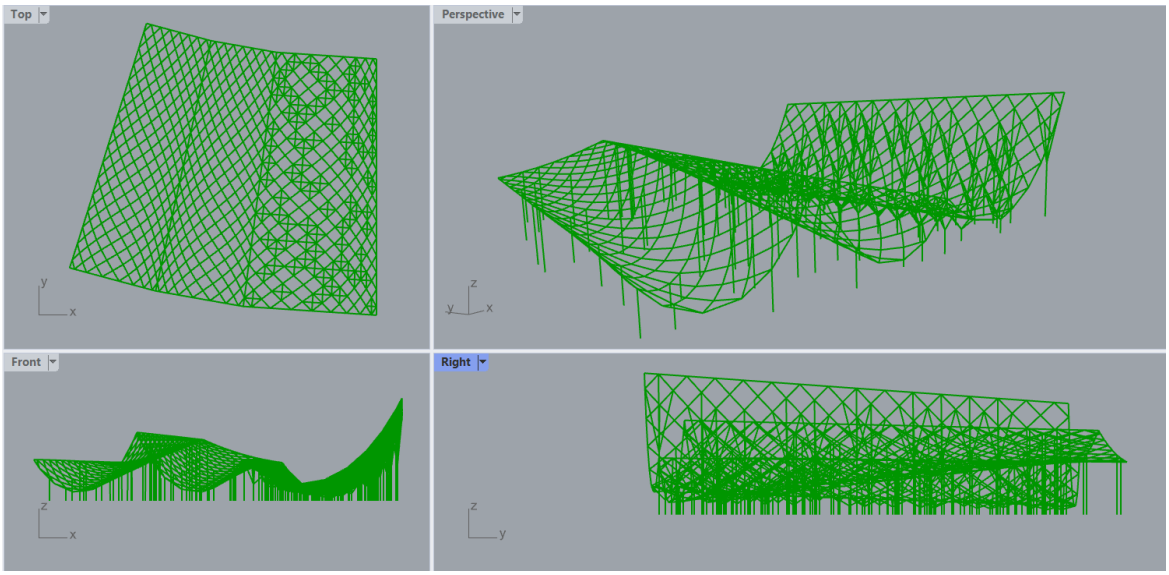


Figure 78 Second solution

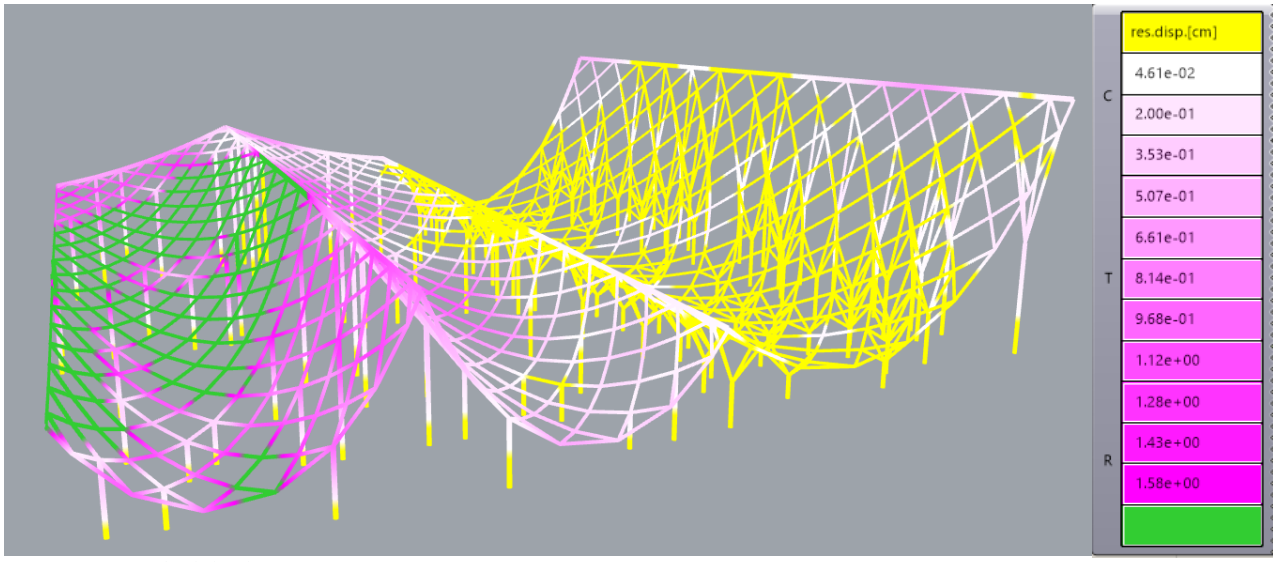


Figure 79 Residual displacement

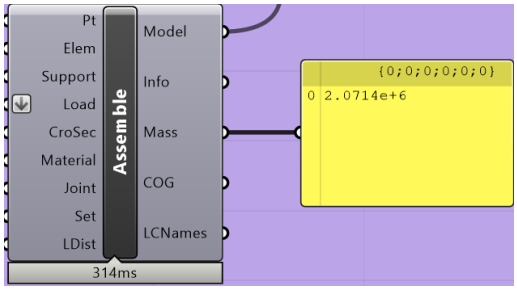


Figure 81 Mass

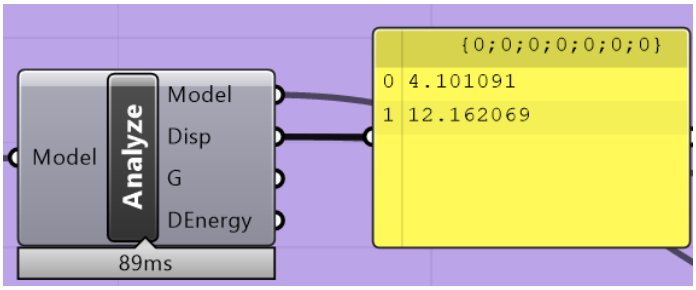


Figure 80 Displacement

Displacement: 12.16 [sm]
Total mass: 2e+6 [kg]

3rd Solution:

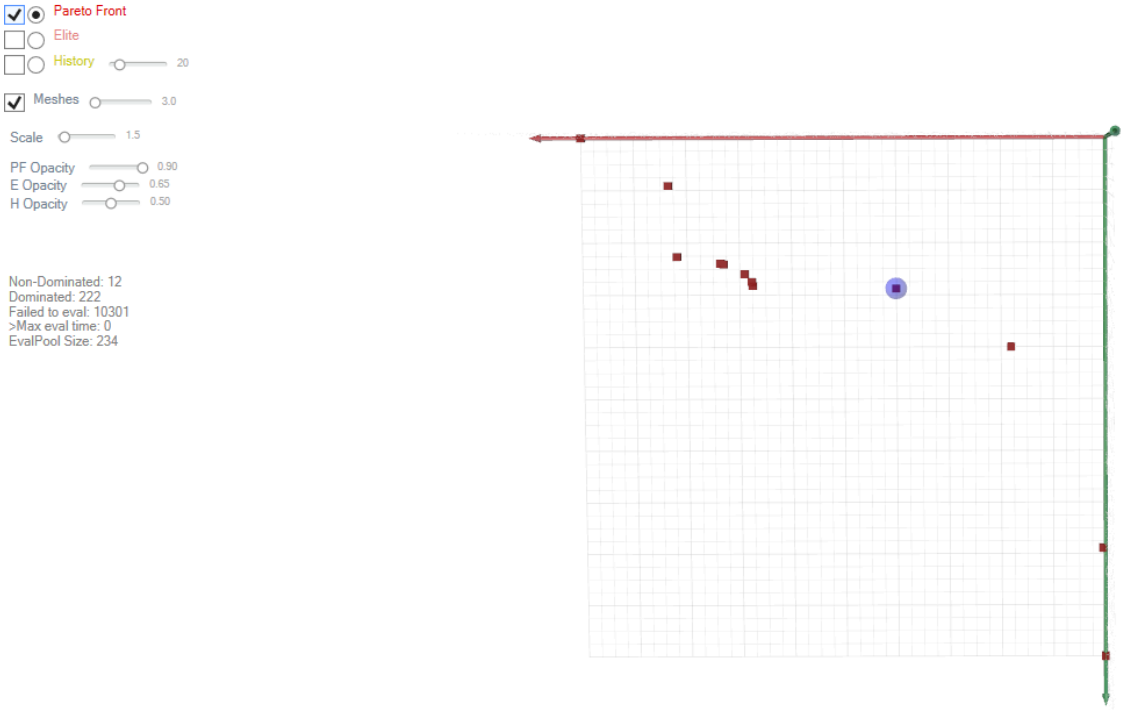


Figure 82 Third solution on Pareto Front

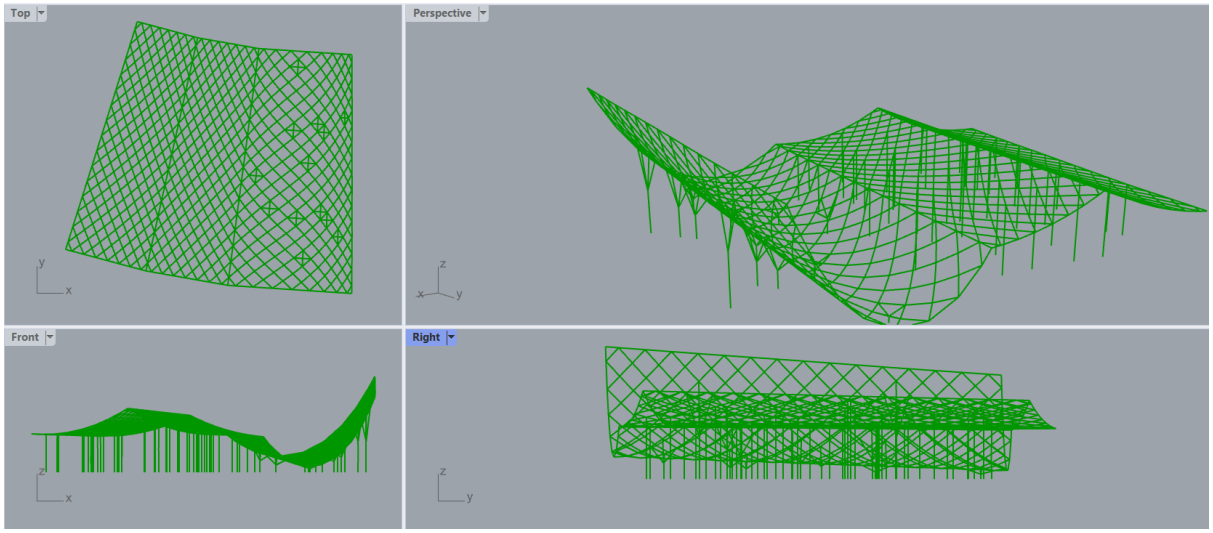


Figure 83 Third solution

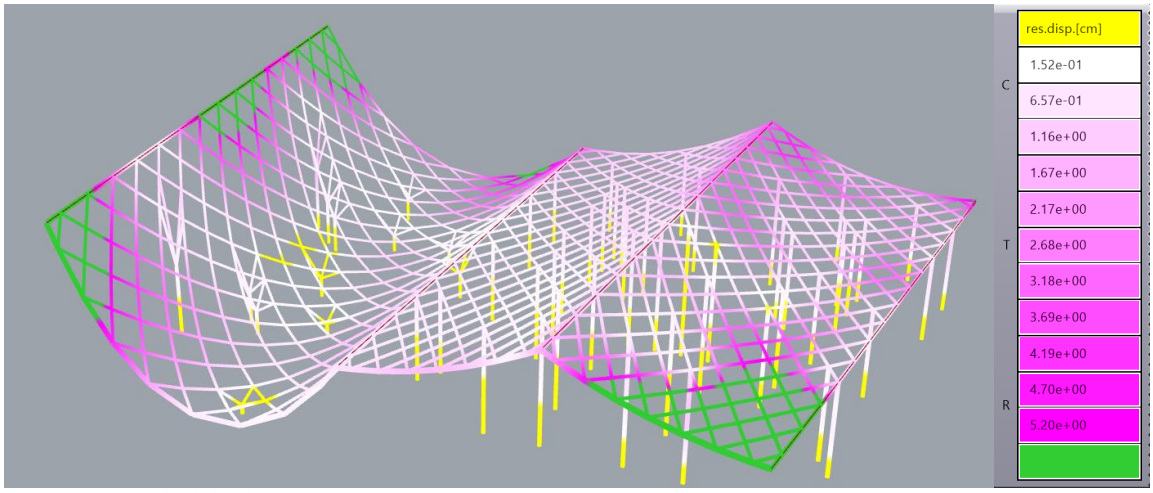


Figure 84 Residual displacement

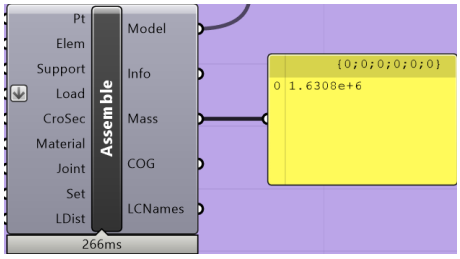


Figure 86 Mass

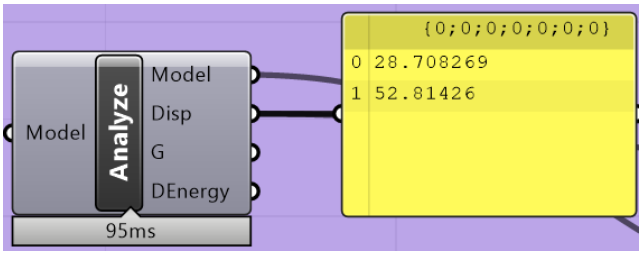


Figure 85 Displacement

Displacement: 52.8 [sm]
Total mass: 1.6e+6[kg]

4th Solution:

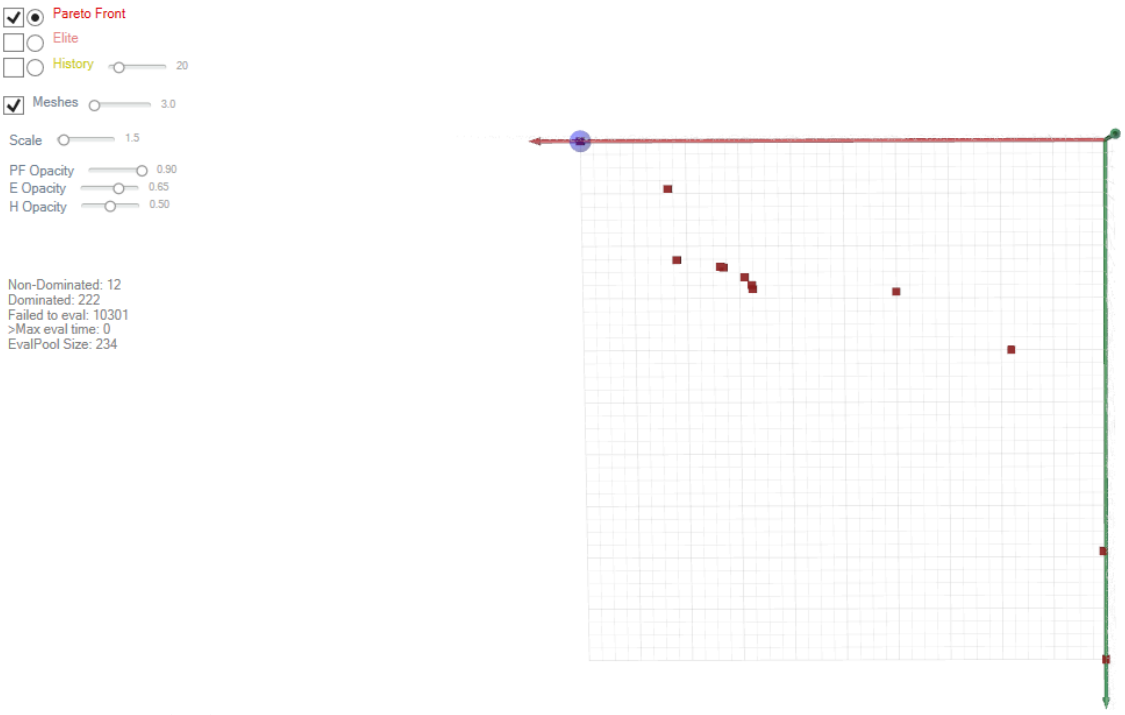


Figure 87 Fourth solution on Pareto Front

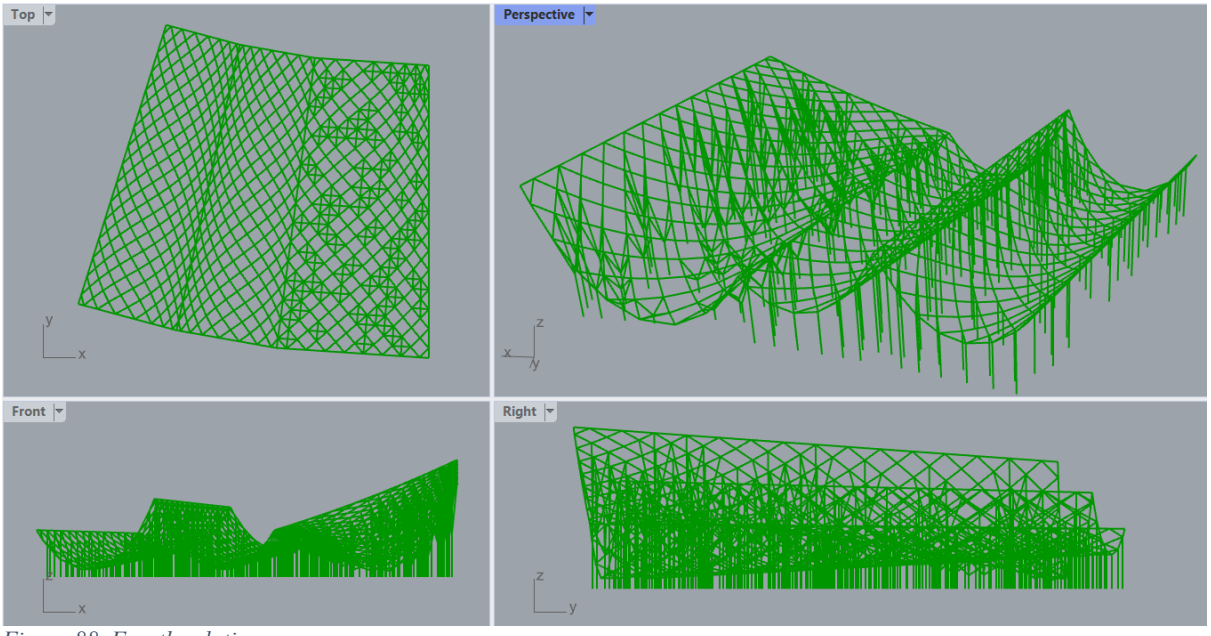


Figure 88 Fourth solution

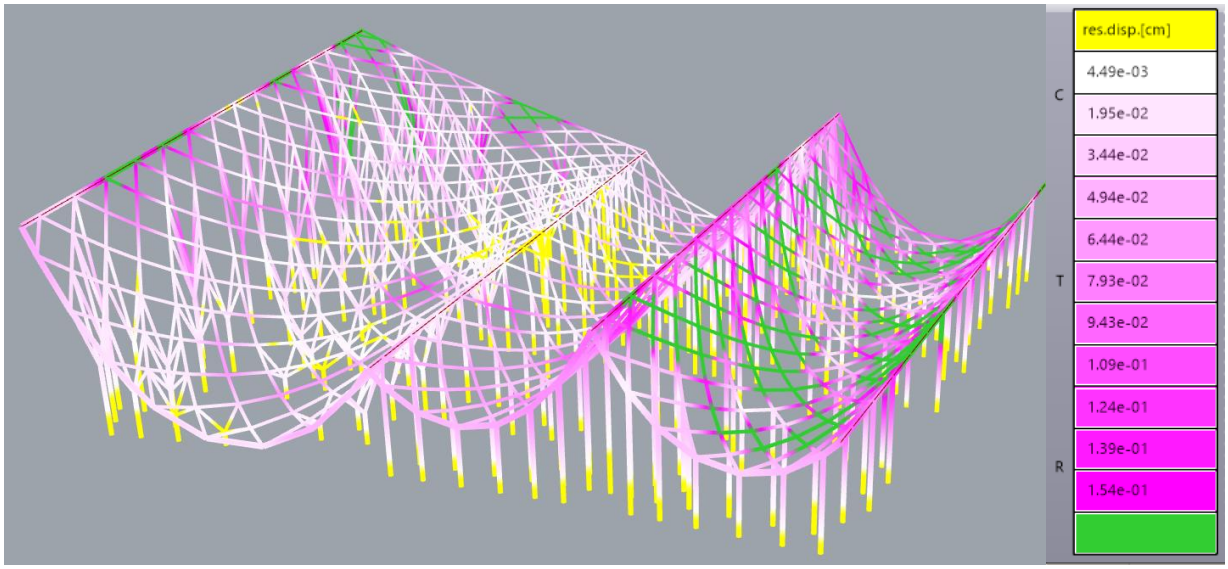


Figure 89 Residual displacement

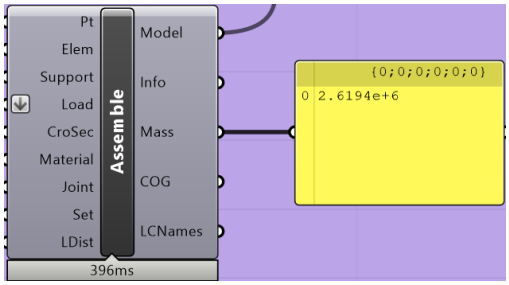


Figure 91 Mass

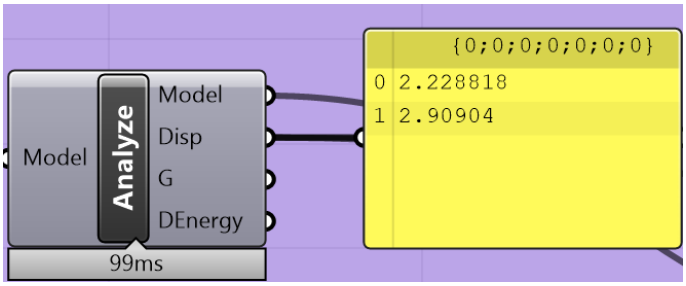


Figure 90 Displacement

Displacement: 2.9 [sm]
Total mass: 2.6e+6[kg]

8. Conclusion

The question this thesis addresses is the variability of design of gridshells and how optimal solution can be extracted among thousands of possibilities. Through the test case, the code is benchmarked and proven to be performing accurately. As a real-time structural analysis plug-in in a parametric environment, the user can easily interact with the model during run-time.

Modified designs are restricted to a fixed topology and thus can be written using a limited number of optimization variables. Topology optimization advances the size and shape optimization and gives no restrictions to the optimized structure. It simply tries to find the optimal domain of the governing equations contained within some design field. It provides minimum distribution of material in selected design space. Compared to size and shape optimization, topology optimization allows more freedom as no initial structure is required. Only the design space, the loads and the boundary conditions are required in order to find an optimized structure which satisfies the given restriction.

In this thesis, software RhinoCeros was used with the help of Grasshopper for parametric modelling, whereas Karamba3D and Octopus were used for Finite Element analysis and Optimization respectively. Latter two plug ins are just tool that have a capability

to help and work together with software which uses mathematical algorithms. The long span roof structure was quite big and variables inside parametric model were too much, which resulted in high time consumption. As a important lesson from this problem, was that identification of variables has to be clearly done before modelling.

9.Bibliography

T. Wortmann, "Efficient, Visual, and Interactive Architectural Design Optimization with Model-based Methods From Randomized Design Exploration to Design Space Cartography: Model-based Optimization for the Architectural Design Process View project Informed Design Group View project," 2018, doi: 10.13140/RG.2.2.15380.55685.

E. Poulsen, "Structural design and analysis of elastically bent gridshells The development of a numerical simulation tool."

F. Buonamici, M. Carfagni, R. Furferi, Y. Volpe, and L. Governi, "Generative design: An explorative study," *Computer-Aided Design and Applications*, vol. 18, no. 1, pp. 144–155, 2020, doi: 10.14733/cadaps.2021.144-155.

G. Cagdas and G. Varinlioglu, "An Alternative Approach to Structural Optimisation in Generative Design Serious Games View project Mission Antarctica View project," 2012. [Online]. Available: https://www.researchgate.net/publication/328661918

L. Mei and Q. Wang, "Structural optimization in civil engineering: A literature review," *Buildings*, vol. 11, no. 2. MDPI AG, pp. 1–28, Feb. 01, 2021. doi: 10.3390/buildings11020066.

C. Paoli, "Past and Future of Grid Shell Structures," 2007.

Alan. Blanc, Michael. McEvoy, and Roger. Plank, "Architecture and construction in steel," p. 619, 1993.

"Genetic Optimization in Building Lifecycle Analysis by Grasshopper."

"CHS Section properties-Dimensions and properties ® Section designation Dimensions and properties Mass per metre Area of section Ratio for local buckling Second moment of area Radius of gyration Elastic modulus Plastic modulus Torsional constants Surface area Per metre Per tonne." [Online]. Available: http://www.tatasteelconstruction.com/en_GB

E. Tyflopoulos, D. T. Flem, M. Steinert, and A. Olsen, “State of the art of generative design and topology optimization and potential research needs,” 2018.

N. D. Lagaros, K. M. Abdalla, G. C. Marano, M. C. Phocas, and R. al Rousan, “Optimization-Driven Architectural Design,” *Procedia Manufacturing*, vol. 44. Elsevier B.V., pp. 1–3, 2020. doi: 10.1016/j.promfg.2020.02.266.

N. R. M. Sakiyama, J. C. Carlo, L. Mazzaferro, and H. Garrecht, “Building optimization through a parametric design platform: Using sensitivity analysis to improve a radial-based algorithm performance,” *Sustainability (Switzerland)*, vol. 13, no. 10, May 2021, doi: 10.3390/su13105739.

“Generative design software will give designers ‘superpowers’ - 3ecruit.” https://3ecruit.eu/generative-design-software-will-give-designers-superpowers/ (accessed Feb. 01, 2022).

“The Generative Design Process – The Seven Key Stages That You Need to Know About - Archistar.” https://archistar.ai/blog/the-generative-design-process-the-seven-key-stages-that-you-need-to-know-about/ (accessed Feb. 01, 2022).

“What is Generative Design and Why it’s Important for Your Company | Cad Crowd.” https://www.cadcrowd.com/blog/what-is-generative-design-and-why-its-important-for-your-company/ (accessed Feb. 01, 2022).

“The Top 5 Buildings That Make Use of Parametric Design - Archistar.” https://archistar.ai/blog/the-top-5-buildings-that-make-use-of-parametric-design/ (accessed Feb. 02, 2022).

“Structural Optimization | Division of Solid Mechanics.” https://www.solid.lth.se/research/structural-optimization/ (accessed Feb. 02, 2022).

R. Aish and R. Woodbury, “Multi-level interaction in parametric design,” in *Lecture Notes in Computer Science*, 2005, vol. 3638, pp. 151–162. doi: 10.1007/11536482_13.

“BIM-Based multi-objective optimization process for energy and comfort simulation: existing tools analysis and workflow proposal on a case study by Zanchetta, Cecchini, and Bellotto.” http://www.insightcore.com/journal/bim_based_multi-objective_optimization_process_energy_comfort_simulation_existing_tools_analysis_workflow_proposal_case_study_zanchetta_cecchini_bellotto.html (accessed Feb. 02, 2022).

10.Acknowledgements

First and foremost, I would like to express my sincere gratitude to my supervisor Professor Giusseppe Carlo Marano for supporting me and showing me the correct way to accomplish this task.

My special thanks to my co-supervisor Laura Sardone who helped me in development of case study and did huge amount of contribution for finding correct materials and references for my thesis. Her guidance and advice carried me through all the stages of writing my project.

I would also like to acknowledge with much appreciation the crucial role of my family who supported me in all aspects of life, who is always my motivation for studies and who never stopped believing me.

Finally, thanks to god for giving me everything that I have.