A COMPUTATIONAL APPROACH FOR RENEWABLE ARCHITECTURE

MSc. Architecture - Building Technology Thesis Report

IDIL GUMRUK
4740297
i.gumruk@student.tudelft.nl

Delft University of Technology
Faculty of Architecture and Built Environment
Building Technology

A COMPUTATIONAL APPROACH FOR RENEWABLE ARCHITECTURE

A Generative Design Approach
Using Bioplastics and Earth

Master of Science (MSc) Thesis

Author

Idil Gumruk
4740297
i.gumruk@student.tudelft.nl
idilgumruk@gmail.com

Mentors

Dr. Ir. Pirouz Nourian
Architectural Engineering + Technology
Technical Design and Informatics

Dr. Ir. Fred Veer
Architectural Engineering + Technology
Structural Design & Mechanics

Sina Mostafavi
Architectural Engineering + Technology

PREFACE

This report covers the Idil Gumruk's graduation project as part of the Sustainable Graduation Studio graduation studio, with focus on Design Informatics and Structural Mechanics, for Master of Science in Building Technology at the Delft University of Technology.

I would like to thank all three of my mentors Pirouz Nourian, Fred Veer and Sina Mostafavi for their time and help during this process.

I would like to thank my roommate Sara Hammond for helping me with my model, proof reading my report and feeding me towards my deadlines.

I would like to thank Amey Thakur for helping me with my model and for always offering me inspiring but also comical philosophical conversations on life, environment, movies, books and music, with Soujanya Prasad and Pierre Kauter to get through this long and difficult year.

I would like to thank Claire Weil for always being a phone call away whenever I need her.

I would like to thank the entire class of 2019 Building Technology class for the amazing two years.

Finally, I would like to thank my mom Zeynep, my dad Ercument, my sister Elif, my aunt Sermin and my uncle Ali for doing so much from so far. Without them, I wouldn't be the person I am today.

ABSTRACT

Architecture has been using materials that are extremely durable, disregarding the lifespan and purpose of the structure. While this approach to materiality provides a high standard for structural stability and environmental control, it also causes a large volume of waste at the end of the life span of the building. While the material that's been used mostly cannot be recycled, reused or biodegrade; it also forced the next structure to extract new resources from the environment. Introducing environment friendly materials and utilizing these materials efficiently in the design process is critical as the consumption of natural resources is becoming dangerously high.

Computation has been used for optimization of form and shape for decades. This research attempts to understand environmentally friendly materials, mud and bioplastics, and develop a computational design method that will implement these new materials behaviour and optimizing their use of them in the design process

Keywords: earthy architecture, bioplastics, biodegradable, topology, optimization, structural analysis

TABLE OF CONTENTS

I. RESEARCH

I. I. INTRODUCTION

Climate change, depleting resources and increasing waste requires architecture to reconsider the way it uses its materials and their implementation into architectural design.

Historically humans were accustomed to use biodegradable materials such as mud, bamboo and timber for construction. These structures required frequent maintenance from occupants to last for longer periods of time. Over the years, we have moved away from these materials towards more durable materials, such as concrete and plastics or petrol-based preservatives to reduce the maintenance and increase the strength of our structures.

Whilst we have extended the life of our buildings, we have created materials that were too resistant to the natural processes. This has led us to large amounts of waste and depletion of resources.

If we reintroduce these biodegradable materials into the built environment again, how would the implementation of these materials into contemporary architecture and technology be?

I. II. PROBLEM STATEMENT

Non-renewable materials have been used in the construction industry redundantly for centuries. This behaviour resulted in a large amount of waste of raw materials and energy. By choosing renewable and ecological materials and using them efficiently and effectively, building industry can lower the environmental footprint, reduce waste and energy consumption. Construction industry needs to implement renewable materials, in optimized forms.

I. III. RESEARCH OBJECTIVES

The main objective is to develop a computational approach to optimize material uses of the construction materials -earth and bioplastics- in response to structural requirements.

I. IV. RESEARCH QUESTIONS

1. How to develop building unit forms for bioplastics and earth in consideration of the material properties?

2. How to compute an architectural form in consideration of the building units and material performance of the units?

3. How to optimize material use in the given design problem?

I. V. CASE STUDY

To implement the idea, a bus station in Rotterdam Centraal is proposed. While the site is chosen for its convenient location, the program was chosen more intentionally for its function and size.

A bus station doesn't require indoor climate control or complex envelope design. This simplicity enables the project to experiment with the material and computation more freely. The program of a bus station still requires frequent updates in its use, which makes it an appropriate program for a fully biodegradable structure.

Figure 1.1: The map of the site, the base image is taken from Google Maps and edited by the author to mark the proposed project site and the major landmarks around it.

I. VI. SCOPE AND LIMITATIONS

The main goal of this project is to develop a computational method that for a fully biodegradable bus station in Rotterdam Centraal that simulates growth over time in response to surrounding environment, including humans and climate.

In order to propose a realistic project, there will be some experimentation with the proposed materials and fabrication methods. The goal of the material and fabrication experimentation is limited to informing the computational model but not to suggest a new construction method.

While there will be suggestions of construction methods for the proposed design project, these methods will not go beyond the suggestions. Therefore, the configuration of the construction methods will not be included in the computational model and will not be within the scope.

While the physical and mechanical properties of bioplastics will be obtained through experimentation and structural tests, in the case of mud elements, the desired technical values will be obtained from the literature.

In conclusion, this project overall only covers the computational model and the basic development of bioplastic units that will generate the final form and simulate the overall growth of the structure. It does not cover the full realisation of the design.

It will only include a set of construction method suggestions that could be used but these suggestions should be taken only as suggestions.

Similarly, the material-fabrication study will only be constructed to inform the computational model. Therefore the data gathered from the material-fabrication study should not be used as a reference for any other project.

I. VII. RELEVANCE

Reintroducing biodegradable materials into architecture in the contemporary context, could help reduce waste created by the construction industry, and reduce the environmental footprint of architecture. In many cases, these biodegradable materials are also locally available which reduces the material cost of construction by minimising transportation and fabrication process. This aspect of these materials can contribute to not only environmental but also social sustainability.

II. LITERATURE REVIEW

II.I. EARTH

Earth is one of the oldest and most widely used building materials in the world. This type of construction includes clay, gravel, sand silt or other fireable soils[6]. This chapter is interested in earthy architecture that require gradual drying in ambient temperatures. The difference between explained earthy construction materials and other types is that they don't require post processing such as baking in high temperatures or compression like rammed earth or soil blocks [1].

There are several challenges when it comes to using earth as construction material. First, earth is not a standardized material and its properties differ based on site, as the composition of clay, silt, sand and aggregates changes for each location. Second, they lose volume as they are drying due to loss of water. This may result in cracks and weakened structural properties. There are strategies to minimise shrinkage. They will be further discussed on this chapter. Lastly, earth is not water-resistant, and various strategies should be present in the construction site to protect the material.

Earth, when it's not baked, can be reused infinite times by just adding water into it. Therefore, it creates zero waste at the end of its life-cycle. Furthermore, it reduces both the transportation and material costs, since it's usually readily available on site or could be brought to site from closer proximity than most industrial materials. Finally, mud protects organic building materials from bacteria, fungi and other insects as they reduce the water ratio in its environment[1].

II. I. I COMPOSITION

Earth contains clay, silt, sand and other aggregates such as gravel and stones.
The ideal percentages for mud brick is as follows [6]:

- 2-7% gravel
- 61-6% sand
- 22-32% silt
- 14-15% clay

Clay in earth functions as a binder like cement in concrete. Clay typically found in nature mixed with organic substances and chemical compounds particularly iron, manganese, lime and magnesium compounds. These different substances and compounds give clay different hues of colour between white, yellow, red and brown. Clay particles are usually formed around silicon or aluminium cores. The binding performance of clay depends on its silicon and aluminium content, as they create different chemical bonds. Silt, sand and gravel, different from clay don't have binding behaviour. They are simply infill for the clay[17].

The soil from depths of 40 cm of the surface typically has dead plants and humus. The earth needs to be free of these organic compounds to be used in architecture[17].

II. I. II STRUCTURAL BEHAVIOUR

"The tensile resistance of loam in a plastic state is termed its "binding force""[1]. This behaviour of earth is mostly related to its clay content and clay type. While this value can be measured anywhere between 25 to 500 g/cm², German building code does not consider blocks below 50g/cm² as construction material[1].

The compressive strength of earth depends on quantity and type of clay, grain size distribution, the preparation of the block and its compaction. The compressive strength of earth can differ between 5 to 50 kg/cm². The German standards allow between 3 and 5 kg/cm², which corresponds to a safety factor of 7 [1].On the other hand, earth virtually have no tensile strength.

The different elements in brick -aggregate and sand type and size, clay and silt percentage- play different role in brick's performance. Larger aggregate and sand particles increase the strength of the brick whereas higher silt and clay increase its water resistance, so more durable to erosion [6].

II. I. III BEHAVIOUR IN WATER

Interacting with water causes swelling and shrinkage in earth blocks. Swelling requires a large amount of water. There needs to be enough water in direct contact to shift earth's solid state to plastic state. Humidity would not cause swelling [1].

When the earth blocks are exposed to freezing temperatures, the water inside the earth increases in volume and causes cracks in the earth. This behaviour is mostly observed in earth with hairline cracks. Hairline cracks typically occur when the earth is clayey. On the other hand, rain erosion occurs more frequently in sandy earth. Therefore, while sandy earth has better frost resistance, the clayey earth is better for rain resistance [1].

During the drying process, to reduce shrinkage and cracks in earthy building elements, it is recommended to provide a slow and even drying process. This is best provided by sheltering the elements from direct sunlight and wind. Alternatively, reducing the clay and water content, including additives to the mixture and optimizing the grain size distribution could minimise shrinkage. The earth typically dries within 30 days under temperature of 23°C and 50% relative humidity in still air conditions, unlike concrete that does not dry even after 100 days under the same conditions [1].

In the case of not reaching the ideal percentages of ingredients in the soil for the desired purposes, it is recommended to use straw as a binder. Straw increases the strength of the brick while providing a better drying process for the brick. Alternative additives for binding and stabilizing mud bricks include, lime, bitumen, cement, cactus mucilage, oxblood, paper, corn husks, soda water glass, urine, blood, casein, animal glue and manure. Furthermore, while cactus mucilage can work as water repellent, high degree water protection can be achieved by double-boiled linseed oil. It is speculated that cooked starch and molasses may increase the stability. On the other hand, the manure can be used to repel insects [1],[6].

II. I. IV ADOBE

Mud bricks are traditionally used in hot-dry, subtropical and moderate climates [20]. Germany despite its colder climate who also adopted use of mud construction in the 6th century. Additionally, a different type of earth blocks -sod- was used in Scandinavia and England around 17th and 18th century. This method later was exported to the United States and Canada in the 18th and 19th century. Sod, as opposed to mud bricks, uses the top layer of soil including the grass growing on it [20]. Lastly, in New Mexico, silty soil blocks named terronis or terrones, extracted from riverbeds containing living plant roots were used for construction. This construction is still permitted in New Mexico [20].

Mud bricks are traditionally used in hot-dry, subtropical and moderate climates [11]. Germany despite its colder climate who also adopted use of mud construction in the 6th century. Additionally, a different type of earth blocks -sod- was used in Scandinavia and England around 17th and 18th century. This method later was exported to the United States and Canada in the 18th and 19th century. Sod, as opposed to mud bricks, uses the top layer of soil including the grass growing on it [11]. Lastly, in New Mexico, silty soil blocks named terronis or terrones, extracted from riverbeds containing living plant roots were used for construction. This construction is still permitted in New Mexico [11].

"Adobes are made either by filling moulds with a pasty loam mixture or by throwing moist lumps of earth into them" [11]. Adobe moulds are typically built using timber and can be varied in form. Some traditional moulds can be seen in Figure 2.1. The mud is thrown into the mould in order to reach the desired compaction for higher strength [11].

To increase compaction, a number of manual soil block press machines can be used. This technique has been used in Europe since the 18th century. Since then a variety of presses were developed. An example of this machinery can be seen in Figure 2.2. These machines increase the structural performance of the produced bricks

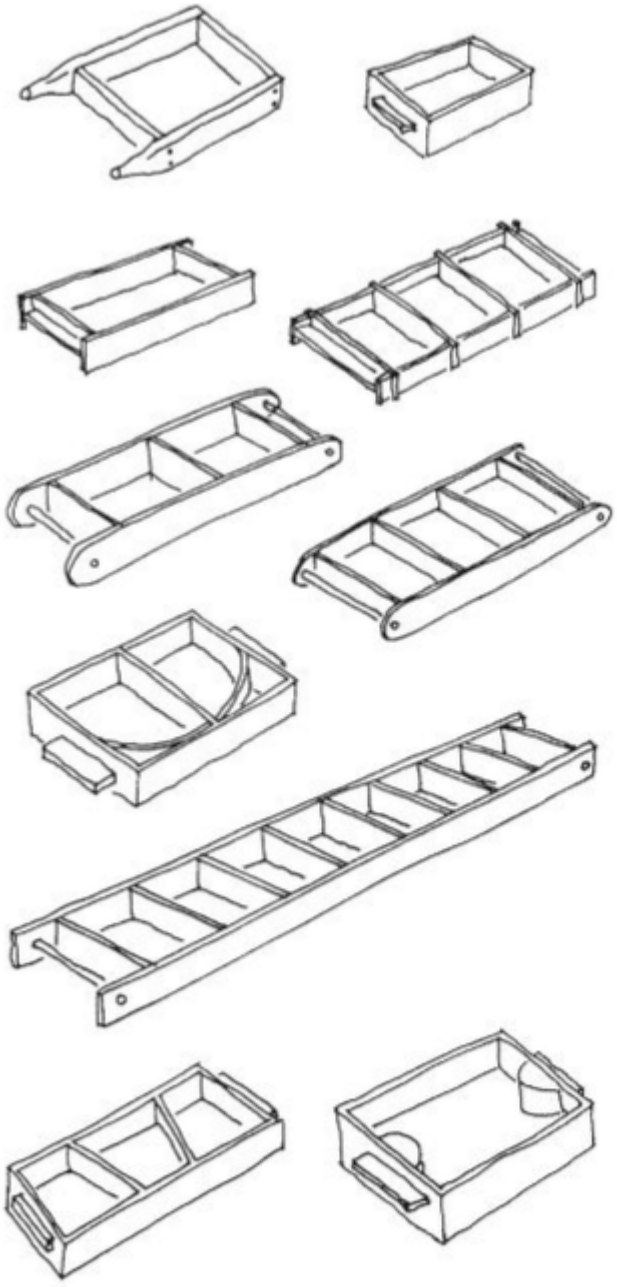


Figure 2.1: Typical moulds examples for adobe [1]

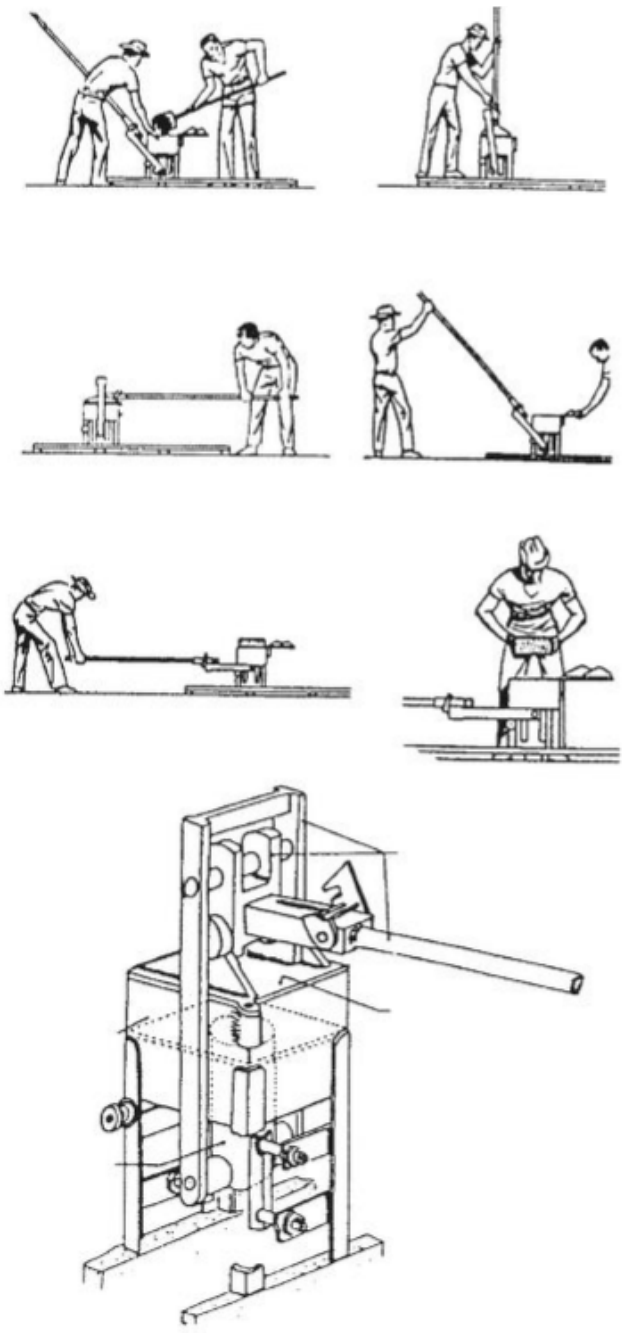


Figure 2.2:"The best-known press worldwide in the CINVA Ram, developed in Colombia by the Chilean engineer Ramirez"[1]

and enable soil with lower water content to be used during the process. As a result, the drying process is significantly reduced, and bricks are ready to be used in construction right after it's been pressed. However, the pace of brick production reduces about 20% less and the blocks need 4-8% cement content to accommodate for lower water content. Otherwise, their compressive strength is not as high as handmade ones [11]. Furthermore, in order to maintain the size of the bricks and structural behaviour consistent, the soil mix needs to be at a "constant level of moisture and composition" [11]. Alternative to manual soil block press machines, there are a number of fully automatic block making presses that is able to produce bricks at a much higher pace. However, these machines have much higher costs and require initial investments [11].

Different kinds of mortar can be used during the construction of earth blocks. These include loam mortar, hydraulic lime mortar or high-hydraulic lime mortar. To prevent shrinkage while drying the mortar, sufficient amount of course sand should be used, while keeping the clay content to 4-10%. Alternatively, if the surfaces of bricks that are going to face each other are wet enough, the use of mortar may not be necessary, however this technique requires higher craftsmanship [11].

PROTECTION

There are a number of techniques to protect mud surfaces from environmental impacts such as wind and rain erosion without using additives in the soil.

Consolidating a mud surface when it's still moist and pliable with a help of metal trowel until the surface is shiny with no pores and visible cracks increases weather resistance remarkably.

Painting the surface is another way to protect it from the environment. However, the paint needs to be chosen carefully and should be reapplied regularly. The paint needs to be water-repellent and porous to allow water diffusion to outside. A list of different ways of lime paint applications are discussed in [20].

In addition to the previous protections, a number of water repellents can be used for surface treatment. These are

- Silane and siloxanes
- Polysiloxanes (silicone resins)
- Siliconates
- Acrylic resins
- Silicate ester with hydrophobising additives
- Silicates with hydrophobising additives [20].

In addition to these lime plaster, shingles, planks and other covers are recommended to protect the loam surfaces from environmental erosion.

II.II BIOPLASTICS

II. II. I COMMERCIAL BIOPLASTICS

Bioplastics is a group of plastics with physical and mechanical properties. "According to European Bioplastics, a plastic material is defined as a bioplastic if it is either biobased, biodegradable, or features both properties"[16]. Therefore a plastic could be considered bioplastic even if it is not biodegradable or contain fossil-based components inside.

Based on this definition, we can divide bioplastics into three categories. Biobased PE, PET, PA and PTT are bioplastics even though they are not biodegradable. PBAT and PCL is biodegradable even though they are fossil-based. Only PLA, PHA or PBS and starch blends are both biobased and biodegradable plastics. For comparison please refer to Figure 2.3.

Among the three groups only the biobased biodegradable plastics worth mentioning its pros and cons of the overall production and end of life cycle processes, given the main objective of this project. These materials include starch blends, PLA or PHA and are commercially used for products like packaging.

II. II. II BIODEGRADATION

It is important to understand the concept of biodegradability in this context. Most things are technically biodegradable, but to consider something to be biodegradable it is important to know the rate of degradation and the conditions it needs to biodegrade.

The biodegradation of PLA can take place in an industrial composter at 60°C in about 40 days. "However at lower temperatures and/or lower humidities, the storage stability of PLA products is considered to be acceptable"[19]. As a result, biodegradable plastics do not degrade in nature [14].

II. II. III ALTERNATIVE BIOPLASTICS

In response to the commercial bioplastics, some researchers started looking into alternatives for bioplastics that could be derived from natural materials. Using gelatine, water, glycerine, agar and corn starch in different combinations and

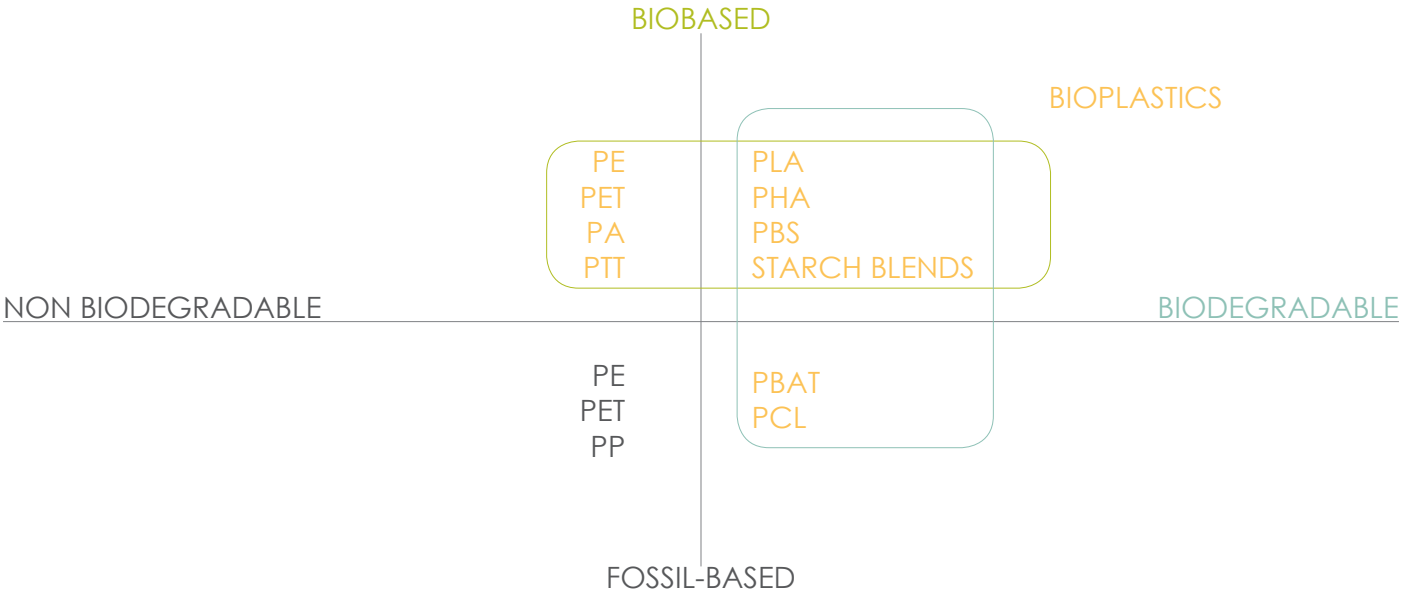


Figure 2.3: Bioplastic chart based on biobased and biodegradability of traditional plastics. Redrawn by the author based on the graph provided in [14]

ratios, one can make bioplastics in different ratios. A variety of additives can be used in addition to the base bioplastics. Coffee grounds, leaves, egg shells, clay, spirulina & sugar mixture, vinegar, soap, burlap, hemp, are some of the additives that can influence the behaviour of plastics. (Kwong Oi-Ying, 2011; Margaret Dunne, 2018; Viladrich, 2014)

There are a series of advantages to these bioplastics. They have a fairly low melting point which make them easier to experiment, melt and reshape. They don't contain any toxic materials. However, there are also a series of disadvantages. They are water-soluble, highly flammable and not resistant to sunlight. Since they are entirely made out of organic material, they may grow mould if not ventilated well enough. A series of organic protection methods could be used to counteract these disadvantages.

RECIPES

There are a large collection of recipes that can be found in the [5], [4], [21]. The recipes I will mention here will be the ones I will experiment with for this project. These recipes are chosen for their suitability for this project's purpose, based on the process and product described by their creators of the recipes.

Ingredients	Brittle			Flexible
Glycerine (g)	0,0	1,8	3,6	7,2
Water (ml)	60	60	60	60
Gelatine (g)	12	12	12	12

Table 1: Bioplastic ingredient ratios based on the material's flexibility drawn by author based on [4]

There are a large collection of recipes that can be found in the [5], [4], [21]. The recipes I will mention here will be the ones I will experiment with for this project. These recipes are chosen for their suitability for this project's purpose, based on the process and product described by their creators of the recipes.

The recipe requires three ingredients; glycerine, water and gelatine. According to the instructions, while gelatine needs to be animal-based, glycerine could be either animal or plant-based. Necessary tools include electric stove, pan, spoon, scale and a measuring cup.

To create a bioplastic piece, one needs to combine the ingredients in a pan and heat it up on an electric stove until the mixture is thickens. The mixture needs to be stirred as it's heating up, Once the mixture reaches to a desired consistency, it should be poured into a mould. Mould needs to be preferably silicone, plastic, glass or metal to prevent sticking. Timber or paper is not desirable. The plastic needs to spend a day in the mould, then needs to be hanged in a frame to dry for another day or two to cure in order to maintain its desired form [4].

II. III OPTIMIZATION

There are three kinds of structural optimization methods; size, shape and topology [12], [13], [22]. Size optimization used when the component's overall form and loads are known but the thickness and sizing of the elements that makes up the given larger element is unknown. Shape optimization is needed when "the form or contour of some part of the boundary of a structural domain" [13] is unknown. Topology optimization is used when the shape or size of the structure is unknown [13].

II. III. I TOPOLOGY OPTIMIZATION

"The purpose of topology optimization is to find the optimal lay-out of a structure within a specified region"[12]. There are several ways to perform topological optimization. They each have their own advantages and disadvantages [13]. divides these methods into two categories: "Optimality Criteria methods and Heuristic or Intuitive methods". This division comes from the way the optimization methods behave. Optimality Criteria methods are more rigorous and "suitable for problems with a large number of design variables and a few constraints" [13]. Heuristic methods on the other hand are derived from intuition and observation of existing processes and systems. They provide efficient but not necessarily optimal solutions [13]. Please refer to Table 2 for the list of optimization methods under these two categories.

Among the eleven methods that were presented in Table 2, this section will focus only four of these: Computer-Aided Optimization (CAO) and Soft Kill Optimization (SKO) in depth. In addition to these two, Evolutionary Structural Optimization (ESO) and Bidirectional ESO (BESO) will be discussed briefly. Furthermore, it is important to mention that even though it will not be covered in this chapter, O. Sigmund [8] provides a 99 line code for Solid Isotropic Material with Penalization (SIMP) on MATLAB which could be beneficial to look into if necessary. The reason why these four methods were chosen among eleven was intuitiveness of the methods and easy implementation of them.

CAO and SKO are two methods developed by Claus Mattheck and well explained in his publications. Since this method is more intuitive than the others, it could be applied to this project using the existing plugins (Karamba 3D) that are readily available.

ESO and BESO method is already implemented in Karamba 3D. Understanding this method could inform the design process and could be applied in this project.

A 99 line MATLAB code was presented in [8], [12] using SIMP method. While this code was created for a simple element, the code itself can be modified to be used for any desired problem. The easy access to this code and its explanation makes it a viable method for this project.

OPTIMALITY CRITERIA METHODS	HEURISTIC OR INTUITIVE METHODS
Homogenization	Fully Stressed Design
Solid Isotropic Material with Panelization(SIMP)	Computer-Aided Optimization (CAO)
Level Set Method	Soft Kill Option (SKO)
Growth Method for Truss Structures	Evolutionary Structural Optimization (ESO)
	Bidirectional ESO (BESO)
	Sequential Element Rejection & Admission (SERA)
	Isolines / Isosurfaces Topology Design (ITD)

Table 2: Optimization methods categorized based on (Querin et al., 2017)

II. III. I. I SKO & CAO

Computer Aided Optimization (CAO) and Soft Kill Optimization (SKO) are two different approachs to optimization that are used together to create an optimized form. These methods are derived from the optimization method of trees and bones. While CAO imitates the way, trees reinforce their structures by adding more material to support, SKO imitates the way bones subtract material from areas that are not under stress. In both biological phenomena, the optimization is not limited to addition or subtraction of materials, but also location specific material mineralization and its mechanical properties. [18]. The desired workflow using CAO and SKO is demonstrated in Table 5.

Table 5: Workflow between SKO and CAO (redrawn by the author based on [18]

SKO

SKO method is derived from optimization mechanism of bones. Higher stress areas have higher degree of mineralization -stiffer materials- than lower stress areas. This enables to have a matrix with varied Young's modulus based on the load case applied to the element [18].

To achieve this smart heterogenous material matrix computationally [15], [18] proposes using Young's modulus as a function of the temperature. "The Young's modulus/temperature relation can be defined as an open polygon by

$$\bar{T} = \frac{1}{m} \sum_{i} T^{(i)} \qquad (2)$$

giving moduli at several different temperatures" [18]. The temperature T = 100 (temperature in arbitrary units) was set to the maximum Young's modulus E_{max} and temperature T = 0 to the minimum value E_{min}." [18] Using equation 2 [18], the FEM software averages temperature of all vertices of the mesh.

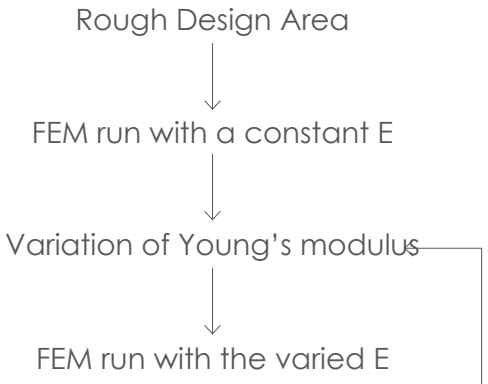


Table 6: Soft Kill procedure (redrawn by the author based on [18]

[18] describes this procedures as described in Table 6.

1. As described in Table 6, the rough design area goes through a FEM run to calculate the stresses based on the load cases applied to it. It has a constant Young's modulus within this given rough design area.
2. After the first run, the previously constant Young's modulus is changed in response to the stresses in each node. With the new locally defined Young's modulus values, a new FEM run applied to the rough design area.
3. Based on the second FEM, a new set of Young's modulus assigned to each vertex.

Second and third steps are repeated until the model is divided into two extreme Young's moduli, which can represent either a material and void or two separate materials. The young's modulus that represent the void is eventually gets "soft killed"[18].

Following this procedure, after an optimum topology is achieved in the design region, a CAO can be applied to the topology for shape optimization [18].

The calculated stress and the new Young's modulus are directly related. New young's modulus could be calculated using equation 3 [18].

$$E_{n+1} = \sigma_n \quad (3)$$

In order to the incrementally change the Young's modulus in relation to the calculated stress level, equation 4 [18] was used. Equation 4 is essentially the implementation of equation 3 [18].

$$E_{n+1} = E_n + k(\sigma_n - \sigma_{n-1}) \quad (4)$$

Constant k controls the shift of the value in relation to the stress change. Constant k has to be larger than 1 [18].

An alternative to equation 4 is to use a predetermined reference stress value as σ_{ref}, as seen on equation 5 [18]. This could be determined manually or referenced to a specific vertex in the model. Using equation 5, it is also important to maintain the limits for the Young's modulus to remain in the desired range [18].

$$E_{n+1} = E_n + k(\sigma_n - \sigma_{ref}) \quad (5)$$

It is recommended to use E_{min} as E_{max} / *1000* to give satisfactory stress level for void. As mentioned before, SKO actually converts these stressed into temperature terms to run the FEM. In order implement equation 5 into a thermal FEM equation 6 [18] has been used. [18]

$$T^{(i)}_{n+1} = T^{(i)}_n + k(\sigma^{(i)}_n - \sigma_{ref}) \quad (6)$$

As mentioned earlier, similar to SKO, CAO also uses stress and temperature for its calculations. Stress - temperature relationship is similar, there are some contrasts between the two. Their σref are not equal to each other, because in the case of SKO, the boundaries of the mesh are still inside the rough design area mesh. Therefore, the boundary conditions are treated the same as non-boundary conditions. As this is not the case in CAO, σref for CAO is half of σref of SKO. Lastly, σref in either method should consider a lower value than the desired Young's modulus. As the iterations will lower the average below the reference [18].

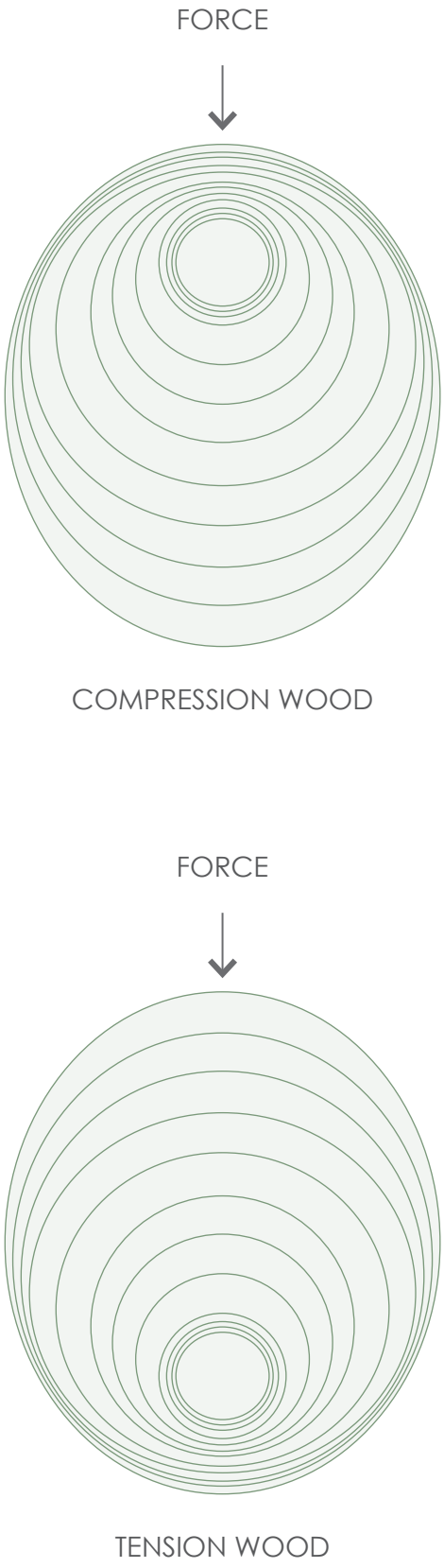
CAO

CAO method uses the concept of adaptive growth in trees and other plants. Adaptive growth is the growth that occurs in weaker parts of branches to reinforce the branch. This growth does not only occur as a volume but also mineralization of the area. A visual representation of adaptive growth can be seen in Figure 2.4.

Using the FEM of the initial design (design after SKO), we will receive a stress, strain and displacement value for each vertex. Using equation 7 [15], we can also calculate Mises reference stresses. This helps us find where the local weaknesses are. These are the areas where adaptive growth will occur. [15]

$$\sigma_{mises} = \frac{1}{\sqrt{2}}\sqrt{(\sigma_1 - \sigma_2)^2 + (\sigma_2 - \sigma_3)^2 + (\sigma_3 - \sigma_1)^2} \quad (7)$$

The computed Mises stresses will determine the temperature distribution on the FEM element. This temperature map is not related to any real-life heat or temperature measures, but just a method to imitate adaptive growth. When the temperature distribution mapped on to the element, places with the highest temperature will correspond to the parts that has the highest stresses. Following this temperature map, we also set the material to have approximately 1/400 of the actual value for its modulus of elasticity. [15]



In the new FE model with the new temperature map as thermal load and reduced modulus of elasticity, the topmost layer of the mesh is also assigned to have a thermal expansion factor of a, which needs to be greater than 0. With the top layer of low modulus of elasticity and thermal expansion of a, the layer will expand in reaction to the heat map created based on the mises stresses. This expansion is calculated using equation 8 [15]. The effect of equation 8 can be increased via a constant [15].

$$\Delta l = l_0 \cdot \alpha \cdot (T - T_{ref}) \quad (8)$$

After allowing thermal expansion and allowing the adaptive growth, the original modulus of elasticity, young's modulus and initial load cases would be applied to the new geometry. A new FE analysis would show the improved stresses on the new geometry. If the new geometry would satisfy the expectation, it would be the final optimized form, otherwise thermal expansion method coul be applied until the desired results were achieved [15].

STRESSES

After allowing thermal expansion and allowing the adaptive growth, the original modulus of elasticity, young's modulus and initial load cases would be applied to the new geometry. A new FE analysis would show the improved stresses on the new geometry. If the new geometry would satisfy the expectation, it would be the final optimized form, otherwise thermal expansion method coul be applied until the desired results were achieved [15].

Figure 2.4: Adaptive growth redrawn from [15]

II. III. I. II ESO & BESO

Hard-kill optimization methods specifically add or subtract finite number of elements to the given design area in each iteration based on heuristic criteria. Evolutionary Structural Optimization is the best known hard-kill method of topology optimization [2]. The main characteristics of hard-kill methods is that the result is not in terms of the material properties but instead either 1 or 0, material or void. This creates a clear boundary of the optimized structure[2].

The first proposed ESO method was only developed to remove materials, and assuming that the remaining materials should have the same and safe stress levels globally. This idea is achieved by using Mises stress level of a given rough design area. To achieve this outcome, a rejection ratio (RR_i) needs to be determined based on the equation 9 [2].

$$\frac{\sigma_e^{vm}}{\sigma_{max}^{vm}} < RR_i \tag{9}$$

In addition to RR_i, in order to optimize displacement and maximize stiffness, equation 10 [2] was implemented to ESO, where "α_i^e is the sensitivity number of element i, u_i is the displacement vector and K_i is the stiffness matrix for the element i"[2].

$$\alpha_i^e = \frac{1}{2} u_i^T \cdot K_i \cdot u_i \tag{10}$$

An alternative to subtractive ESO was additive ESO (AESO), where the material was added to the basic structure. This leads to the early version of bi-directional ESO (BESO), where materials can be both added and subtracted from the given rough design area. This approach was able to consider both stress-based and stiffness/displacement criterion. One of the main challenges of this approach is the checkerboard problem. In order to overcome this problem, equation 11 [2] was used for mesh dependency filter by averaging the sensitivity numbers of the adjacent nodes [2].

$$\alpha_j^n = \sum_{i=1}^{M} w_i \alpha_i^e \tag{11}$$

In this w_i is the weight factor that's defined by equation 12 [2]. r_{ij} is the distance to the adjacent node [2].

$$w_i = \frac{1}{M-1} \left(1 - \frac{r_{ij}}{\sum_{i=1}^{M} r_{ij}} \right) \tag{12}$$

Finally the mesh dependency filter is calculated by equation 13 [2], "where K is the total number of nodes to inside the filter domain with radius r_{min} and $w(r_{ij})$ is the linear weight factor"[2] calculated by equation 13 [2].

$$\alpha_i = \frac{\sum_{j=1}^{K} w(r_{ij}) \alpha_j^n}{\sum_{j=1}^{K} w(r_{ij})} \tag{13}$$

II. III. II. STRESSES & TOPOLOGY OPTIMIZATION

Von Mises stresses, as shown in equation 7, always result in positive value [25]. While this makes von Mises values applicable for metals and plastics, where the material behaves similarly both in tensions and compression. For this project, where the main structural material is adobe -a material that has virtually has no tensile strength- instead of using von Mises stresses for analysis, normal stresses will be analyzed and used for the topology optimization.

III. MATERIAL STUDY

III.I FABRICATION

III. I. I INTRODUCTION

In order to understand the structural strength and overall material behaviour, and decide on a recipe that will suit the building unit the best, we decided to have a tensile strength test to a series of different ratios of bioplastics. This chapeter will cover the specimen creation process, the test results and the conclusions drawn from the tests.

III. I. II MATERIAL TESTING REQUIREMENTS

Tensile strength test was necessary to understand the behaviour of the bioplastics and to further design the building units out of it. To test the tensile and yield test, a series of samples needed to be produced. The desired form of the sample was as drawn in Figure 3.1. The specimen's dimensions should be as show in the diagram and the thickness should be somewhere between 2-4 mm.

III. I. III MOULD DESIGN

Based on the literature research the mould needs to have a smooth surface to be removed easily. Given the condition, a vivak mould seemed to be easier, faster and cheaper way to create the mould.

First, the required shape was manually cut from 4 mm Medium-density fibreboard (MDF). Then it's negative was created through vacuum forming for vivak. Another sheet of vivak was used as a cover for the mould. In order to keep the mould flat and prevent bending during the curing process, enough weight was placed on top of the mould once the mould was filled with bioplastic.

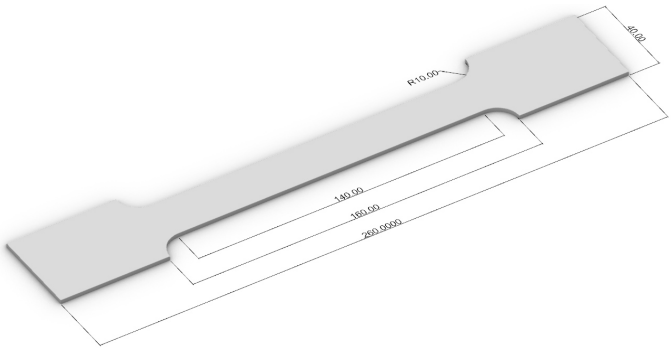


Figure 3.1: Specimen dimensions

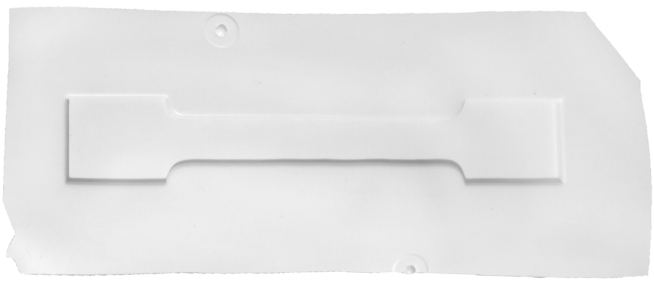


Figure 3.2: Vivak mould for bioplastics

III. I. IV INSTRUCTIONS

1

2

3

4

5

6

7

8

9

10

Figure 3.3: Tools and ingredients used.

1. Jacob Hooy brand beef-based gelatine powder.
2. La Saponaria brand vegetable-based glycerine
3. Tap water [2]
4. Kitchen scale
5. Metal pan
6. Wooden spoon
7. Coffee grounds
8. Egg shells
9. Electric stove [3]
10. Paper clip

01. Place the metal pan on the kitchen scale and start the kitchen scale. Make sure the scale reads 0.0 kg.

02. Add the necessary amount of water, glycerine and gelatine, in this order. If the additives are included in the recipe add the additives into the mixture as well.

03. Place the metal pan on the stove and start heating the pan in medium heat -from the settings between 0-9 on the given stove set it to 4.

04. Constantly stir the mixture with a wooden spoon as it heats up.

05. Once the mixture thickens -depending on the mixture this time can take sometime between 9-19.5 mins. (see Table 3).

06. Once the required time is done pour the mixture into the mould and cover the lid immidiately, before the mixture starts to cool down.

III. I. V INITIAL ATTEMPTS

SPECIMEN 1:

The specimen 1 is made out of 2 g of glycerine, 60 ml of water and 12 g of gelatine. This mixture was heated until the mixture was thickened and poured into the mould. The mould was covered with a plastic bag and some weight. This cover caused the specimen to have unintentional grooves and imperfections on the surface. The next day the specimen was removed from the mould and placed on a table, left to dry and cure. Within hours the specimen started deforming. It is observed that the deformation follows the imperfections and the grooves that was caused by the use of plastic bag as a cover. Furthermore, the specimen was behaving more flexible than the desired performance.

SPECIMEN 2:

The specimen 2 is made out of 1 g of glycerine 60 ml of water and 13 g of gelatine. The recipe was changed to have less glycerine and more gelatine. Based on the literature research on Section II. II. III, glycerine ratio influences the flexibility of the sample. This mixture was heated until the mixture was thickened and poured into the mould. The mould was covered with another sheet of vivak and some weight. This covering method prevented unwanted grooves and imperfections that occurred in Specimen 1. The next day the specimen was removed from the mould. An uneven amount of extra material was creeped out of the mould while drying inside the mould. This extra material was used to attach the clips onto the specimen and hanged from a rod. A significant uneven deformation occurred on the specimen, even though it was not as drastic as the first specimen. The deformation seemed to follow the unwanted extra material that was creeping out of the mould, but hanging model proved to be an effective way to create a better drying process.

SPECIMEN 3:

The specimen 3 was created the same cooking, mould, covering and drying procedure as the specimen 2, except used a different recipe of 1 g of glycerine, 50 ml of water and 23 g of gelatine. A different recipe was chosen to observe the influence of gelatine to water percentage on the specimen. The specimen with less water required less time to thicken. The specimen was also deformed similar to specimen 2, by following the extra material that creeped while the specimen was inside the mould.

SPECIMEN 4:

The specimen 4,5 and 6 were created on the same day. The specimen 4 is made out of 5 drops of glycerine, 50 ml of water and 23 g of gelatine. The recipe was changed to even less glycerine. This mixture was heated until the mixture was thickened and poured into the mould. The mould was covered with another sheet of vivak and some weight. An uneven amount of extra material was creeped out of the mould while drying inside the mould. This extra material was manually removed using a blade. The clips were attached on the specimen on both ends to create a balanced effect and some weight on the lower end of the specimen. Removal of the extra materials around the specimen significantly improved the behaviour of the deformation, even though it still occurred. The holes created by the pressure of the clips seemed to influence the deformation to a certain degree and suggested the need for a better way to hang the specimen to dry.

SPECIMEN 5:

The specimen 5 is made out of 1 drop of glycerine, 50 ml of water and 23 g of gelatine. This mixture was heated until the mixture was thickened and poured into the mould. The mould was covered with another sheet of vivak and some weight. An uneven amount of extra material was creeped out of the mould while drying inside the mould. This extra material was manually removed using a blade. The specimen was hanged to a thin wooden stick using a string going through the specimen on both corners. While this method helped preventing to create holes on both ends of the specimen, it did not fully prevent deformation. Additionally, it was not ideal to sew through material during the drying process considering its future uses.

SPECIMEN 6:

The specimen 6 is made out of 3 drops of glycerine, 50 ml of water and 23 g of gelatine. This mixture was heated until the mixture was thickened and poured into the mould. The mould was covered with another sheet of vivak and some weight. An uneven amount of extra material was creeped out of the mould while drying inside the mould. This extra material was manually removed using a blade. The specimen was hanged to a thin wooden stick using a string going through the specimen on both corners. Another wooden stick was hanged from the specimen to create some weight and therefore force to straighten the specimen more. This attempt was also not successful and led to unwanted deformation. A better method for drying process is still needed.

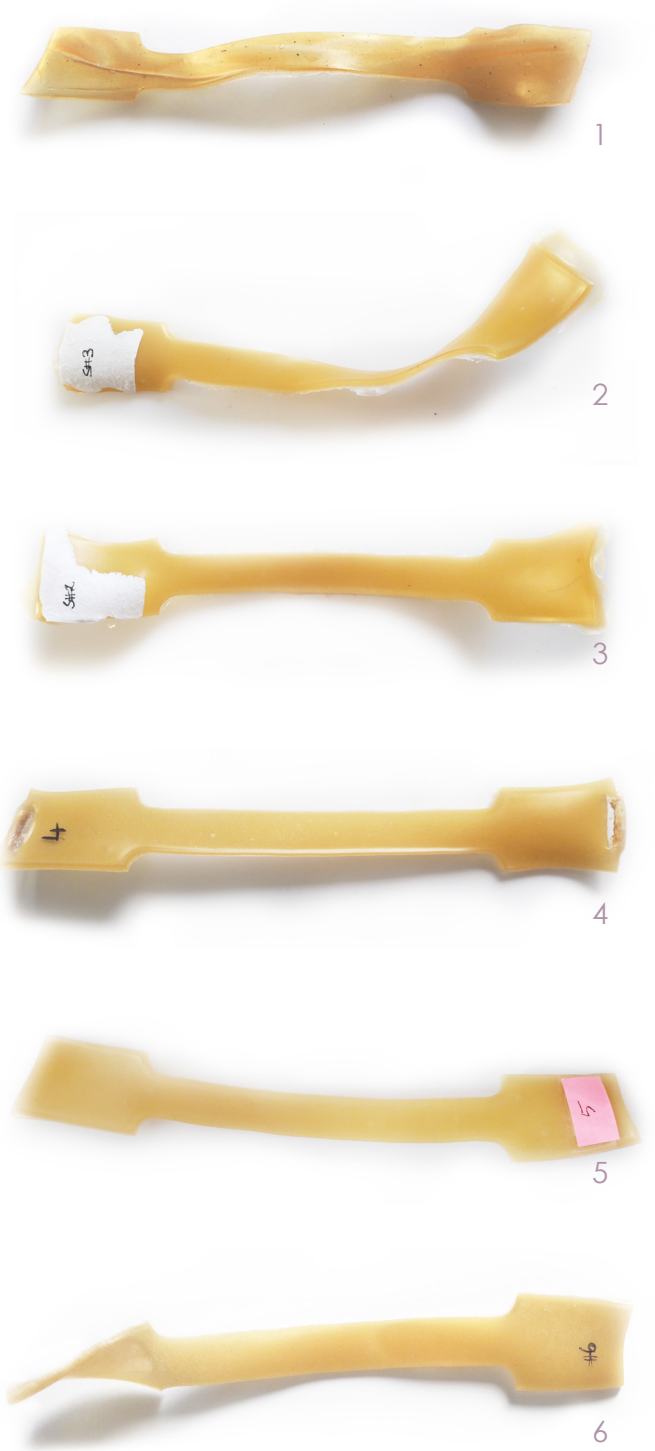


Figure 3.4: First six specimens that failed to meet the requirements

III. I. VI FRAME DESIGN

Following the first six specimen a better drying process needed to be developed. The research in section II. II. III suggests the use of frames. Initial attempt to avoid frames was to observe if the behaviour of the material could be controlled without a frame, as it could open up opportunities to experiment more 3D forms. While this could still be done, given the time frame of the project, this area was not investigated further in this research and a frame to improve the drying process was developed.

For a frame to be effective, it needs to hold the specimen straight while its curing. Because the sample loses water volume as it cures, it needs to be strong enough to endure the forces created during shrinkage. It is important to maintain the flatness and original dimensions as much as possible.

Any frame that will hold the specimen in place will have to deform the specimen where it connects to it. Therefore, the areas where the frame will be attached to the specimen should be chosen on the areas where this deformation would not influence the test.

As a result, a rectangular frame that will pinch the specimen on both ends and keep the at the desired distance while leaving the remaining parts of the specimen untouched. The frame was constructed out of timber to increase the friction between the material and the frame. This way the material won't slip out of the frame while curing. Examples of frames and the drying process can be seen in Figure 3.5.

Figure 3.5: Some of the specimens as they are drying and curing in the frames.

III.II RESULTS

	Base (g)			Additives (g)		Cooking	F_{max}	dL at F_{max}	F_{break}	dL at break	a_0	b_0	S_0
Samples	Glycerine	Water	Gelatin	Coffee	Egg Shells	mins	N	mm	N	mm	mm	mm	mm²
#7	1	50	23	4	-		1108,81	7,34	1108,81	7,34	4	10	40
#8	1	50	23	2	-	17	1646,34	7,73	1646,34	7,73	4	10	40
#9	1	40	23	-	-	12	575,50	4,48	115,09	17,44	4	10	40
#10	1	30	23	-	-	9	709,89	5,60	690,22	5,60	4	10	40
#11	1	50	23	-	4	18,5							
#12	1	50	23	-	6	19,5	324,80	6,10	318,64	6,11	4	10	40
#13	1	50	23	-	-	17	117,47	4,69	23,46	12,76	4	10	40
#14	1	50	23	-	-	17	1582,39	7,61			4	10	40
#15	1	50	23	-	-	17	1964,03	7,49	1876,27	7,50	4	10	40
#16	1	50	23	-	-	17	231,04	15,70	161,94	43,21	4	10	40
#17	1	50	23	-	-	17	1589,95	7,11	1589,95	7,11	4	10	40
#18	1	50	23	2	-	17	827,42	10,01	823,90	10,02	4	10	40
#19	1	50	23	2	-	17	506,95	7,13	101,23	10,31	4	10	40
#20	1	50	23	2	-	17	637,28	6,92	127,42	7,53	4	10	40
#21	1	50	23	2	-	17	598,58	8,30	305,44	10,27	4	10	40
#22	1	30	23	-	-	9	771,58	6,32	763,30	6,33	4	10	40
#23	1	30	23	-	-	9	1286,23	8,56	1284,38	8,56	4	10	40
#24	1	30	23	-	-	9	347,61	7,24	123,49	39,58	4	10	40
#26	1	50	23	4	-	16	124,26	36,49			4	10	40
#27	1	50	23	4	-	16	299,43	6,07	59,45	7,52	4	10	40
#28	1	50	23	4	-	16	106,76	2,44	21,25	31,80	4	10	40
#29	1	50	23	4	-	16	295,52	6,18	57,44	7,59	4	10	40

- Type 01: 1 g Glycerine, 50 g Water, 23 g Gelatin, 4 g Coffee Grounds
- Type 02: 1 g Glycerine, 50 g Water, 23 g Gelatin, 2 g Coffee Grounds
- Type 03: 1 g Glycerine, 30 g Water, 23 g Gelatin
- Type 04: 1 g Glycerine, 50 g Water, 23 g Gelatin

Table 3: Content of the tested specimens and the overall results of the tensile tests

III.III DISCUSSION

Table 3 shows all the tested specimens' content and the results for the tests. The contents that were identical are colour coded.

The goal of these tests is to understand this experimental material's (bioplastic's) tensile strength and overall behaviour in order to design, optimize and produce building components that could work well with earth in places where earth's tensile strength is not sufficient.

The output of this test includes following information: standard force-strain graph, maximum force, length at the maximum force, force when the specimen breaks and the length of the specimen when it breaks.

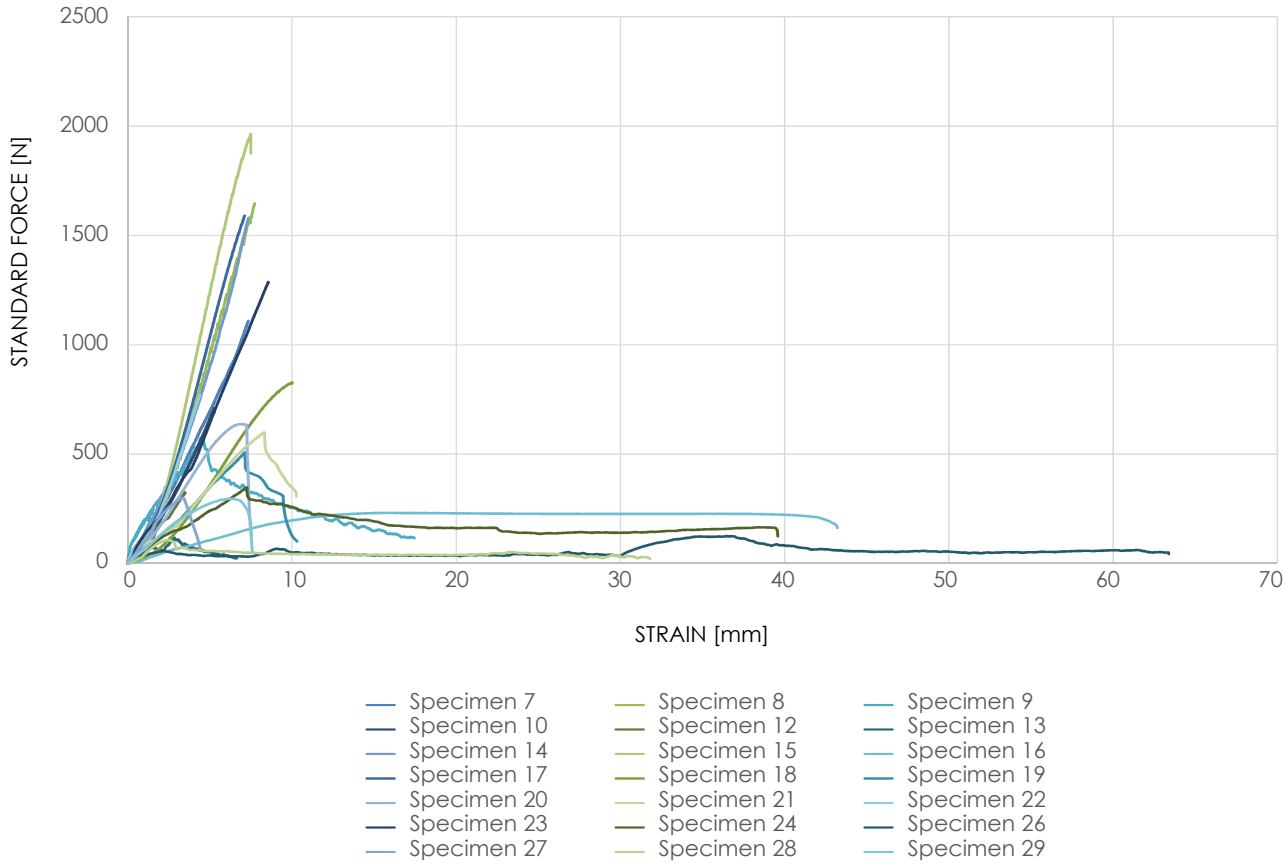
Stress can be defined as the units of force per unit area and can be represented as σ (sigma). Assuming the force applied on a cross-section is distributed evenly, the stress on a given cross section can be calculated following Equation 1 [9], where P stands for Force[N], A for cross-section area [mm2] and σ for stress [N/mm2] [9].

$$\sigma = \frac{P}{A} \qquad (1)$$

Young's modulus is equal to "the slope of the initial linear-elastic part of the stress-strain curve in tension [10]. We can obtain this curve using the standard force-strain graph and equation 1, by dividing the standard force values to cross-section area -in this case 40 mm^2.

Tensile strength is the tensile "stress required to break the material. For most plastics and ceramics, the peak in tensile stress occurs at break" [10]. Since we are given the value for F break, this divided by the cross-section area will give us the tensile strength of the bioplastics.

The yield strength (elastic limit) is "the stress at the first peak of the stress strain curve"[10]. This value also can be obtained from the standard force-strain graph by using the equation1.



Graph 1: The overlay of all the standard force-strain graphs of the specimens.

Figure 3.6: The specimens [07, 26, 27, 28, 29] before and after

III. III. I TYPE 01

Five different specimens with the same recipe of 1 g of glycerine, 50 g of water, 23 g of gelatin and 4 g of coffee was created. Even though the recipes were identical, the performance of the specimens were fairly different. The difference can be seen on Table 4 and Graph 2.

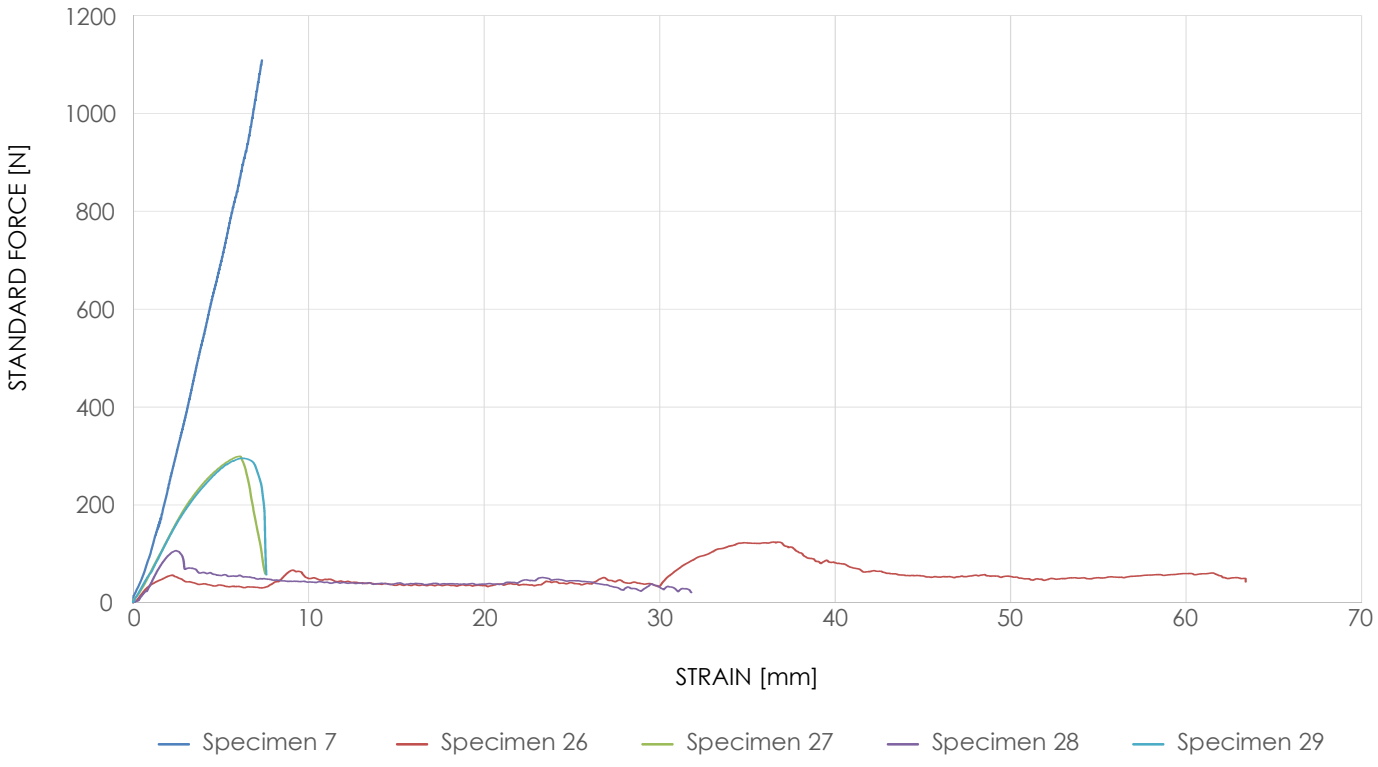
While Specimen 7 performed extremely well in respect to its young's modulues tensile and yield strenght, other specimens did not perform as successfully.

The difference can also be observed in the way the specimens broke. While specimen 7 broke without any necking, specimen 26, 27, 28 and 29 yielded for longer. This specific difference could be the result of the curing period after it's been removed from the frame. Specimen 7 had significantly more time to cure than specimens 26, 27, 28 and 29.

It could be speculated that higher curing time increases the young's modulus, tensile and yield strength. Given the data, we can conclude that specimen 07 is an outlier and calculate the **tensile strength of Type 01 as 1.64 ± 1.07 MPa. Yield strength is 4,74 ± 3,16 MPa. Young's modulus is 3,55 ± 0,59 MPa.**

Specimens	Fmax	Fbreak	Tensile Strength	Yield Strength	Young's Modulus	
	[N]	[N]	[MPa]	[MPa]	[N/mm2]	[GPa]
07	1108,812	1108,812	27,720	27,720	34,598	0,035
26	124,265		3,107	1,400	2,764	0,003
27	299,432	59,448	1,486	7,486	3,786	0,004
28	106,763	21,246	0,531	2,669	3,473	0,003
29	295,518	57,442	1,436	7,388	4,162	0,004

Table 4: The mechanical properties of Type 01 specimens.



Graph 2: The overlay of all Type 01 Specimens with 1 g of glycerine, 50 g of water, 23 g of gelatine and 4 g of coffee

Figure 3.7: The specimens [08, 18, 19, 20, 21] before and after

III. III. II TYPE 02

Five different specimens with the same recipe of 1 g of glycerine, 50 g of water, 23 g of gelatin and 4 g of coffee was created. The results can be seen on Table 5 and Graph 3.

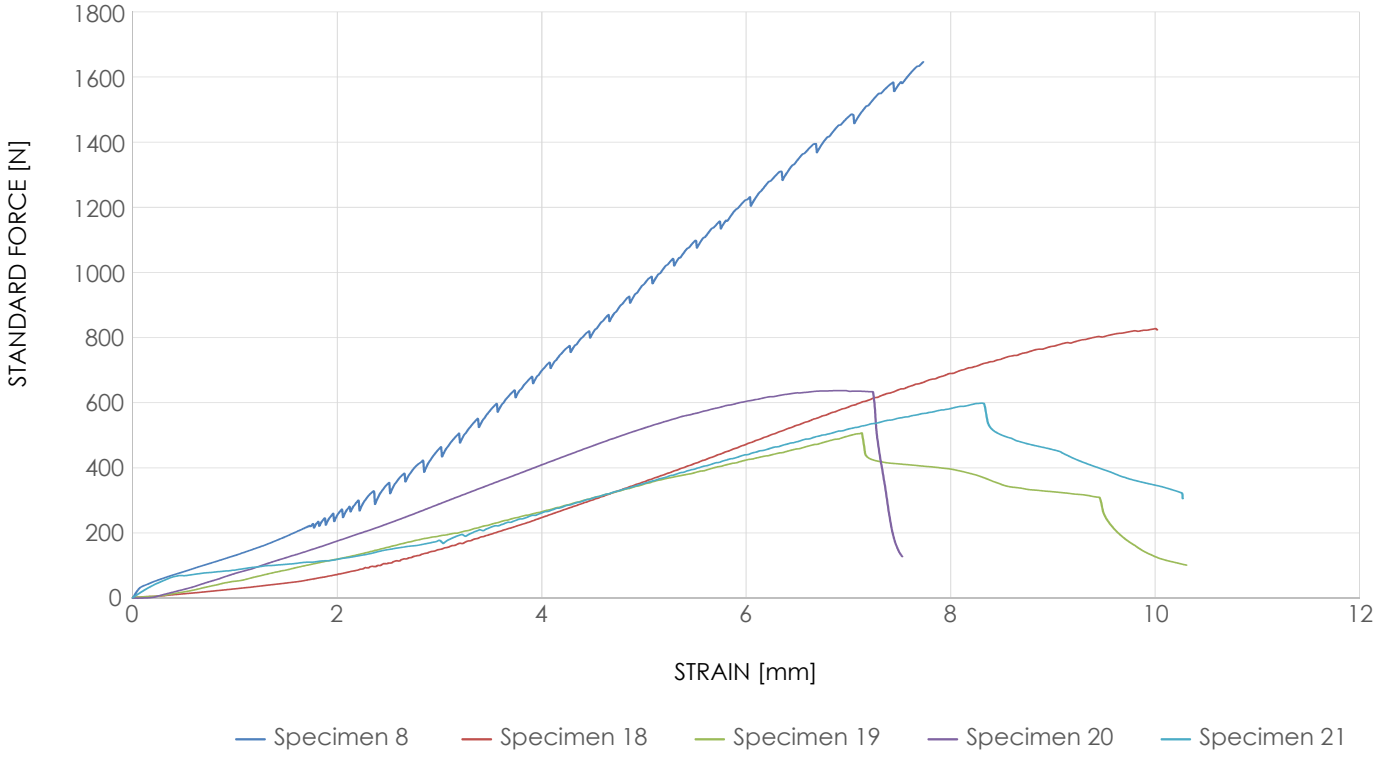
Similar to Type 01 specimens, Specimen 08 in Type 02 performed significantly better than the rest of the specimens. This supports the previous arguement about the curing process and its influence on the material's mechanical properties.

However, it could also be argued that the specimens 18, 19, 20 and 21 behaved fairly similar and the specimen 08 was an exception. Based on the results we can conclude that **Type 02 has the tensile strength of 18.78 ± 14.54 MPa, yield strength of 23.04 ± 14.36 MPa and young's modulus of 14.44 ± 10.50 MPa.**

If we compare specimen 07 to 08, we see the 08 or calculated mechanical properties, it can be argued that using higher percentage of coffee grounds weakens the material's young's modulus, tensile and yield strength.

Specimens	Fmax	Fbreak	Tensile Strength	Yield Strength	Young's Modulus	
	[N]	[N]	[MPa]	[MPa]	[N/mm2]	[GPa]
08	1646,342	1646,342	41,159	41,159	34,099	0,034
18	827,422	823,896	20,597	20,686	2,165	0,002
19	506,951	101,226	2,531	12,674	4,524	0,005
20	637,281	127,423	3,186	15,932	6,138	0,006
21	598,583	305,444	7,636	1,720	10,823	0,011

Table 5: The mechanical properties of Type 02 specimens.



Graph 3: The overlay of all Type 02 Specimens with 1 g of glycerine, 50 g of water, 23 g of gelatine and 2 g of coffee

Figure 3.8: The specimens [10, 22, 23, 24] before and after

III. III. III TYPE 03

Four different specimens with the same recipe of 1 g of glycerine, 30 g of water and 23 g of gelatin was created. The results can be seen on Table 6 and Graph 4.

Looking at the graph, specimen 24 seems to have a different behaviour than the rest. This difference can be the result of how the specimen broke. Unlike the others, it did not broke in the middle part. This might be the result of an error in its fabrication.

The rest of the specimens seems to perform close to each other. Unlike the previous types, curing process may not have effected this recipe. Coffee grounds may be extending the curing process.

Type 03 has the tensile strength of 17.88 ± 15.73 MPa, yield strength of 19.47 ± 14.79 MPa and young's modulus of 7.39 ± 4.63 MPa.

Specimens	Fmax	Fbreak	Tensile Strength	Yield Strength	Young's Modulus	
	[N]	[N]	[MPa]	[MPa]	[N/mm2]	[GPa]
10	709,886	690,216	17,255	17,747	9,516	0,010
22	771,575	763,298	19,082	19,289	8,380	0,008
23	1286,234	1284,382	32,110	32,156	7,221	0,007
24	347,605	123,486	3,087	8,690	4,439	0,004

Table 6: The mechanical properties of Type 03 specimens.

Graph 4: The overlay of all Type 03 Specimens with 1 g of glycerine, 30 g of water and 23 g of gelatine.

Figure 3.9: The specimens [13, 14, 15, 16, 17] before and after

III. III. IV TYPE 04

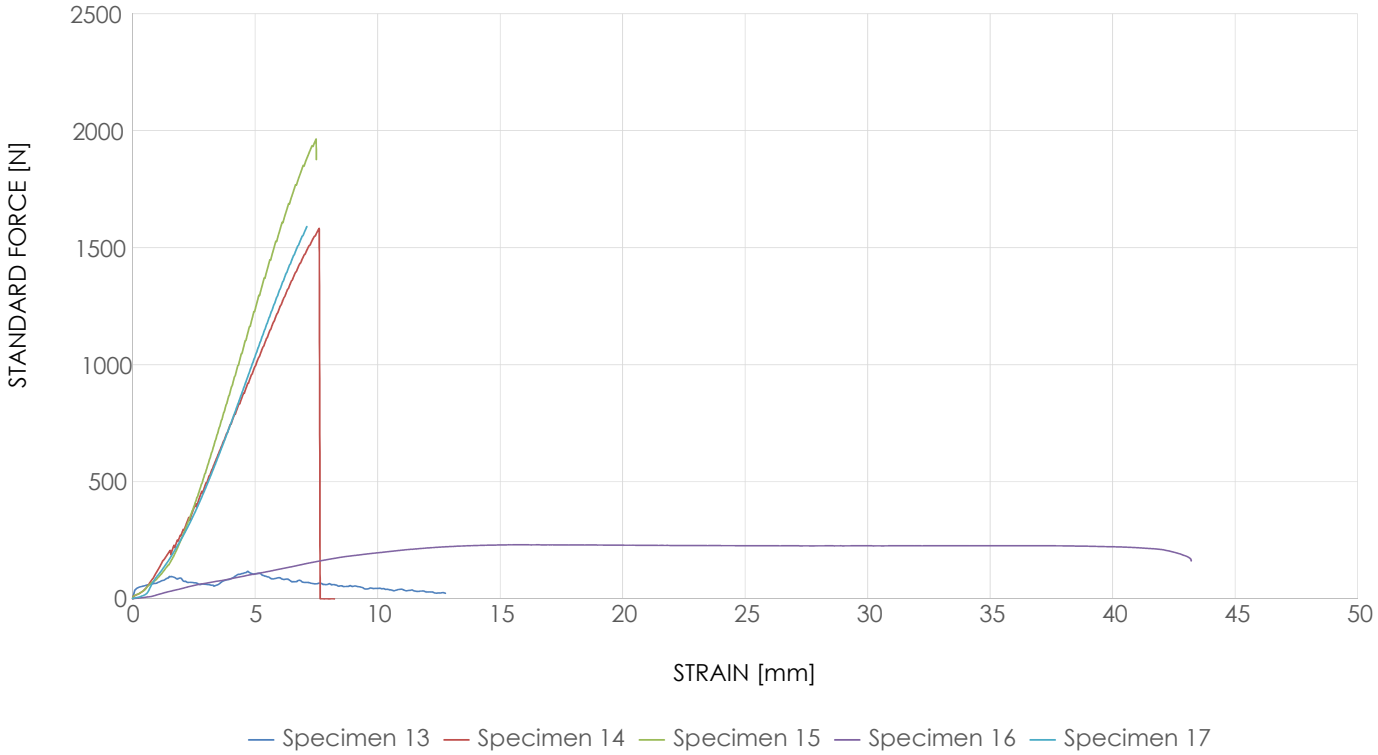
Five different specimens with the same recipe of 1 g of glycerine, 50 g of water and 23 g of gelatine was created. The results can be seen on Table 7 and Graph 5.

Specimen 13 and 16 perfomed significantly worse than the rest. However specimen 13 was not broken during the test. Therefore this might be the error during the testing.

Type 04 has the tensile strength of 32.71 ± 19.32 MPa, yield strength of 33.59 ± 18.48 MPa and young's modulus of 11.41 ± 7.82 MPa.

Specimens	Fmax	Fbreak	Tensile Strength	Yield Strength	Young's Modulus	
	[N]	[N]	[MPa]	[MPa]	[N/mm2]	[GPa]
13	117,468	23,461	0,587	2,378	24,985	0,025
14	1582,394	1582,394	39,560	39,560	4,632	0,005
15	1964,027	1876,273	46,907	46,907	5,016	0,005
16	231,042	161,939	4,048	5,776	1,859	0,002
17	1589,953	1589,953	39,749	39,749	9,166	0,009

Table 7: The mechanical properties of Type 04 specimens.



Graph 5: The overlay of all Type 04 specimens with 1 g of glycerine, 50 g of water and 23 g of gelatine.

Figure 3.10: Tensile strength test at 3mE TU Delft

III.IV CONCLUSION

If we look at the overall result, we can conclude that **Type 04** is the best option for the proposed panels. It has the highest tensile and yield strength and has the second highest youngs modulus.

However, as discussed in sections III. III. I TYPE 01 and III.III.II TYPE 02, the specimens with coffee grounds may need a longer curing period and may have a lower performance due to the curing process. Considering the specimens 07 and 08, trying the recipe of Type 02 for the panels with the appropriate amount of curing time may improve the performance of the panel.

In conclusion, while Type 04 seems like the best option among the tested recipes, it is valid to test panels with Type 02 recipes.

	Tensile Strength		Yield Strength		Young's Modulus		Glycerine	Water	Gelatine	Coffee
	[MPa]	SD [±]	[MPa]	SD [±]	[MPa]	SD [±]	(g)	(g)	(g)	(g)
Type 01	1,64	1,07	4,74	3,16	3,55	0,59	1	50	23	4
Type 02	18,78	14,54	23,04	14,36	14,44	10,50	1	50	23	2
Type 03	17,88	15,73	23,04	14,79	7,39	4,63	1	30	23	-
Type 04	32,71	19,32	33,59	18,48	11,41	7,82	1	50	23	-

Table 8: The calculated mechanical properties of all types of recipes.

Specimens	Fmax	Fbreak	Tensile Strength	Yield Strength	Young's Modulus	
	[N]	[N]	[MPa]	[MPa]	[N/mm2]	[GPa]
07	1108,812	1108,812	27,720	27,720	34,598	0,035
08	1646,342	1646,342	41,159	41,159	34,099	0,034
09	575,498	115,092	2,877	14,387	27,418	0,027
10	709,886	690,216	17,255	17,747	9,516	0,010
12	324,798	318,641	7,966	2,031	5,077	0,005
13	117,468	23,461	0,587	2,378	24,985	0,025
14	1582,394	1582,394	39,560	39,560	4,632	0,005
15	1964,027	1876,273	46,907	46,907	5,016	0,005
16	231,042	161,939	4,048	5,776	1,859	0,002
17	1589,953	1589,953	39,749	39,749	9,166	0,009
18	827,422	823,896	20,597	20,686	2,165	0,002
19	506,951	101,226	2,531	12,674	4,524	0,005
20	637,281	127,423	3,186	15,932	6,138	0,006
21	598,583	305,444	7,636	1,720	10,823	0,011
22	771,575	763,298	19,082	19,289	8,380	0,008
23	1286,234	1284,382	32,110	32,156	7,221	0,007
24	347,605	123,486	3,087	8,690	4,439	0,004
26	124,265		3,107	1,400	2,764	0,003
27	299,432	59,448	1,486	7,486	3,786	0,004
28	106,763	21,246	0,531	2,669	3,473	0,003
29	295,518	57,442	1,436	7,388	4,162	0,004

Table 9: The mechanical properties of all specimens

IV. DESIGN PROCESS

IV. I. CONCEPTUAL DESIGN

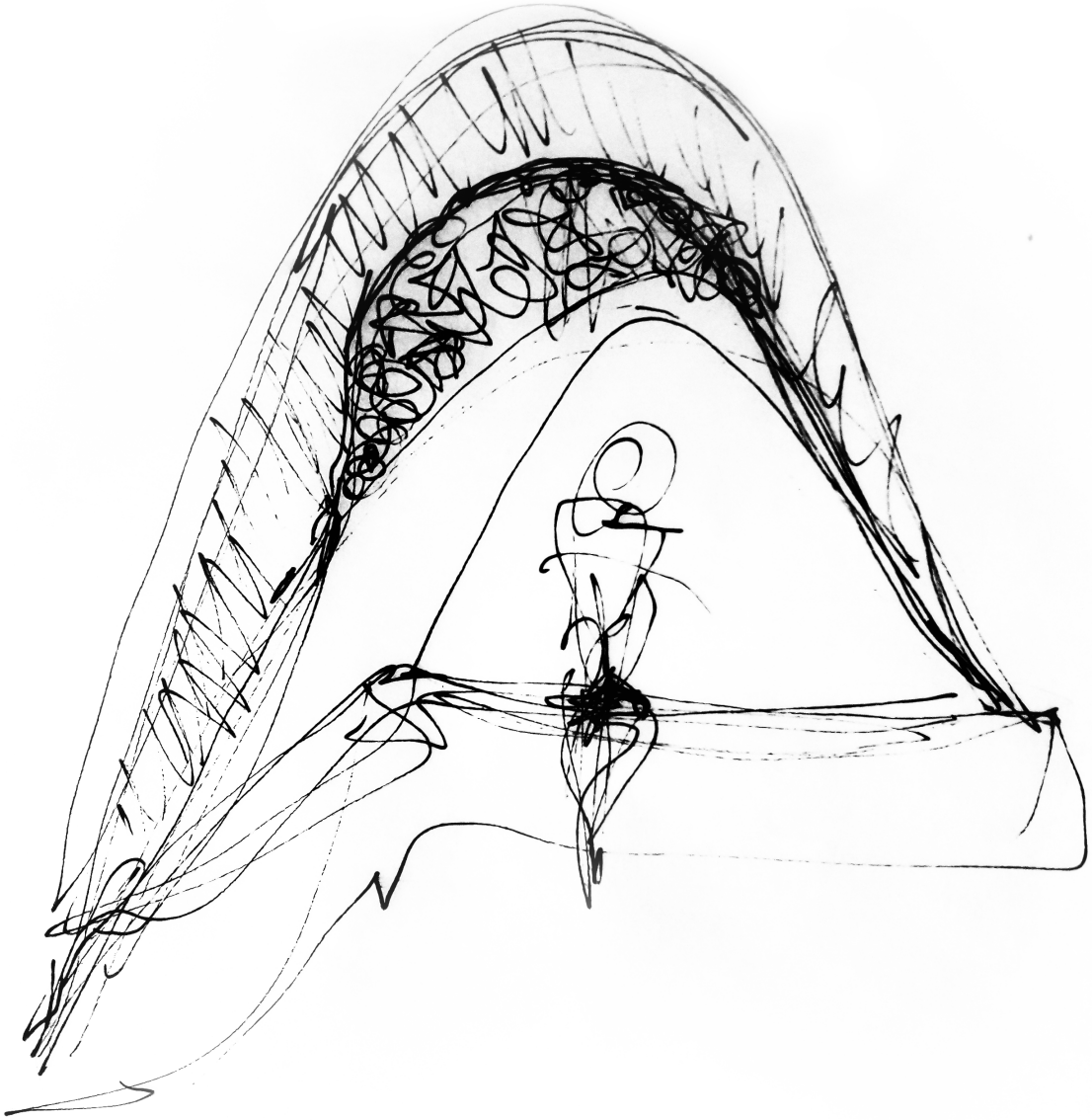
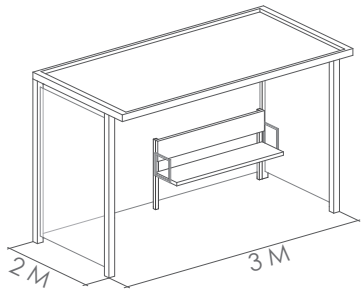


Figure 4.1: Handsketch

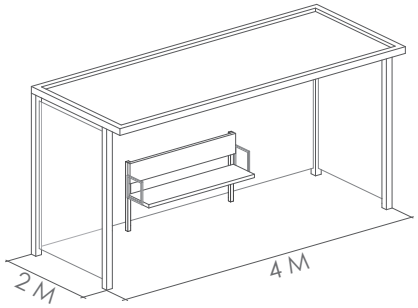
IV. I. I. EXISTING BUS STATIONS

SMALL BUS STATION



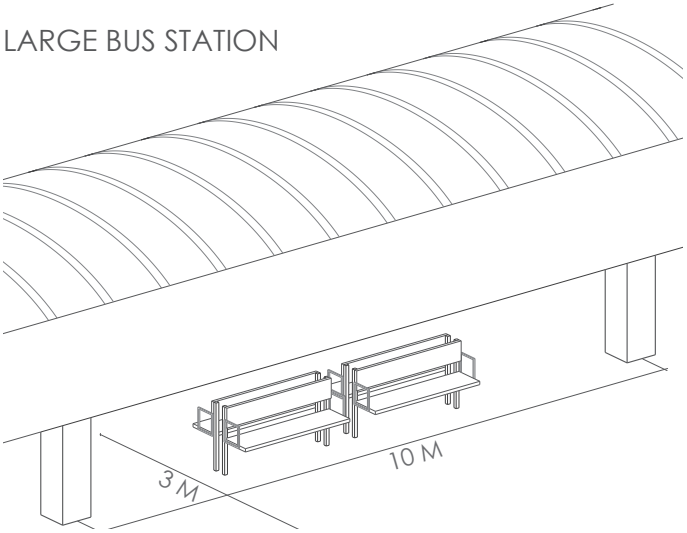
A small size bus stop is 2m x 3 m with glass casing and a seat for two people.

MEDIUM BUS STATION



A medium size bus stop is 2m x 4 m with glass casing and a seat for two people.

LARGE BUS STATION



A large scale bus station is usually more site specific. This particular one is drawn based on den Haag HS Station. The station expends over 60 m consists of 10m modules of station areas. As the station serves both sides, the depth of the station is increased to 3 m.

Figure 4.2: Existing bus station analysis

IV. I. II. INITIAL FORM

The dimensions, need for seats and canopy size was defined based on the standard dimensions found in section IV.I.I. Existing Bus Stations,

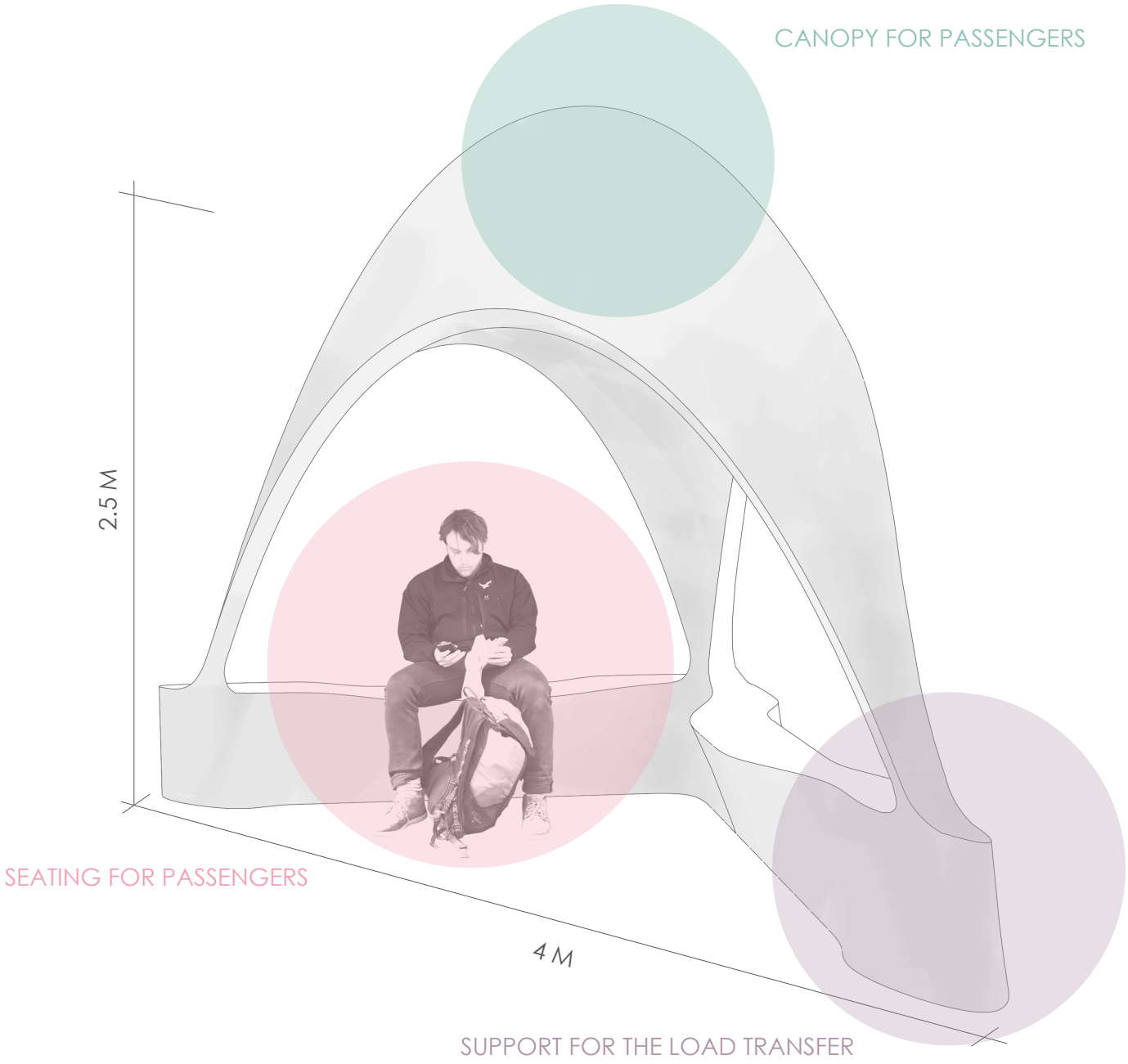


Figure 4.3: Proposed conceptual design

IV. I. III. COMPUTATIONAL WORK FLOW

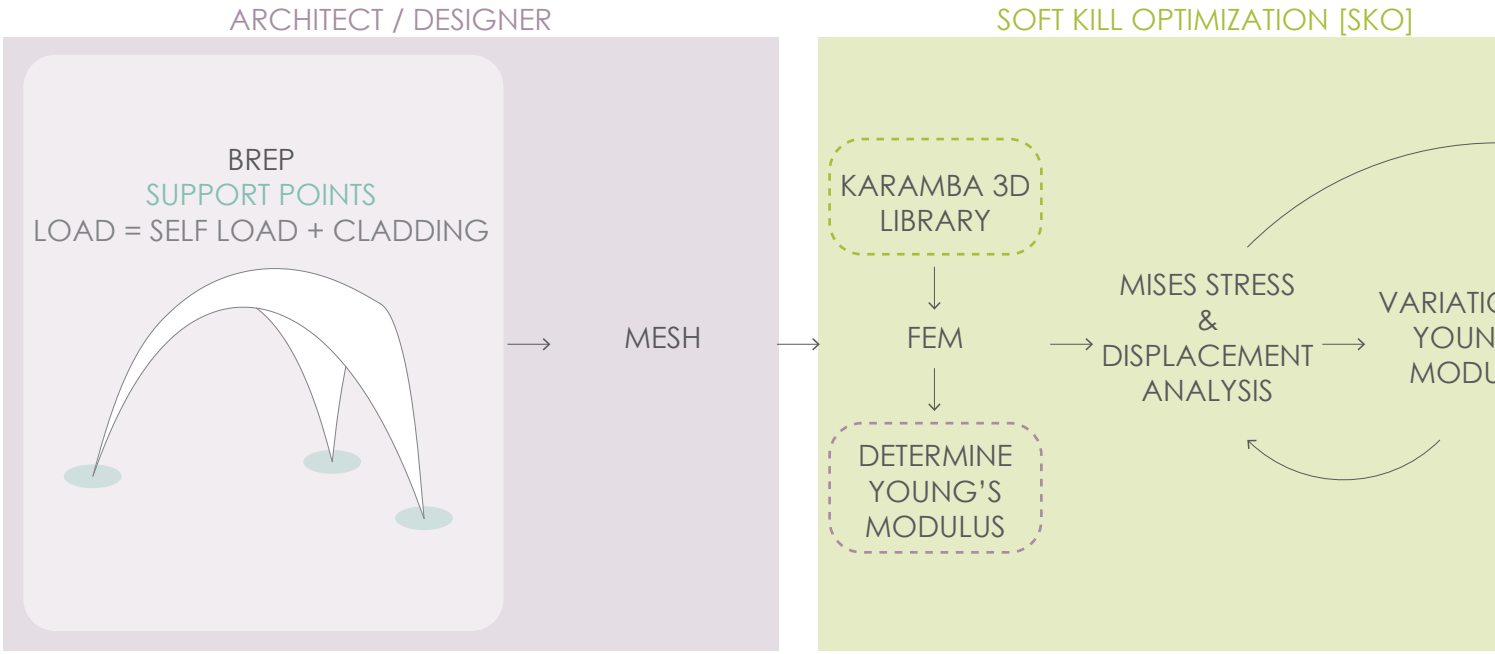
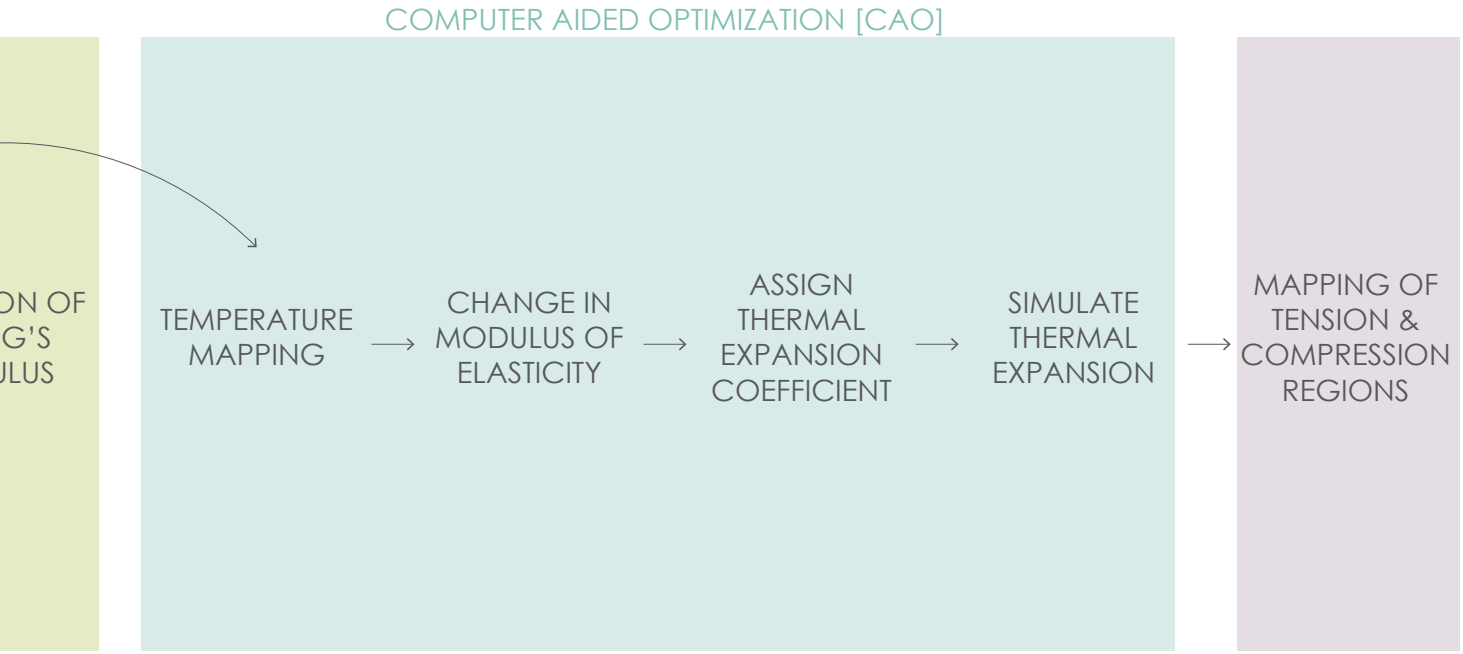


Figure 4.4: Conceptual computational workflow


COMPUTER AIDED OPTIMIZATION [CAO]
ON OF
G'S
ULUS
TEMPERATURE MAPPING
CHANGE IN MODULUS OF ELASTICITY
ASSIGN THERMAL EXPANSION COEFFICIENT
SIMULATE THERMAL EXPANSION
MAPPING OF TENSION & COMPRESSION REGIONS

IV. II. UNIT / JOINT DESIGN

The most important finding of section III. MATERIAL STUDY was that the unit design should absolutely consider the fabrication process. The fabrication of the samples showed that in order to control the behaviour of the elements, it needs to be streched while curing to control the deformation during the process. Therefore the unit should be designed to take this into consideration.

An alternative could have been that the folding behaviour has been considered and the units are designed accordingly. In some ways, that was also considered in the current unit design. However, leaving the control of the unit too much into the material would have resulted in units that are extremely difficult or maybe impossible to engineer for its purpose.

Initial goal was to 3D print these units, this idea was later dropped. The specimens were created without 3D printing, and changing the fabrication process may have effected the materials mechanical properties. Furthermore, it would have added a brand new line of experiments to understand the material's behaviour under a different fabrication process. In order to eliminate further experimentation with the fabrication process, using moulds to fabricate panels and then streching them like it's been done with the dog-bone specimens simplified the fabrication process.

While considering fabrication process for the unit design, it was also important to consider how these units will come together in an architectural setting. It needs to consider how they will connect to each other and how they will connect to the earth substructure, which is essentially the main structure that carries the whole.

Even though during the design process these considerations were happening spontenously, this chapter will start with the unit design and explore more into the joints between the units and their connection to the mud structure.

IV. II. I. BIOPLASTIC UNIT GEOMETRY

The geometry was basically derived from the amount of points or edges the unit will be streched from, and if they can come together when they were created. When that's the case units streched from two, three, four and six corners or edges seemed to have the most potential. So I started with a number of iterations as can be seen in Figure 4.5.

Among these iterations, the second from the top, or the triangular one was the one I chose to develop further, because as the structure curves, the panels will need to rotate along and this rotation could be most attainable by triangular panels.

The units with two pieces were too small and would require too many joints for the structure. square panels both architecturally and structurally was not appealing. Even though hexagonal panels had a more appealing geometry, as stiff panels both the square and hexagonal panels would have had a hard time accomodating the curvature of the main mud structure.

These diagrams also helped questioning the units of adobe bricks. Less standard and more suitable bricks could be developed as they are also developed using moulds.

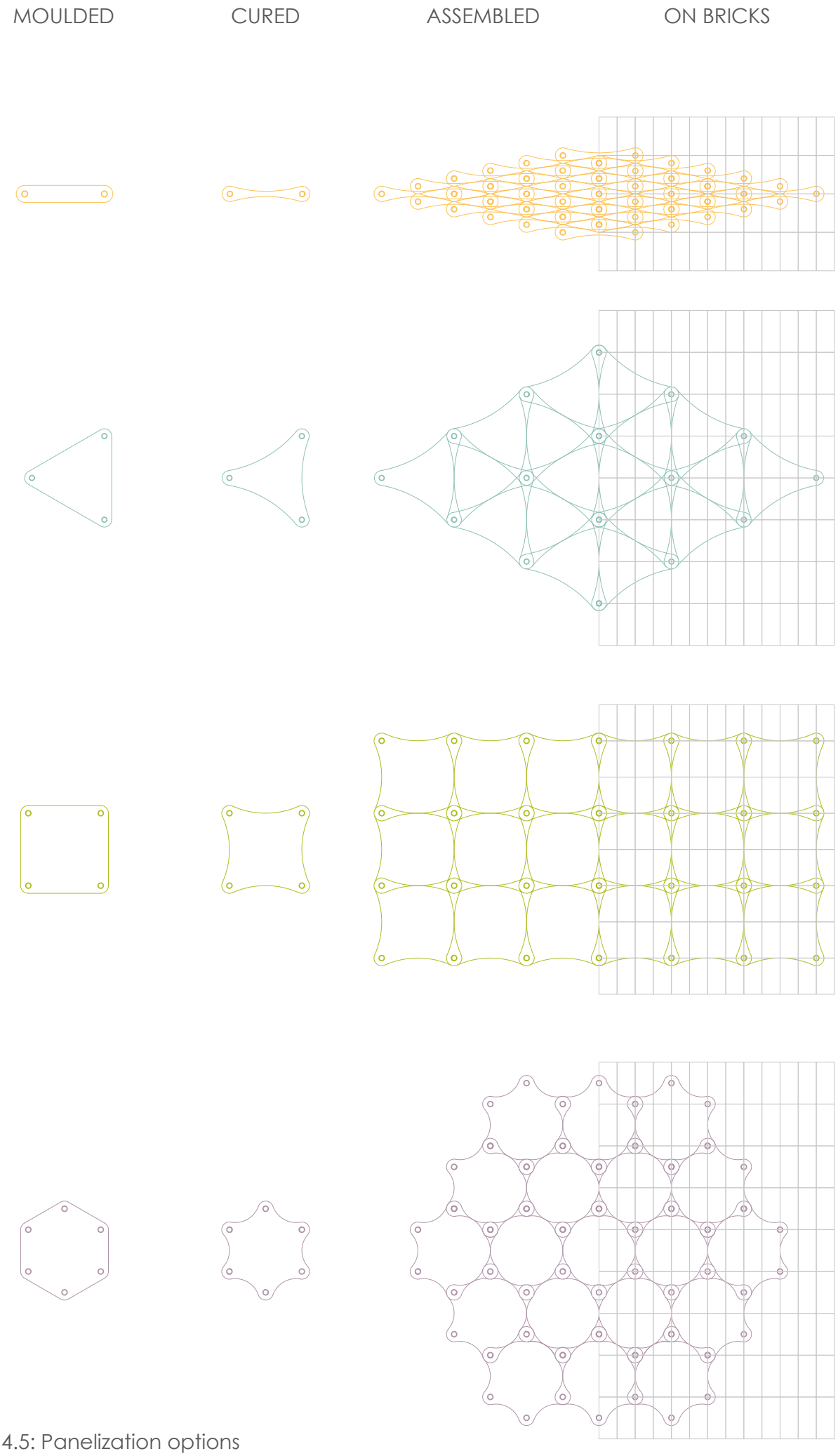


Figure 4.5: Panelization options

IV. II. II. UNIT FABRICATION PROCESS

MOULD

FRAME

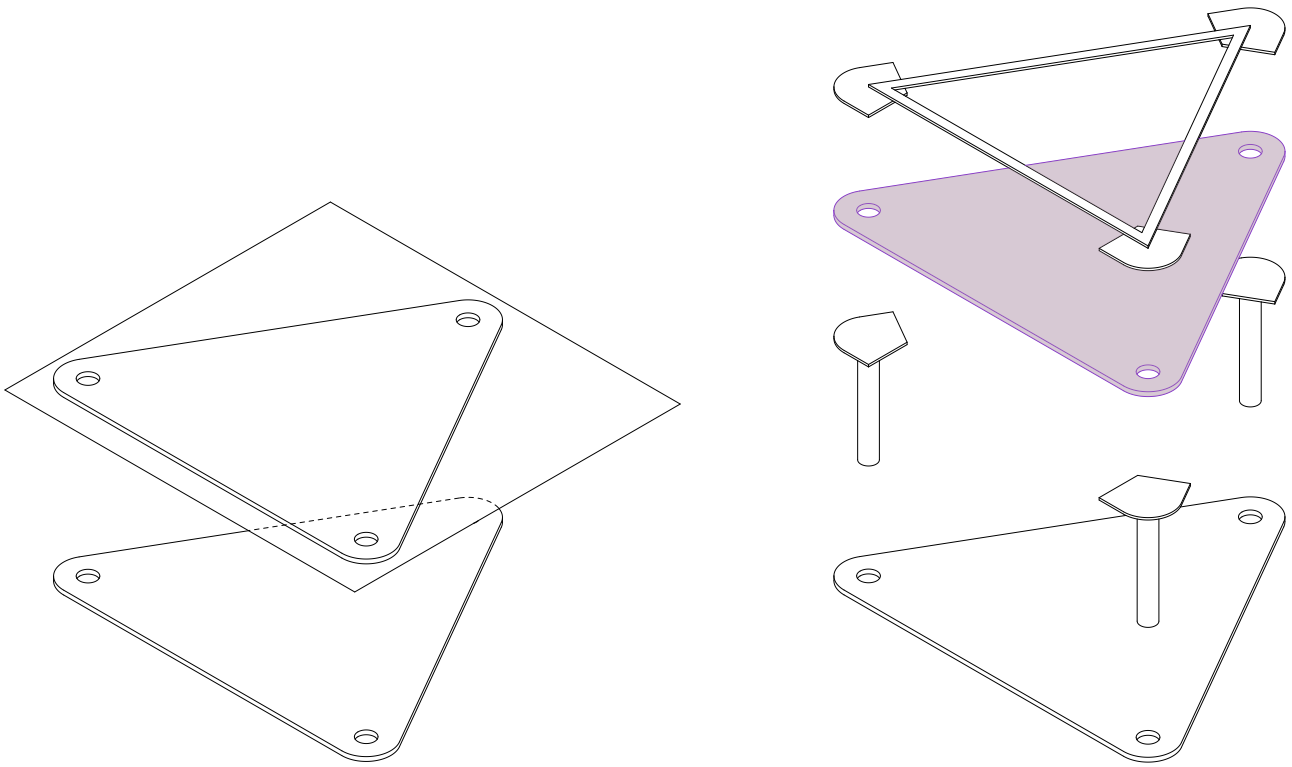
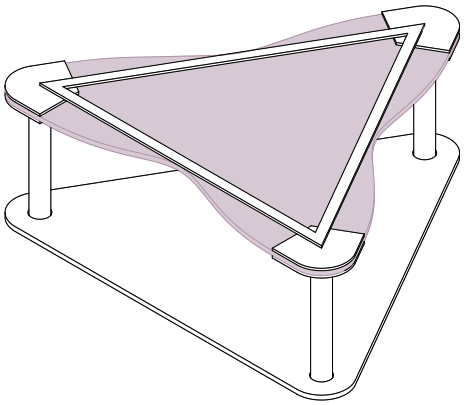
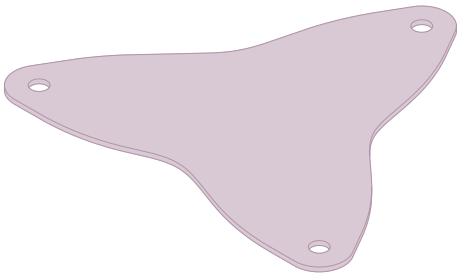


Figure 4.6: Panel fabrication process

DEFORMATION



PANEL



IV. II. III. UNIT SAMPLES

Figure 4.7: Fabricated panels

IV. II. IV. ASSEMBLY

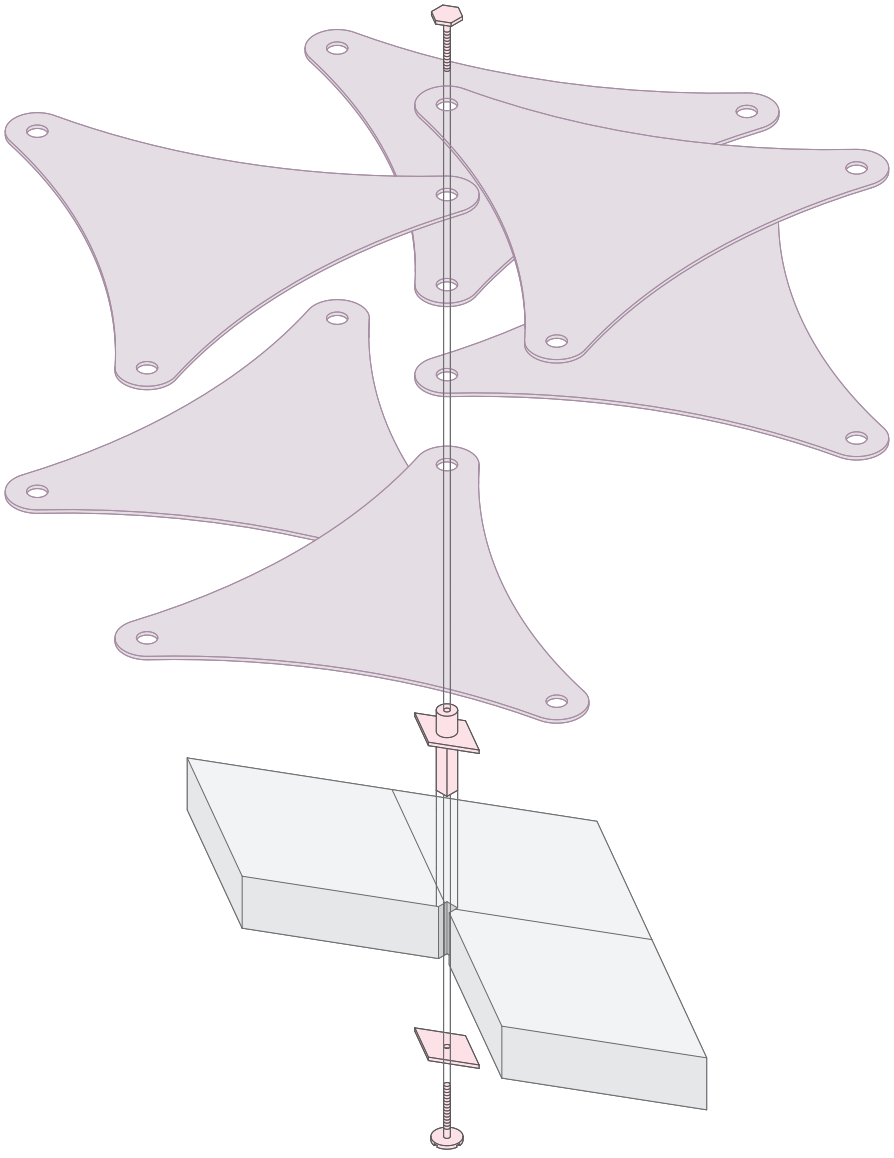


Figure 4.8: Panel assembly sequence

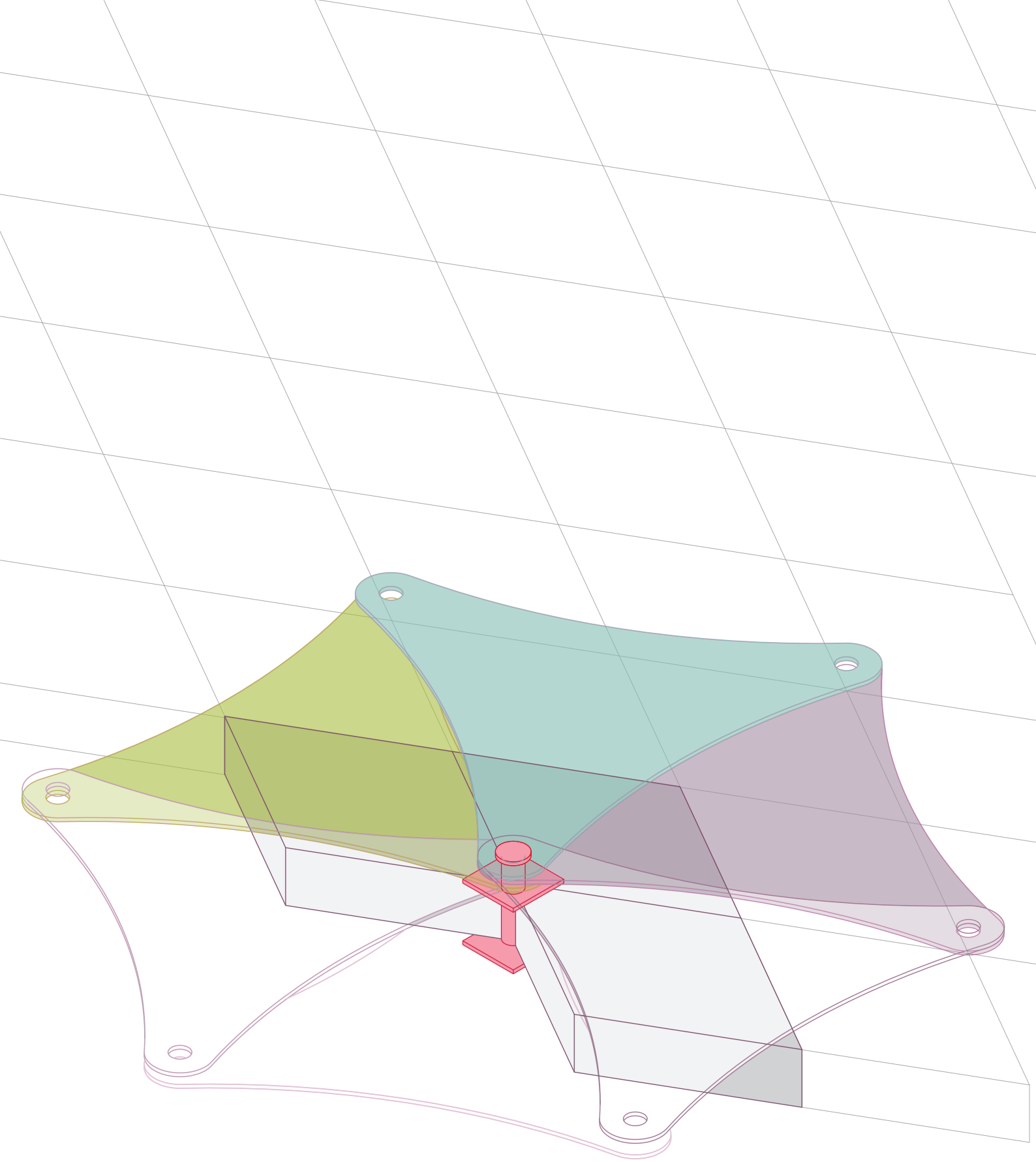


Figure 4.9: Panel assembly

IV. II. IV. TESTING MATERIALS

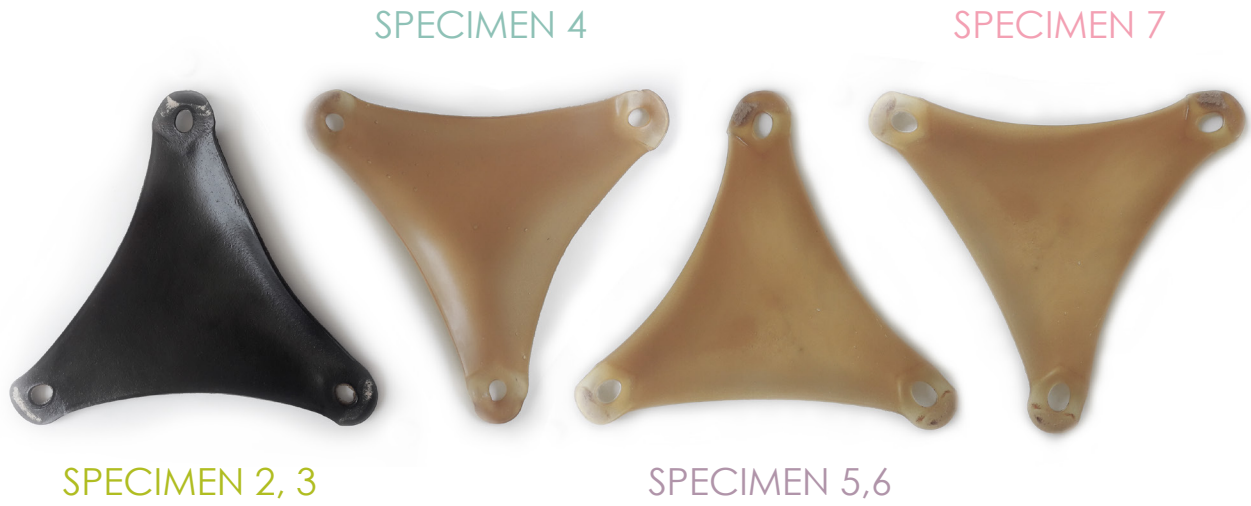


Figure 4.10: Panels to be tested

IV. II. IV. I INTRODUCTION & OBSERVATIONS

On May 24th 2019, in we ran tensile strength tests at 3mE TU Delft. For this test, there were four panels produced, following two different recipes. To clarify, specimen number do not represent individual panels but rather represent the test results obtained during the tesint of the panels. Therefore, specimen 2 and 3 belongs to panel 01 (green), specimen 4 belongs to panel 02 (blue), specimen 5 and 6 belongs to panel 03 (purple) and specimen 7 belongs to panel 04 (pink). To make this distinction clearer, color coding was used on this report. Please see Figure 4.10 and Table 10 for further clarification.

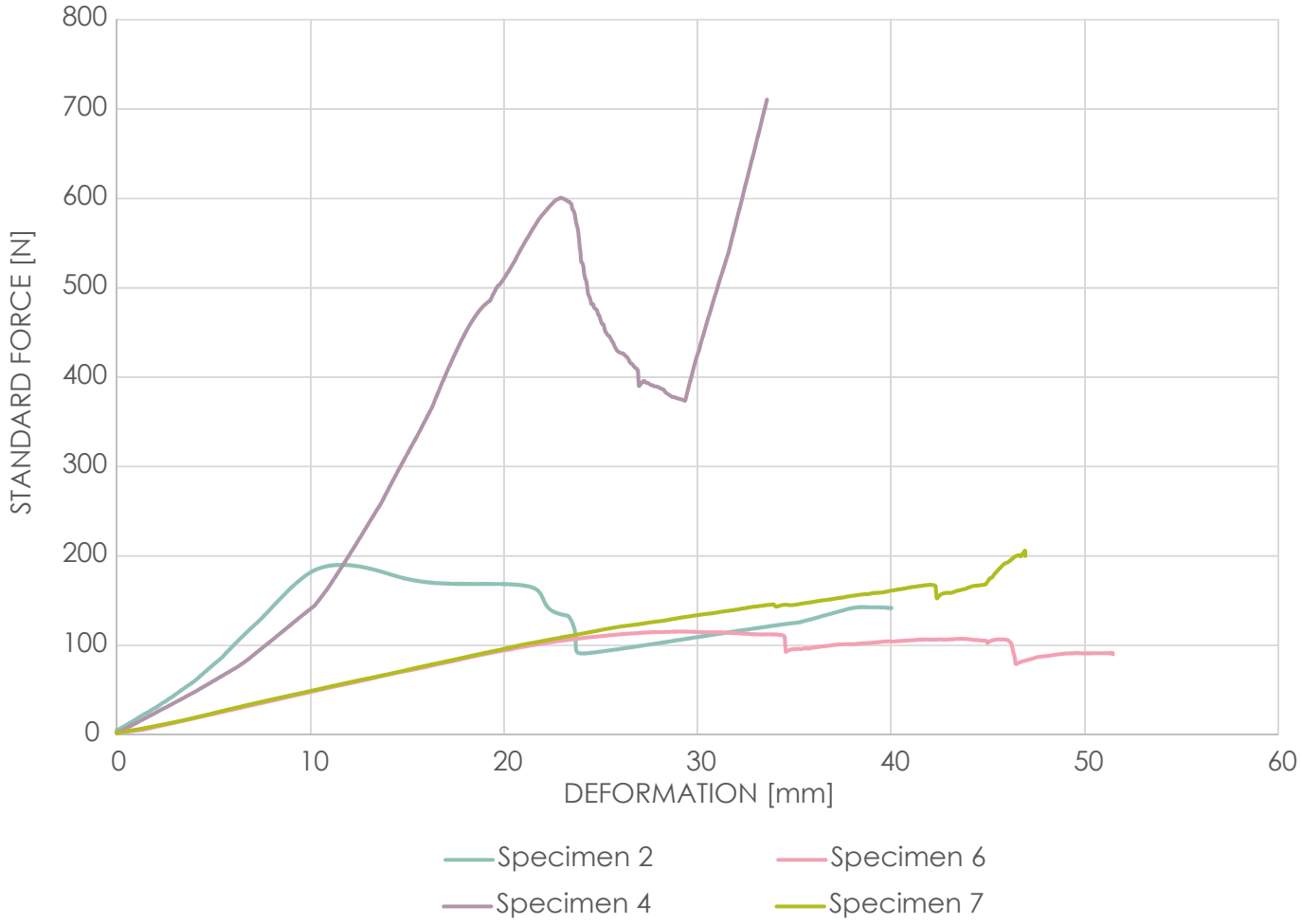
The main recipe that was followed was 23 unit gelatine, 1 unit glycerine and 50 unit water. Second recipe added 2 units of coffee grounds. The addition of the sample with coffee grounds was to clarify the uncertainty about the influence of curing process on the material behaviour.

For the first recipe -the one without coffee grounds- we tried keeping the same ratio but in different amounts of material to observe if the production process is scalable.

It is also important to note that due to the

	F_{max} N	dL at F_{max} mm	F_{Break} N	dL at break mm	a_0 mm	b_0 mm	S_0 mm²	Gelatine g	Glycerine g	Water g	Coffee g
Specimen 2	190,11	11,66			100,00	100,00	10000,00	207,00	9,00	450,00	18,00
Specimen 3	171,65	39,89	150,61	54,92	100,00	100,00	10000,00	207,00	9,00	450,00	-
Specimen 4	710,53	33,58	710,53	33,58	100,00	100,00	10000,00	276,00	12,00	600,00	-
Specimen 5	103,91	26,04			100,00	100,00	10000,00	207,00	9,00	450,00	-
Specimen 6	115,47	29,20			100,00	100,00	10000,00	207,00	9,00	450,00	-
Specimen 7	205,96	46,91			100,00	100,00	10000,00	207,00	9,00	450,00	-

Table 10: Panel test result



Graph 6: Panels test result

non-standardized fabrication method, each panel had a slightly different form, which influenced the structural performance of the form. While panel 02 had a visibly more symmetrical form, panel 01, 03 and 04 had a less symmetrical shape.

As a result following observations and conclusions were drawn.

IV. II. IV. I INTERPRETATION OF THE RESULTS

When the specimens are compared, based on the graphs drawn for four different types of panels, panel 04 was able to withstand the highest stress levels. This may be the result of the geometry more than the difference in the amount of materials used in the specimen.

Panel 03 and 04 having the exact same recipe, have fairly similar behaviour in comparison to the others. This shows the consistent behaviour of the material. However, the question of form and scalability unanswered.

Both panel 01 and 02 broke during the testing. Which was not the desired behaviour for a material that's been tested for structural purposes. The ideal result could have been deforming and

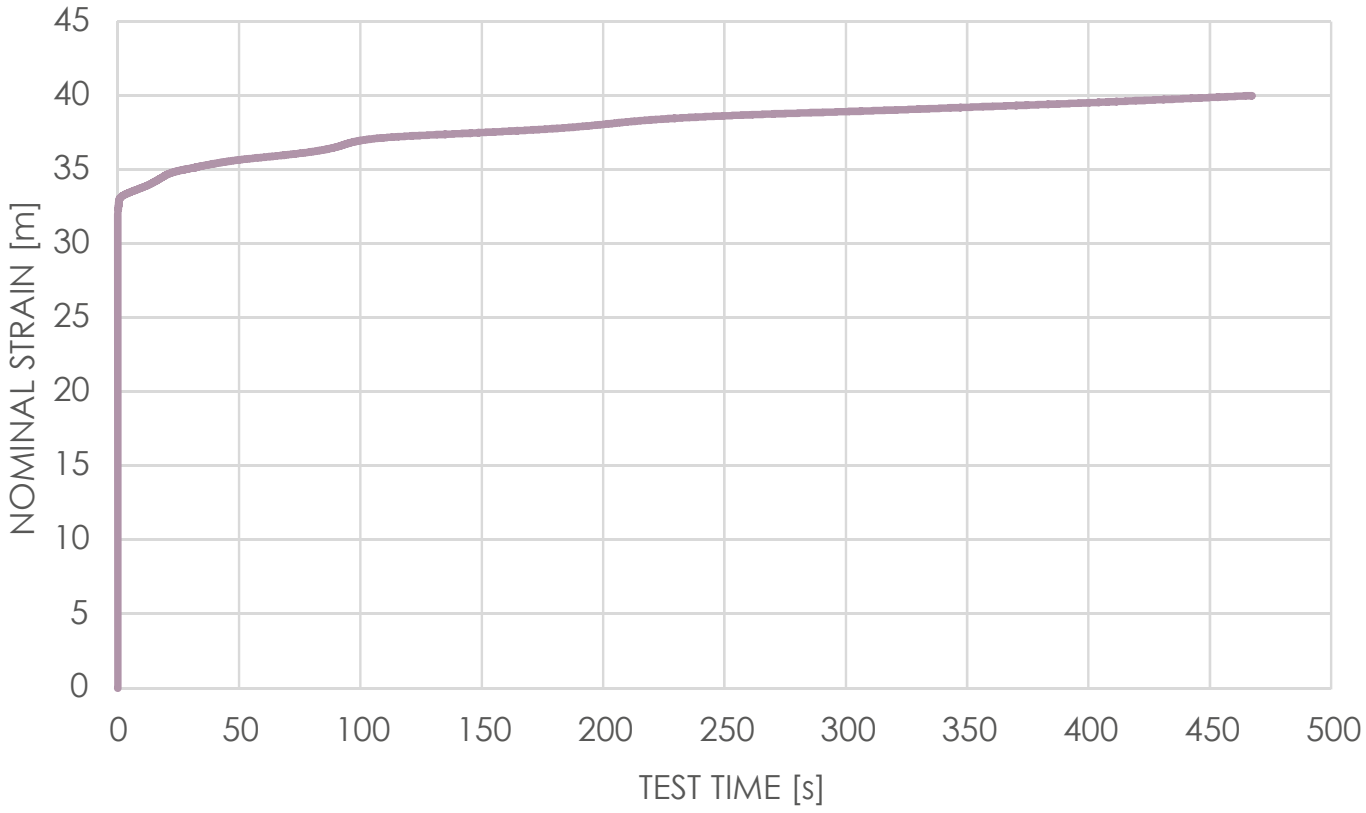
returning to its original state.

This behaviour was observed for panel 03 and 04, as they were seemingly deformed under the applied force and returned to its original form once the force was removed. This behaviour of these panels encouraged us to do a separate test to see its structural behaviour under constant stable force. See Graph 7 for the results.

Graph 7 shows an unstable behaviour under constant pressure, which is not a desired performance for a structural material.

IV. II. IV. II CONCLUSION

It was decided that these panels cannot be used for structural purposes due to their unstable behaviour. However they could still be used in the project as panels and they could support their own weight when a substructure was provided.



Graph 6: Panel 04 creep test result

IV. III. COMPUTATIONAL DESIGN

IV. III. I. INTRODUCTION

There has been two separate approaches explored for this thesis. This chapter will explain both approaches and how they are created using specific inputs, Karamba3D library, custom python and Karamba3D components for Grasshopper. Neither of the processes are fully functioning, however especially the second approach gives promising results for further research.

First approach is looking at a case where the main design input is a surface, specifically shell structure. This rough surface is then topologically optimized to create and optimized geometry using SKO.

Second approach receives a rough design volume, support areas and desired void areas as input and created a topology optimization through voxelization.

IV. III. II. INITIAL APPROACH

IV. III. II. I WORKFLOW DIAGRAM

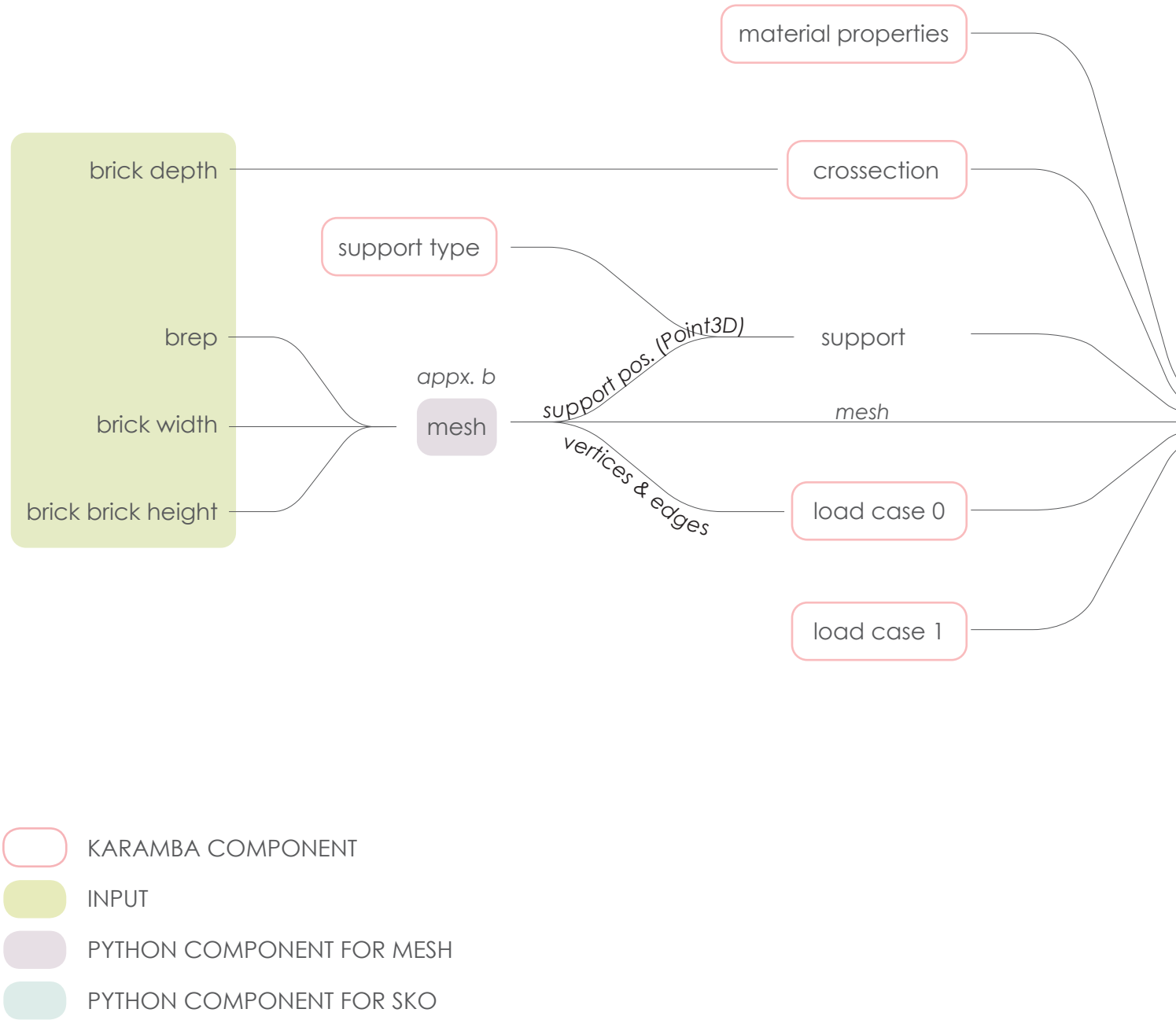
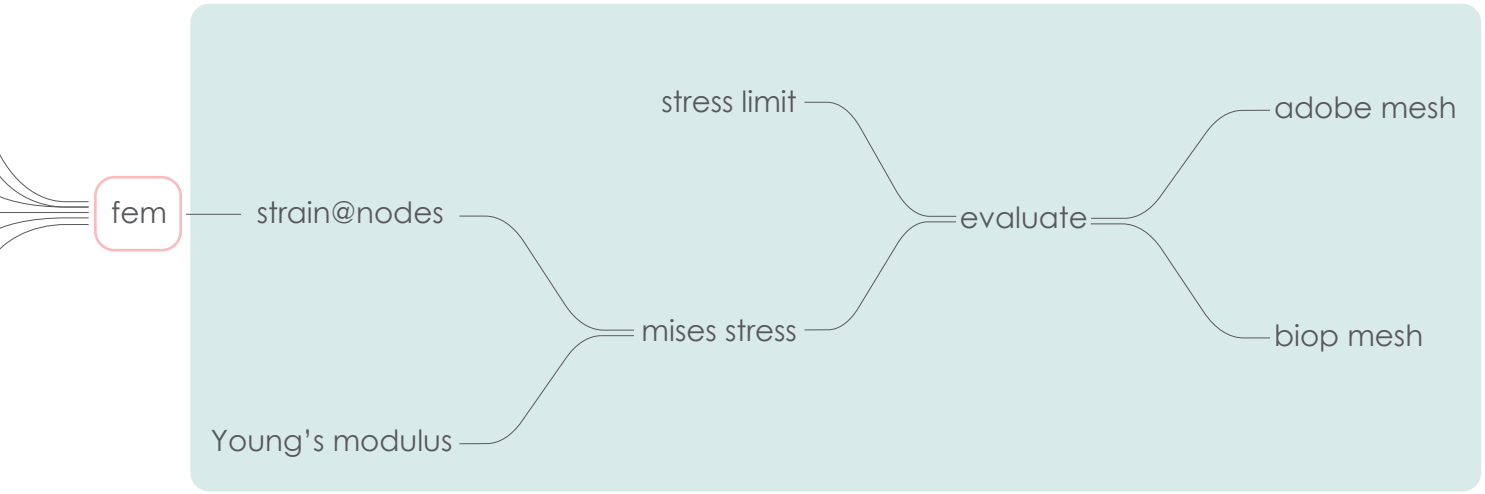


Figure 4.11: SKO flow diagram


fem
strain@nodes
stress limit
Young's modulus
mises stress
evaluate
adobe mesh
biop mesh

IV. III. II. II. GRASSHOPPER SCREENSHOT

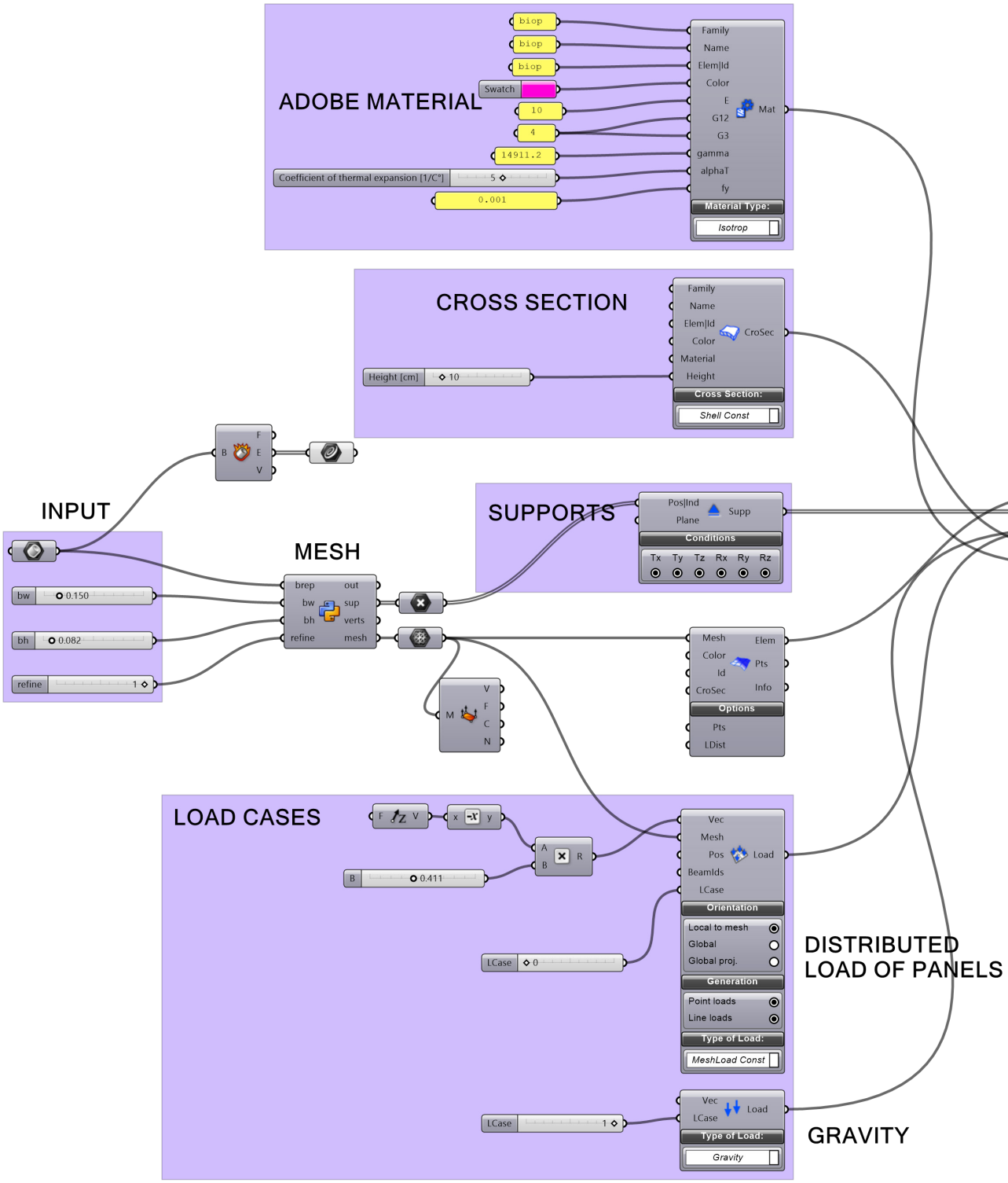
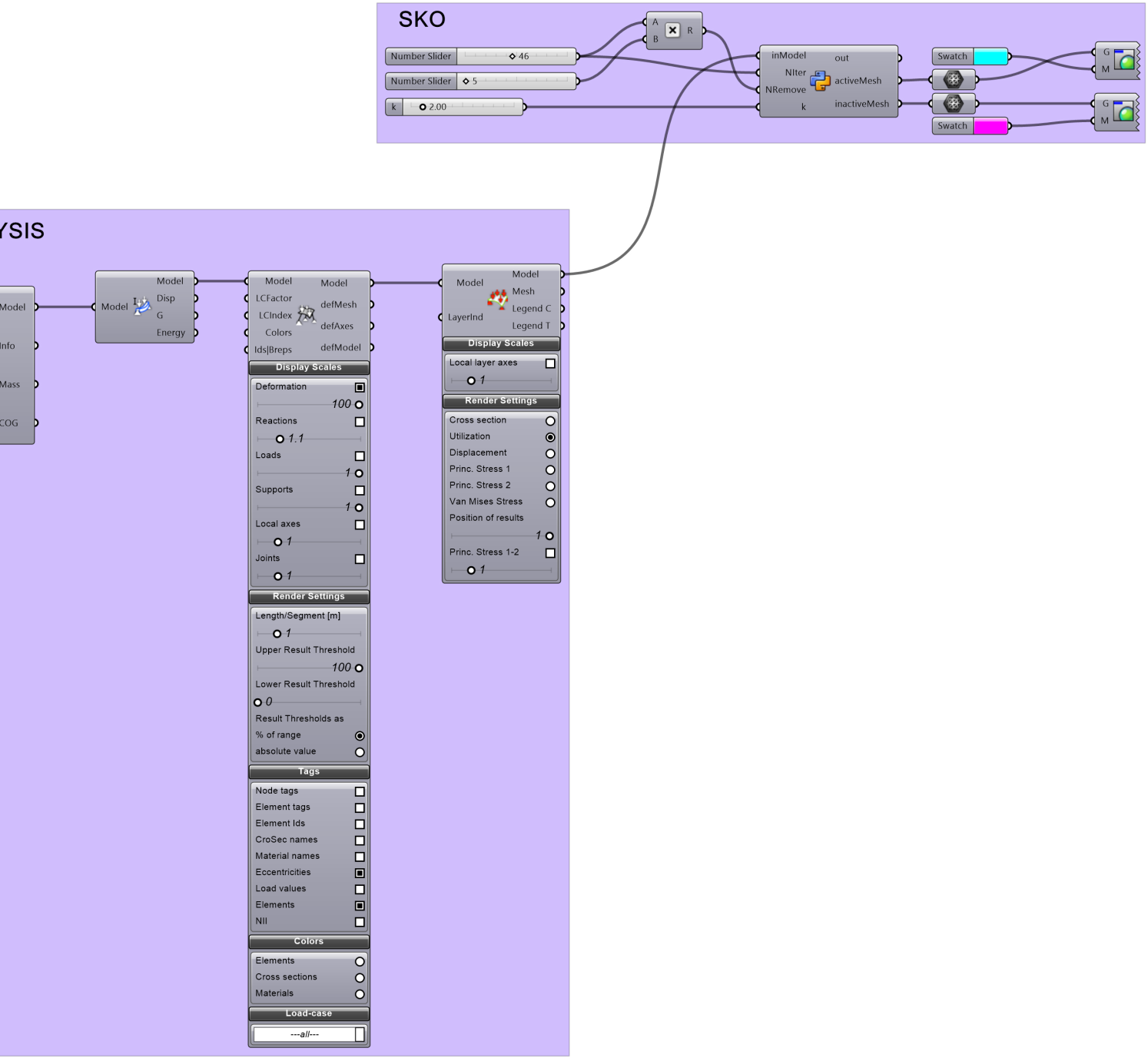


Figure 4.12: SKO Grasshopper screenshot


SKO
Number Slider
46
Number Slider
5
k
2.00
A
B
R
inModel
NIter
NRemove
k
out
activeMesh
inactiveMesh
Swatch
Swatch
G
M
G
M
YSIS
Model
Model
Disp
G
Energy
Model
LCFactor
LCIndex
Colors
Ids|Breps
Model
defMesh
defAxes
defModel
Display Scales
Deformation
100
Reactions
1.1
Loads
1
Supports
1
Local axes
1
Joints
1
Render Settings
Length/Segment [m]
1
Upper Result Threshold
100
Lower Result Threshold
0
Result Thresholds as
% of range
absolute value
Tags
Node tags
Element tags
Element Ids
CroSec names
Material names
Eccentricities
Load values
Elements
NII
Colors
Elements
Cross sections
Materials
Load-case
---all---
Model
LayerInd
Model
Mesh
Legend C
Legend T
Display Scales
Local layer axes
1
Render Settings
Cross section
Utilization
Displacement
Princ. Stress 1
Princ. Stress 2
Van Mises Stress
Position of results
1
Princ. Stress 1-2
1

IV. III. II. III. SKO PYTHON COMPONENT

```
import rhinoscriptsyntax as rs
import clr

clr.AddReferenceToFileAndPath("C:\Program Files\Rhino 6\Plug-ins\karamba.gha")

import Karamba.Models.Model as Model
import Karamba.Elements.ModelShell as Shell
import feb.ShellMesh as ShellMesh
import feb.TriShell3D as TriShell3D
import feb.VectSurface3DSigEps as TriStates
import feb.Deform as Deform
import feb.Response as Response
import Rhino.Geometry as rg

import Karamba.Materials as fm
from operator import attrgetter

#import feb model

femodel = model.Clone()
femodel.deepCloneFEModel()
```

Karamba3D 1.1.0 Hacker's Essentials [24] folder come with an example file names "SimpleShellEso" [23] for ESO. This script is drived from the ESO component provided.

The file first runs a Karamba analysis for a given shell. Then, it brings the results into a python component. The resulting "Model" output of the "Karamba Shell View" component is imported into python as "model".

The other two inputs are NRemove and Niter. They both only accept integer. NRemove limits the number of removal of nodes (soft kill) that is allowed in each iteration. Niter determins the number of iterations.

Inside the component it's cloned and converted into finite element model.

```
#bioplastic material
bf = "biop"
bn = "biop"
bE = 1.35
bG = 0.65
bgamma = 0.6
balphaT = 0.5
bfy = 1.1

biop = fm.FemMaterial_Isotrop(bf,bn,bE,bG,bG,bgamma,bfy,balphaT,None)
biop.addTo(femodel.febmodel)
feb_biop = femodel.febmodel.material(femodel.febmodel.numberOfMaterials()-1)

#adobe material
af= "adobe"
an= "adobe"
aE = 350
aG = 150
agamma = 14.5
aalphaT = 5
afy = 0.0005

adobe = fm.FemMaterial_Isotrop(af,an,aE,aG,aG,agamma,afy,aalphaT,None)
adobe.addTo(femodel.febmodel)
#feb_adobe = femodel.febmodel.material(femodel.febmodel.numberOfMaterials()-1)
```

Following, created both bioplastic and adobe materials based on the literature research and material study finding. The created materials are also added into the finite element model.

```
class skoVert:
    def __init__(self,shell_elem,totstress):
        self.active = True
        self.fitness = 0
        self.shell_elem = shell_elem
        self.area = shell_elem.area()
        self.node = node
        self.stress = self.shell_elem.strain(femodel.febmodel,crosec,self.node)*aE
    def update(self):
        self.fitness = self.stress/totstress

sko_verts = []
for i in femodel.elems:
    if type(i) != Shell:
        continue
    tri_mesh = femodel.febmodel.triMesh(i.fe_id)
    for j in xrange(tri_mesh.numberOfElems()):
        sko_verts.append(skoVert(tri_mesh.elem(j),j))
```

Following a new class for each node created with attributes that will be used for evaluation of the node.

The major difference between the SKO and the ESO is the fitness criteria. SKO evaluates mises stress. If this value is below the limit, it simply "soft kills" the node. However Karamba does not provide mises stress value for tri-mesh nodes, but it does provide the strain on each node. Using strain and Young's modulus, we can calculate the stress on each node. From the nodal stress we can then calculate nodal mises stress.

```
nremove_per_iter = int(NRemove/NIter+1)
n_removed = 0

#SKO iterations
for iter in xrange(NIter):
    analysis = Deform(femodel.febmodel)
    response = Response(analysis)

    try:
        response.updateNodalDisplacements()
        response.updateMemberForces()
    except:
        raise Exception("problem")

    for sko_vert in sko_verts:
        sko_vert.update()
```

In each iteration the model is let to deform and update the attributes of each node created in class skoVert.

```
sko_verts = sorted(sko_verts, key = attrgetter("fitness"))
    n_removed_per_iter = 0
change = False

for sko_vert in sko_verts:
    if (n_removed >= NRemove):
        break
    if (n_removed_per_iter >= nremove_per_iter):
        break
    if (sko_vert.active == False):
        continue
    sko_vert.shell_elem.material(feb_biop)
    sko_vert.active = False
    n_removed+=1
    n_removed_per_iter +=1
change = True
if (change == False):
    break
```

Depending on the result, within the limits rpovded by number of iterations and allowable number of nodes removal the nodes are removed from the mesh. (soft kill process)

```
femodel.febmodel.touch()
#split mesh based on material
adobemesh = rg.Mesh()
biopmesh = rg.Mesh()

for i in xrange(femodel.febmodel.numberOfNodes()):
    feb_pos = femodel.febmodel.node(i).pos()
    adobemesh.Vertices.Add(rg.Point3d(feb_pos.x(),feb_pos.y(),feb.pos.z()))
    biopmesh.Vertices.Add(rg.Point3d(feb_pos.x(),feb_pos.y(),feb.pos.z()))

for sko_vert in sko_verts:
    ind0 = sko_vert.shell_elem.node(0).ind()
    ind1 = sko_vert.shell_elem.node(1).ind()
    ind2 = sko_vert.shell_elem.node(2).ind()

    if (sko_vert.active):
        adobemesh.Faces.AddFace(rg.MeshFace(ind0,ind1,ind2))
    else:
        biopmesh.Faces.AddFace(rg.MeshFace(ind0,ind1,ind2))
```

Finally two different meshes are created as one mesh being “soft-killed” (bioplastic), the other being the structural (adobe).

A similar approach could also be applied for shape optimization (CAO).

IV. III. III. SECOND APPRAOCH

IV. III. III.I INTRODUCTION

This approach as explained in section IV. III. I. Uses volumes and voxelization to receive a topologically optimized form.

This chapter will further explain how Karamba3D plugin for Grasshopper was used for finite element modeling and analysis, how custom voxel class was utilized to create a coherant flow of data and some of the major issues that were encountered throughout the process. While some problems were solved, others remained unsolved within the time frame given for this project.

IV. III. III.II DISCRETIZATION OF THE VOLUME

For finite element modeling, Karamba3D plugin for grasshopper was used for this project. One of the first problems with this tool was that it does not provide FEM for solid object. Karamba3D only provides FEM for shell structures, beams and trusses.

In order to overcome this obstacle, using the center points of voxels were used to create a three-dimensional grid of beams with the square cross section of the voxel size. These beams assumed to be acting like the actual volume.

IV. III. III.III CUSTOM VOXEL CLASS ATTRIBUTES

As voxels fills up the given rough design volume, they are given x, y and z value based on their sequence in the volume. While this gives a unique index number for each voxel, it also makes it easier to find the adjacent voxels.

self.x = int
self.y = int
self.z = int
self.index = self.x, self.y, self.z

This attribute creates the visual Rhino interface representation of the voxel.

self.vox = rg.Box()

If the voxel represents void this value is set to False, otherwise it is set to True.

self.active = bool

If the voxel's center is inside the support region this value is set to True, otherwise False.

self.support = bool

This attribute recieves the global coordinates of the center of the voxel.

self.center = rg.Point3d()

This value shows the voxels adjacent to the given voxel as a list of six booleans. The voxels are checked based on their self.index and if their self.Active Is true depending on their location it changes that item in the list to true or false.
Item 0 corresponds to x-1
Item 1 corresponds to y-1
Item 2 corresponds to x+1
Item 3 corresponds to y+1
Item 4 corresponds to z+1
Item 5 corresponds to z-1

self.adjacency=[bool,bool,bool,bool,bool,bool]

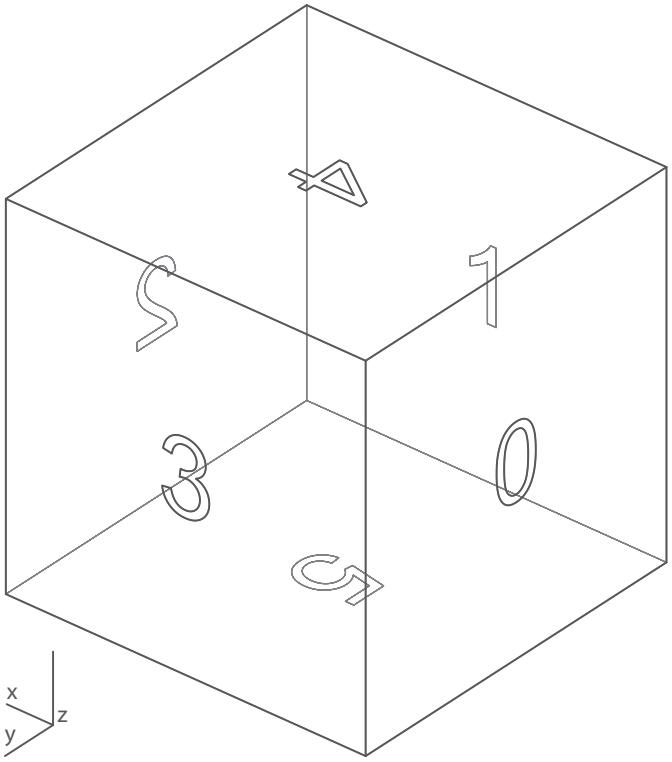


Figure 4.13: Voxel visual representation

Axial force values obtained from Karamba3D on the beams connected to the voxel assigned to the voxel as shown below

self.totalForceX = float
self.totalForceY = float
self.totalForceZ = float

The values received from axial values are divided into cross section area of the beams to receive total stress values on each axes.
self.totalStressX = float
self.totalStressY = float
self.totalStressZ = float
self.totalStress = self.totalStressX
+ self.totalStressY + self.totalStressZ

IV. III. III. III. GENERAL WORKFLOW DIAGRAM

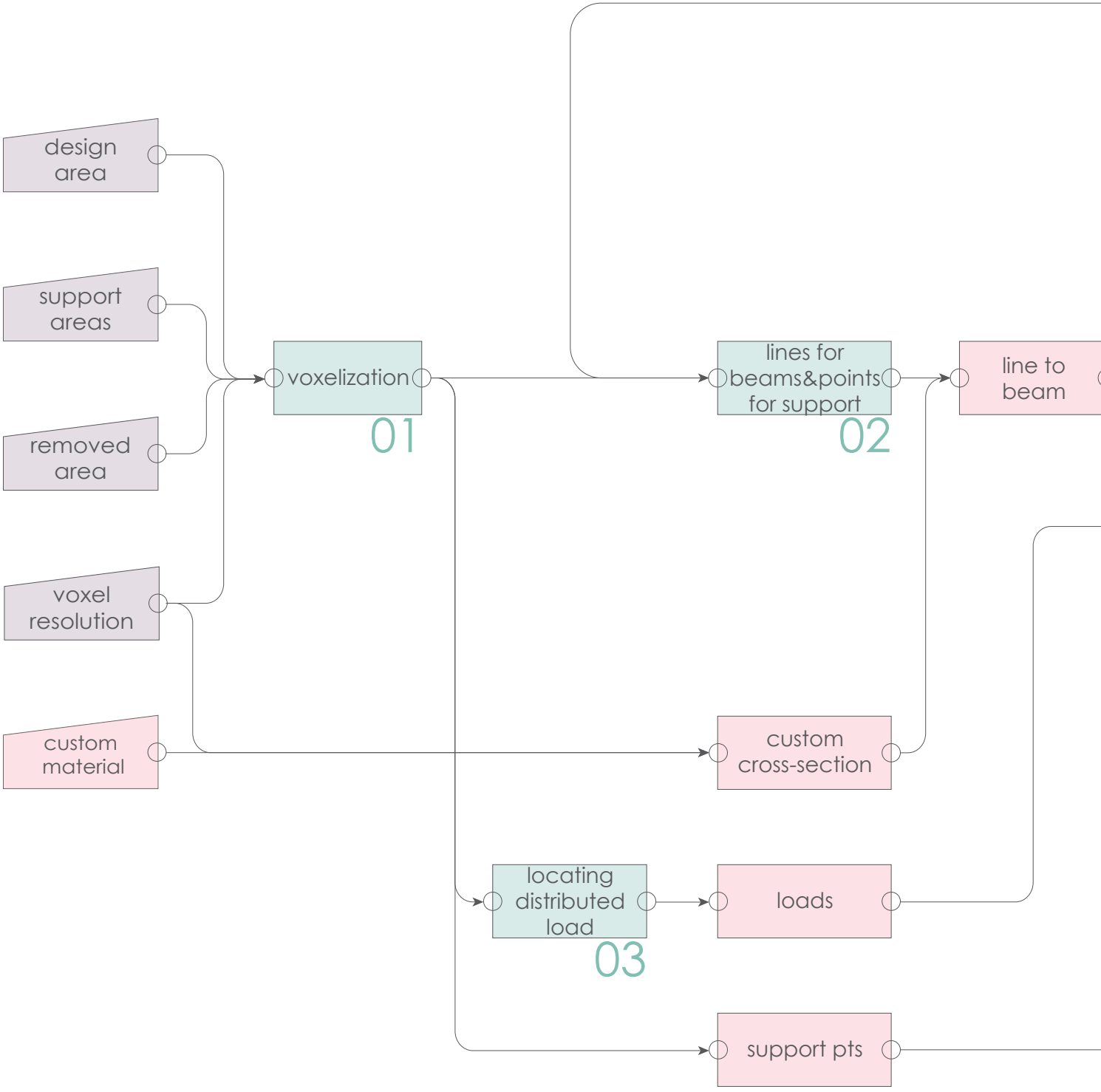
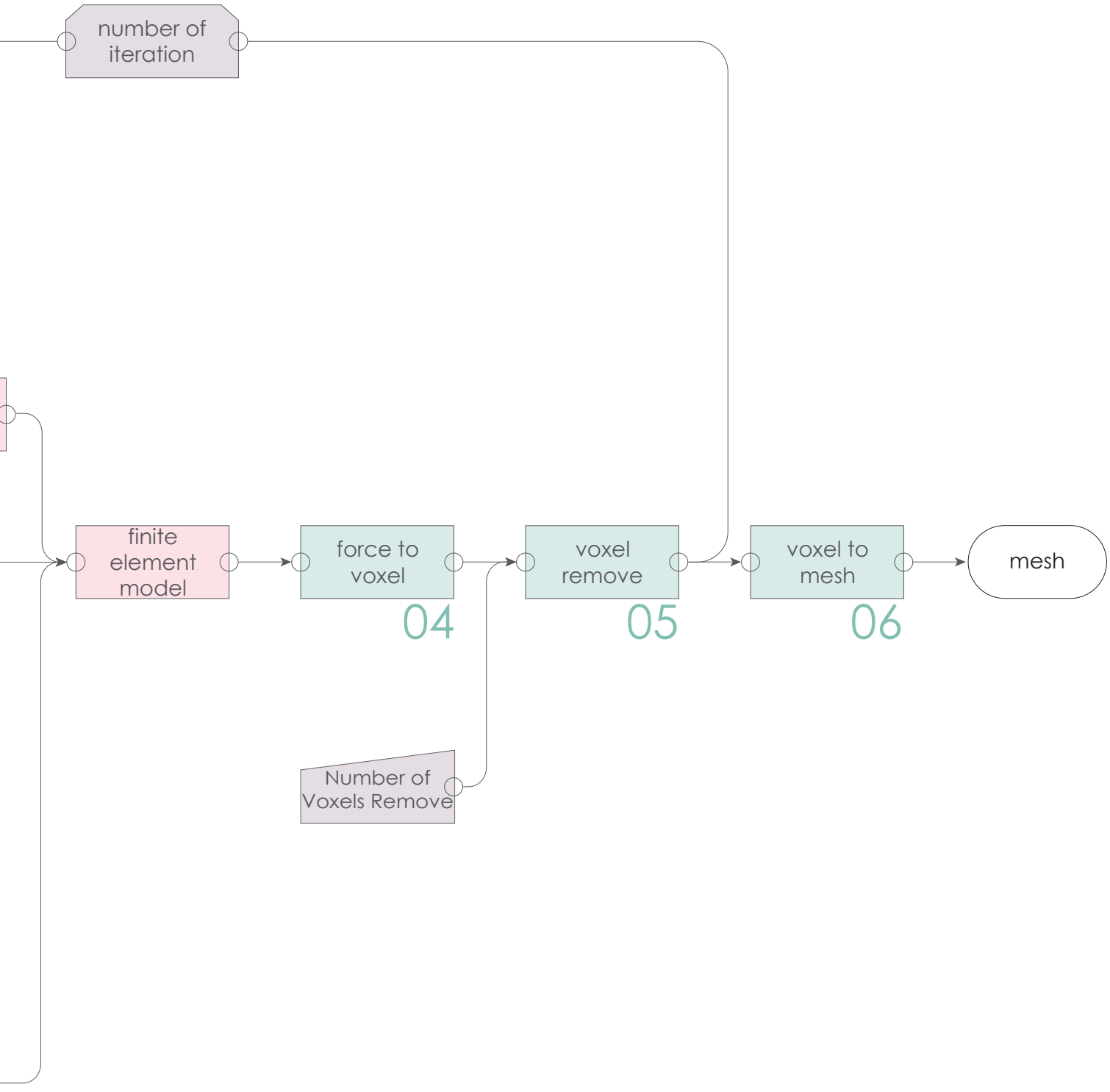


Figure 4.14: General workflow diagram



00 REFERENCE NO. FOR APPENDIX

PYTHON

KARAMBA 3D COMPONENT

MANUAL INPUT

IV. III. III. IV. LOW RESOLUTION ITERATIONS

Figure 4.15: Low Resolution iterations of the code

Voxels Removed each Iteration = 10
No of Iterations = 13

IV. III. III.II LOOPING

Using Karamba3D plugin for FEM, even though Karamba3D's library partially accessible, it was not very easy to navigate through as there is no real documentation of it. Within the given time, desired functions were not able to figured out and therefore the looping had to be done manually.

The data was collected at the end of the line through Grasshopper data component. The Data was internalized and re-plugedin at the beginning of the script.

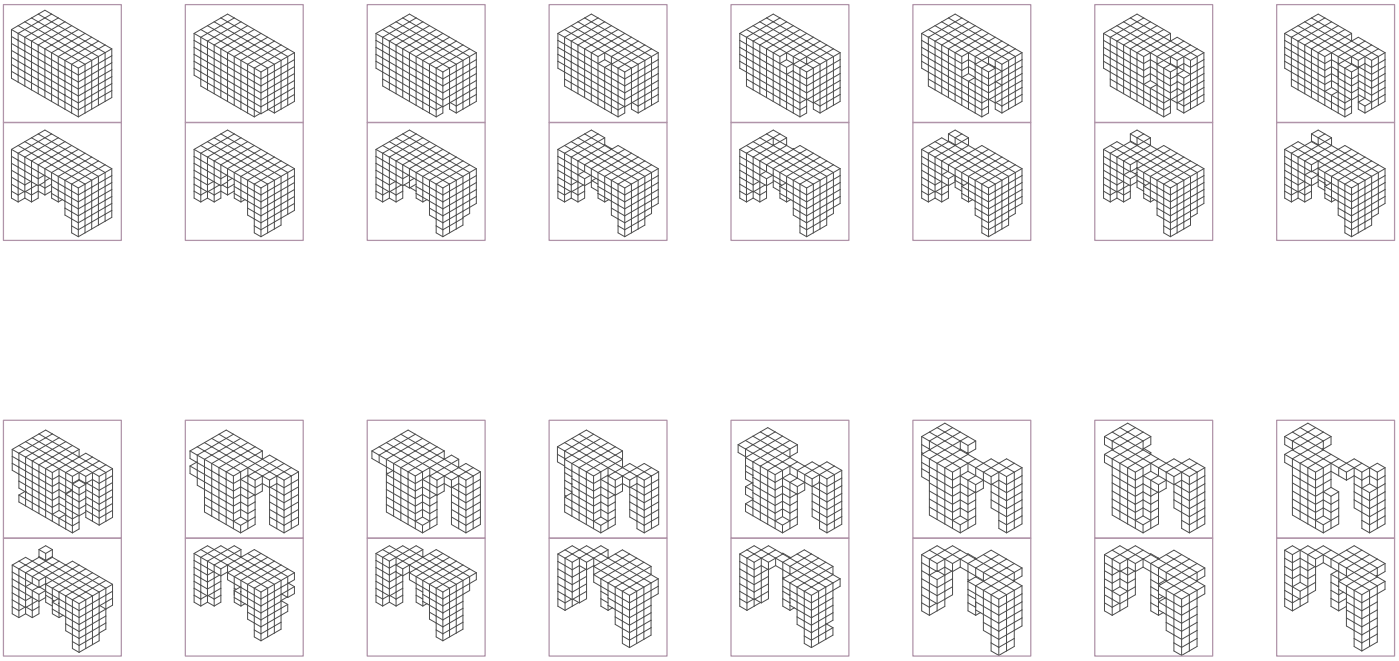


Figure 4.16: Low Resolution iterations of the code fixed with DFS

Voxels Removed each Iteration = 10
No of Iterations = 16

IV. III. III.III SINGLE VOXELS

After a number of iterations, it was observed that there were a number of individual voxels that were remaining outside the main body. This resulted in individual beams floating in space without any support and causing the analysis to be wrong. In order prevent this problem, after each iteration a check point was created for each voxel using Depth First Search (DFS) algorithm using the beams as edges and voxels as vertices. This way, after each iteration, if there is a voxel that is not connected to the first support, that voxel was removed from the iteration.

IV. III. III. IV. GRASSHOPPER SCREENSHOT

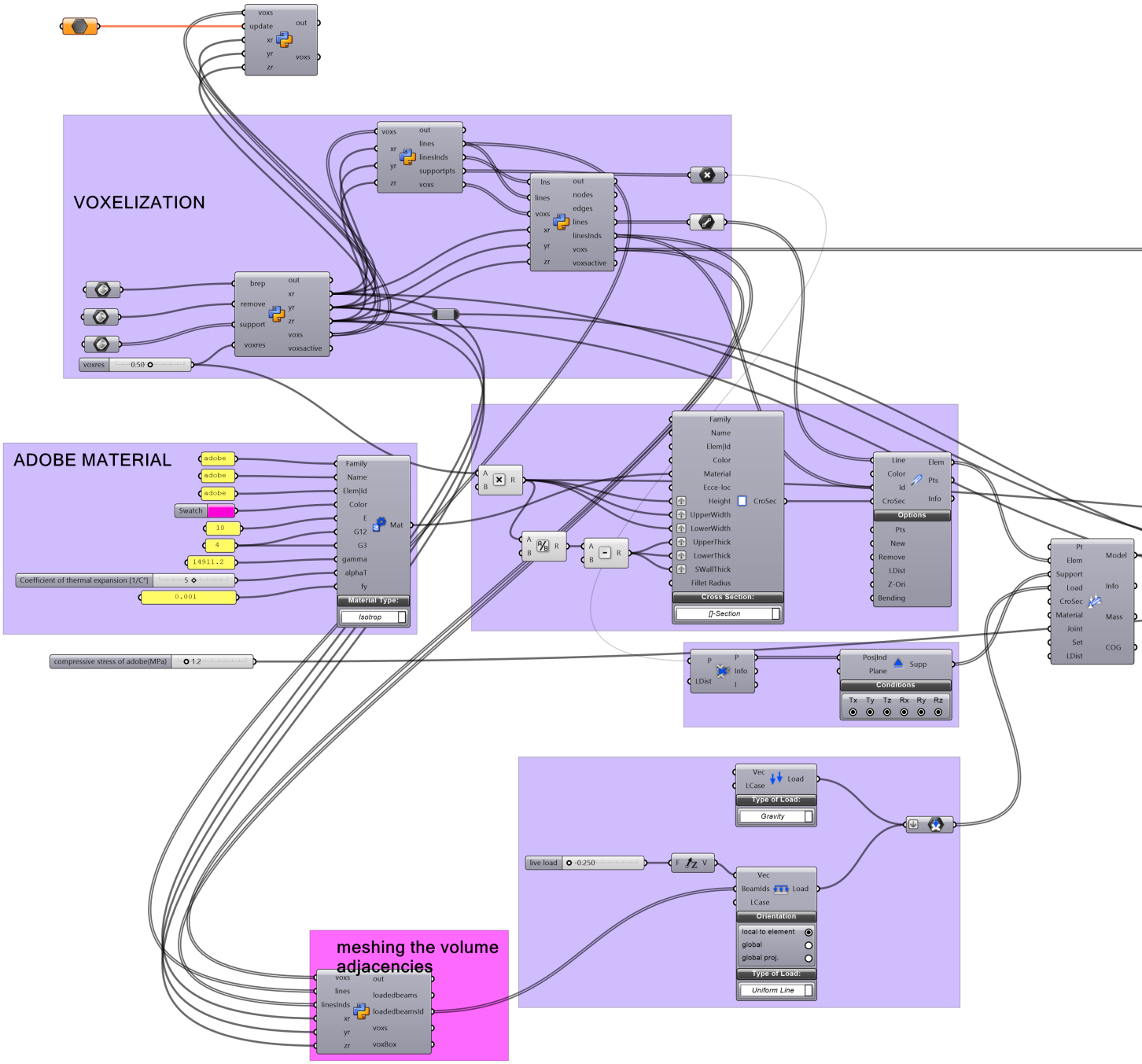
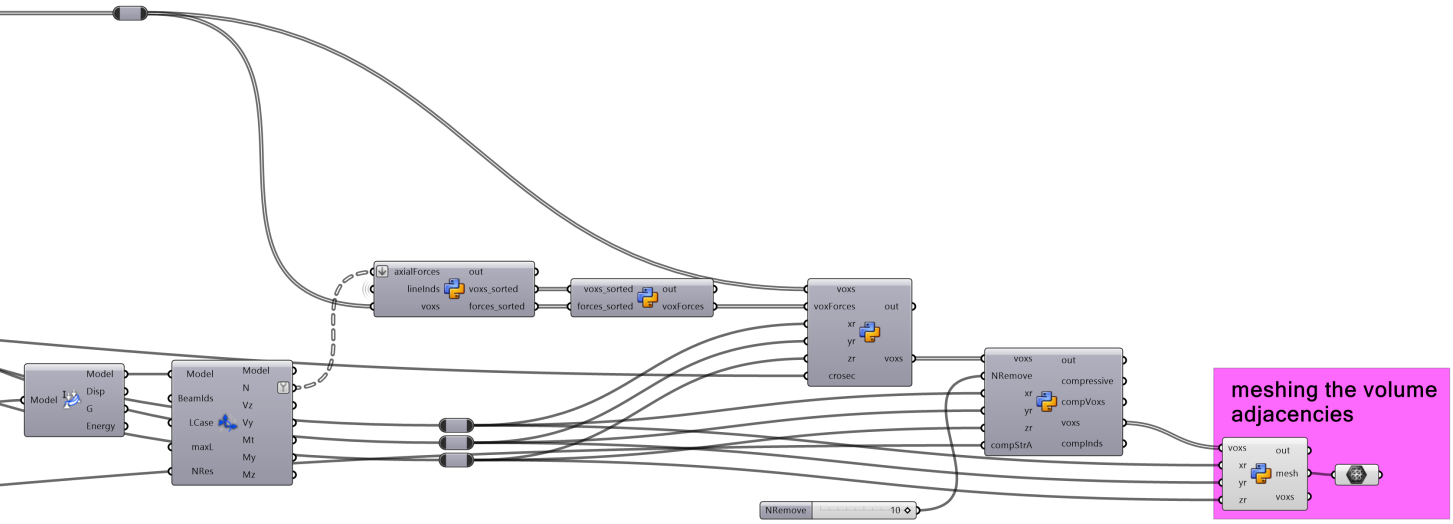


Figure 4.17: Grasshopper interface of the code

Voxels Removed each Iteration = 10
No of Iterations = 16


axialForces
lineInds
voxs
out
voxs_sorted
forces_sorted
voxs_sorted
forces_sorted
out
voxForces
voxs
voxForces
xr
yr
zr
crosec
out
voxs
Model
Model
Disp
G
Energy
Model
BeamIds
LCase
maxL
NRes
Model
N
Vz
Vy
Mt
My
Mz
voxs
NRemove
xr
yr
zr
compStrA
out
compressive
compVoxs
voxs
compInds
NRemove
10
meshing the volume adjacencies
voxs
xr
yr
zr
out
mesh
voxs

V. CONCLUSION

V. I. INTRODUCTION

The main objective followed by the secondary objective of this project were proposed in chapter I. III. Reseach Objective repeated here as follows:

to develop a computational approach to optimize material uses of the construction materials -earth and bioplastics- in response to structural requirements.

In pursuance of the main objective, following research questions as proposed in chapter I.IV. Research Questions will be answered in this chapter, followed by further recommendations.

1. How to develop building unit forms for bioplastics and earth in consideration of the material properties?

2. How to compute an architectural form in consideration of the building units and material performance of the units?

3. How to optimize material use in the given design problem?

V. II. CONCLUSION

1. How to develop building unit forms for bioplastics and earth in consideration of the material properties?

BIOPLASTICS

The design of the bioplastic elements followed a modular approach. They are dimensioned with the assumption of a mud substructure. Therefore, the unit dimensions were decided with potential brick forms and dimensions in mind. Furthermore, the elements are designed to have joint locations implemented in the design. Finally, the geometry of the panels was chosen to be triangular in order to be adjustable to potential curvatures of the design.

In order to determine the most suitable recipe for the bioplastic units four different recipes went through tensile testing. As a result, Type 04 recipe was chosen to be the most suitable recipe for the elements. This recipe requires 1 g of glycerine, 50 g of water and 23 g of gelatine. With this recipe one can achieve tensile strength of 32.71 MPa, yield strength of 33.59 MPa and Young's Modulus of 11.41 MPa. As a reference, according to [10] general purpose unfilled Polylactide (PLA) has the tensile strength of 47-70 MPa, Yield Strength of 55-72 MPa and Young's Modulus of 3.3-3.6 GPa. Please refer to Chapter III for more detailed analysis of this process.

During the fabrication of the samples for the tensile test, it is observed that the bioplastic material is subject to significant shrinkage once it's removed from the mould to cure. This process requires additional framing of the element during the curing process to keep the element in tension for maintaining the desired form. This behaviour of the material heavily influenced the design of the bioplastic element. Deformation still occurs during the curing process, however when the element is tensioned intentionally, this deformation becomes advantageous as the shrinkage causes a level of curvature which increases the area moment of inertia of the element. For further explanation of the fabrication process and its influence on the form, please refer to chapter IV. II. Unit / Joint Design.

As mentioned before, the joint locations were implemented in the element design. While the joint design was only left speculative, it's design can be seen in chapter IV. II. III. Assembly. This joint, depending on the load case, can be made out of steel, PLA or recycled plastics.

ADOBE BRICKS

This project did not focus heavily on adobe part of materiality. All the assumptions regarding the adobe was derived from the literature research.

As a result, this project proposes to use an adjustable mould for adobe bricks that will follow the curvy form of the bus stop. The mould expected to have two parallel edges with changing angles on the other two edges. This proposal was considered due to symmetrical form of the bus stop design. This form would require the each brick to be produced at least twice. However, this proposal was only remained in theory and never been tried to be actually implemented into the form.

FURTHER RECOMMENDATIONS

BIOPLASTICS

Bioplastics of this kind is recently introduced to the design world. While it's exciting to be able to make bioplastics in your kitchen, there are also quite a field to explore.

There needs to be further research on what kind of biodegradable additives could be used with this material to increase its mechanical properties. Furthermore, currently this material is very weak against UV and water as well as being highly flammable. There are some research on additives to increase its water resistance including beeswax and gum arabic. However, I haven't encountered anything in regards to UV and fire protection.

As described throughout the report, these bioplastics are 4D materials, meaning that they change their form over time. A detailed investigation into how 4D materials in general can influence architectural design and how time specifically influence bioplastic's form could be beneficial for the further use of this material in architecture. One are in particular could be the influence of deformation into structural behaviour and if this can be manipulated intentionally to achieve desired structural behaviour.

In regards to specific panel design of this project, further structural tests could be run to understand the scalability of the material. Different joinery designs could be proposed.

For the implementation of this design into a computational design, a script could be developed to create the panelization automatically.

ADOBE BRICKS

On the adobe side, a script can also be developed for adobe placement. For that, the meshing script could be used as the base.

Further research into adjustable brick mould could be useful for this and other projects. In relation to adjustable mould, a script could be developed to reduce the variety of unique brick forms to increase the pace of the production, as changing the mould would increase the production time.

Finally, it would be interesting to control biodegradably, and design the structure to control biodegredation.

2. How to compute an architectural form in consideration of the building units and material performance of the units?

The form in this project could be divided into two parts. First is the rough design area to be optimized, and second is the form after the optimization process. This chapter will focus on the former and the next chapter will be focusing on the latter.

Determining the rough design area consists of five main aspects; overall rough design area, support locations, material properties, void regions and load cases. All these five aspects are determined by the designer, based on the program of the assignment and general material behaviour.

To determine these three aspects, a brief investigation to the existing bus stop design was conducted. This analysis informed the design in terms of overall size and the programmatic requirements of the structure. Depending on the location, a single bus stop size can vary between 2 x 3 m and 2 x 4 m. Three main elements stood out in all the sample bus stops; a canopy, a bench and a vertical surface to display the bus schedule.

Following this analysis, the footprint of the structure was determined to be 2 x 4 m. Unlike typical rectangular bus stops, this structure was designed to be triangular through its support points. This decision was made for aesthetical reasons while still creating a modular form. The bus stops, depending on its location may serve one side or either side of the sidewalk, and these bus stops can be used either individually or stacked next to each other facing both sides of the sidewalk.

Finally, there were two load cases assigned to the structure. One is the self-load of adobe and the other is the distributed load of bioplastic panels across the structure. This way, while the panels create canopy for people where the adobe is absent, they also create canopy for adobe where it's present.

The proposed form requires strong bracing at the support points. Therefore, the seating areas are intentionally placed to serve this purpose. This way a program requirement was provided while serving a structural purpose.

Finally, there were two load cases assigned to the structure. One is the self-load of adobe and the other is the distributed load of bioplastic panels across the structure. This way, while the panels create canopy for people where the adobe is absent, they also create canopy for adobe where it's present.

V. II. III. OPTIMIZATION

3. How to optimize material use in the given design problem?

Among many topology optimization methods, this project chose to apply Mattheck's Soft-Kill Optimization (SKO) [15],[18]. This decision made both for conceptual and practical reasons.

Conceptually it works because this project started as a fascination towards natural algorithms, and Mattheck's method does exactly that. He derives the SKO method based on his observations on bones and the CAO method on trees [15],[18].

Practically it was also desirable as it is a fairly intuitive optimization method and compare to most optimization methods it's easier to apply.

FURTHER RECOMMENDATIONS

Optimization Method:
This project used SKO and CAO method for topology and shape optimization. Alternative optimality criteria methods such as SIMP or other heuristic methods such as ESO or BESO could result in more optimal forms – higher structural performance with less amount of material.

Checkerboard filter:
Adding a checkerboard filter as discussed in chapter II. III. I. II ESO & BESO, could improve the results of the topology optimization and would generate a mesh that is more suitable for shape optimization.

Structural Analysis:
This project used Karamba 3D plugin for Grasshopper as the FEM and FEA. Further structural analysis and optimization could be achieved with more advanced FE softwares.

Topology Optimization as a Design Method:
This is the approach I find the most fascinating. Either on computer or on Virtual Reality, a design process could be created using voxelation and live FEM. In this scenario, you will be facing the the design area and that would be filled with voxels. You can start carving spaces in this domain and you would start seeing the stress lines, which would help you make intuitive decisions as you carve interior spaces in the voxels.

In the traditional design process, our designs increase in scale as the design starts getting more details. This change in scale could be achieved by decreasing the voxel size. This method can create a new subtractive approach to the design process. A script or even a computer program can be designed for this purpose.

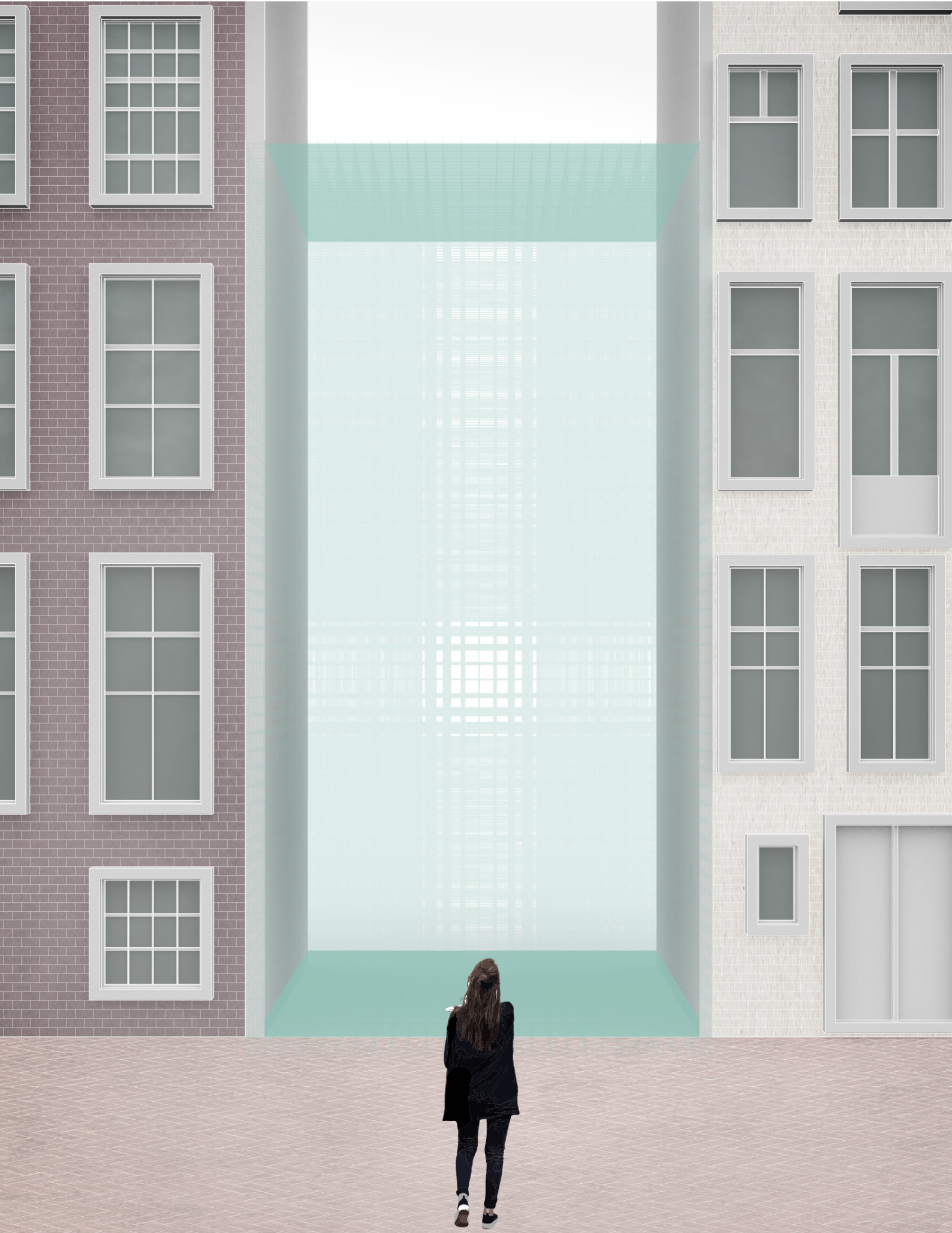


Figure 5. 1: Imagination of a future application

V. IV. REFLECTION

As part of Sustainable Graduation Studio graduation studio, my project was focusing on Structural Mechanics and Design Informatics. The main focus of this project was to develop a computational approach to optimize material uses of the construction material -earth and bioplastics in response to structural requirements. Therefore, the computational approach was addressing the Design Informatics while material properties and topological optimizations were addressing the structural mechanics topics.

The research approach did not work out as smoothly as I was hoping. Not every code I wrote worked out, and I had to change my approach many times which caused me to waste a lot of time, even though those times spent were valuable later as I get better and faster at writing codes. However, I believe I get to the point I aimed for, but I didn't have enough time to explore my arrival point. The design was the process for the research. Therefore, the design was not the main goal but was the enabler of the learning process.

Many of the aspects of the project is experimental. Therefore, the project itself is not a product that can be readily available for public or the practice of architecture. However, many of the codes, tests and findings could be applied to further research within the practice. The projected innovation has been achieved on the theoretical level, and proof of concept level. However, the innovation wasn't ready to be used in practice.

The project contributes to different aspects of sustainable development. On the material end of the project, the production of a panel out of bioplastics was fairly new to the field of architecture. However, the bioplastic panel is not developed to be an off-the-shelf product. It still needs serious further research to be implemented in a larger scale. The use of topology optimization is nothing new in engineering and architecture. However, it's implementation into design process in architecture is not as common. Typically, these processes added on to the design after the design is established. As a result, the project does not propose an ultimate answer or solution to use of biodegradable materials in architecture or reducing material use by topology optimization in design, it does contribute a new element into the pool of research for the future researchers to move forward. The project didn't necessarily focus on socio-cultural aspects of architecture, however it's positive impact on environment can be interpreted as ethical impact. The biggest moral issue I was confronted with was the use of edible materials. However, I still believe it's more acceptable to use edible materials, in a country where there is already plenty of food wasted, than using materials that are essentially bad for the environment, the origin of food.

VI. TERMINOLOGY

Biobased: "The term 'biobased' means that the material or product is (partly) derived from biomass (plants). Biomass used for bioplastics stems from e.g. corn, sugar cane, or cellulose" [16].

Biodegradable: "Biodegradation is a chemical process during which micro-organisms that are available in the environment convert materials into natural substances such as water, carbon dioxide, and compost (artificial additives are not needed). The process of biodegradation depends on the surrounding environmental conditions (e.g. location or temperature), on the material and on the application." [16].

Checkerboard problem: "A checkerboard is defined as a periodic pattern of high and low values of Pseudo-densities...arranged in a fashion of checkerboards. This behaviour is undesirable as it is the result of a numerical instability and does not correspond to an optimal distribution of material. The checkerboards possess artificially high stiffness, and also such a configuration would be difficult to manufacture." [7].

Mises Stress: "where the stress of one element, $\sigma_e^{\upsilon m}$, is compared with the maximum stress of the whole structure $\sigma_{max}^{\upsilon m}$"[3].

VII. REFERENCES

[1] G. Minke, Building with Earth. Birkhaeuser, 2006.

[2] J. D. Deaton and R. V. Grandhi, "A survey of structural and multidisciplinary continuum topology optimization: post 2000," Struct. Multidiscip. Optim., vol. 49, no. 1, pp. 1–38, 2014.

[3] F. A. Gonçalves, "Introduction to Structural Optimization using the ESO and BESO Evolutionary Methods," 2018.

[4] Margaret Dunne, "Bioplastic Cook Book," 2018. [Online]. Available: https://issuu.com/nat_arc/docs/bioplastic_cook_book_3. [Accessed: 27-Mar-2019].

[5] Kwong Oi-Ying, BIO-PLASTIC HANDBOOK. 2011.

[6] R. Rael, Earth Architecture, 1st Editio. New York: Princeton Architectural Press New York, 2009.

[7] A. Shukla, A. Misra, and S. Kumar, "Checkerboard Problem in Finite Element Based Topology Optimization," Int. J. Adv. Eng. Technol., 2013.

[8] O. Sigmund, "A 99 line topology optimization code written in matlab," Struct. Multidiscip. Optim., 2001.

[9] B. J. Goodno and J. M. Gere, "Tension, Compression, and Shear," in Mechanics of Materials, Ninth Edit., Cengage Learning, 2017.

[10] "CES Edupack 2018." Granta, Cambridge, 2018.

[11] G. Minke, "Working with Earthen Blocks," in Building with Earth, Birkhaeuser, 2006.

[12] M. Bendsoe and O. Sigmund, Topology Optimization: Theory, Methods and Applications. Berlin: Springer International Publishing, 2003.

[13] O. M. Querin, A. Cristina, and P. Marti, Topology Design Methods for Structural Optimization. Academic Press, 2017.

[14] A. R. Bagheri, C. Laforsch, A. Greiner, and S. Agarwal, "Fate of So-Called Biodegradable Polymers in Seawater and Freshwater," Glob. Challenges, vol. 1, no. 4, p. 1700048, 2017.

[15] C. Mattheck, A. Baumgartner, D. Gräbe, and M. Teschner, "Design in nature," Struct. Eng. Int. J. Int. Assoc. Bridg. Struct. Eng., 1996.

[16] "What are bioplastics?"

[17] G. Minke, "The Properties of Earth as a Building Material," in Building with Earth, Birkhaeuser, 2006.

[18] A. Baumgartner, L. Harzheim, and C. Mattheck, "SKO (Soft Kill Option): The Biological Way to Find an Optimum Structure Topology," Int. J. Fatigue, vol. 14, no. 6, pp. 387–393, 1992.

[19] J. Lunt, "Large-scale production, properties and commercial applications of polylactic acid polymers," Polym. Degrad. Stab., vol. 59, no. 1–3, pp. 145–152, Jan. 1998.

[20] G. Minke, "Weather Protection of Loam Surfaces," in Building with Earth, Birkhaeuser, 2006.

[21] J. Viladrich, Bioplastic - Tools and Recipes. 2014.

[22] H. Behrooz and E. Hinton, Homogenization and structural topology optimization: theory, practice and software. London: Springer International Publishing, 1999.

[23] "SimpleShellEso." Karamba3D.

[24] "Karamba3D 1.1.0 Hacker's Essentials." Karamba3D. https://www.karamba3d.com/download/

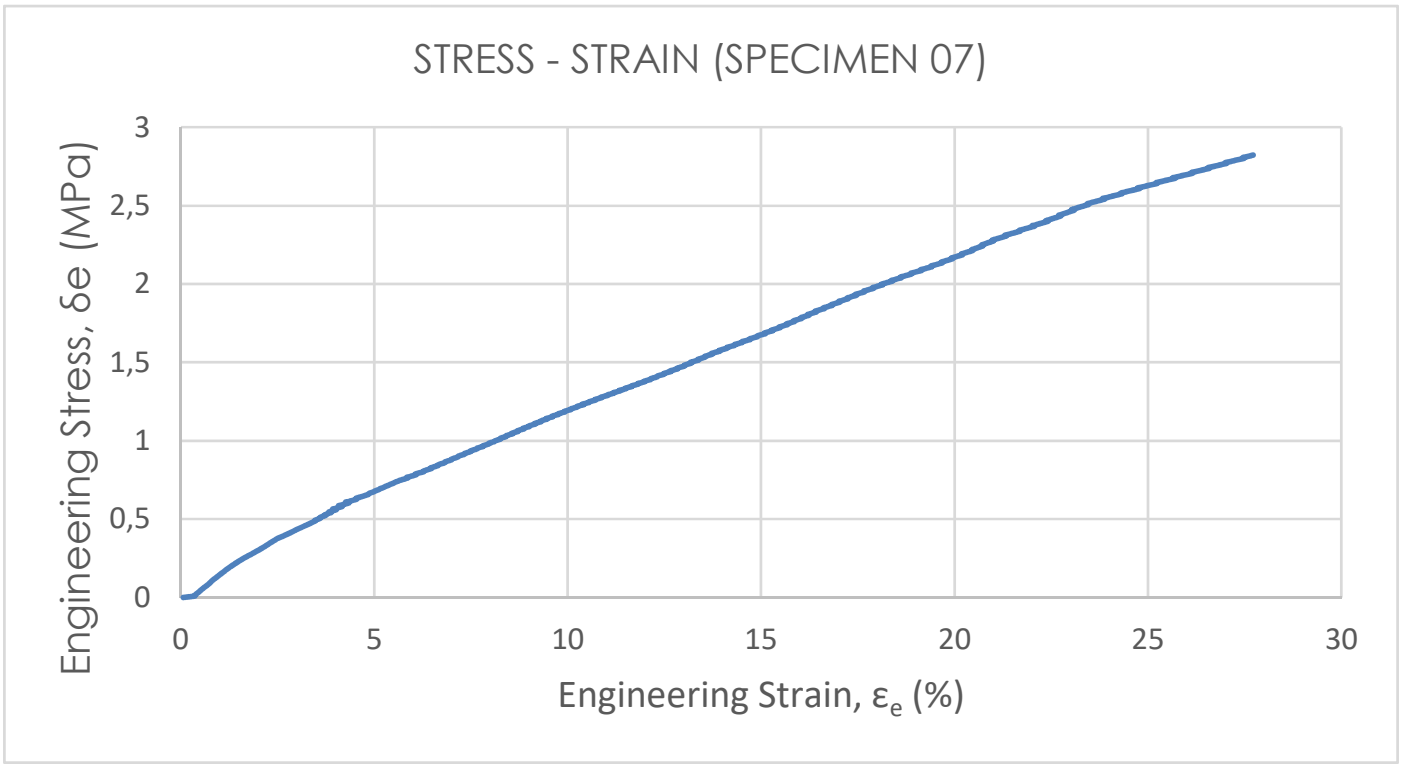
[25] J. S. Rao, Simulation Based Engineering in Solid Mechanics. .

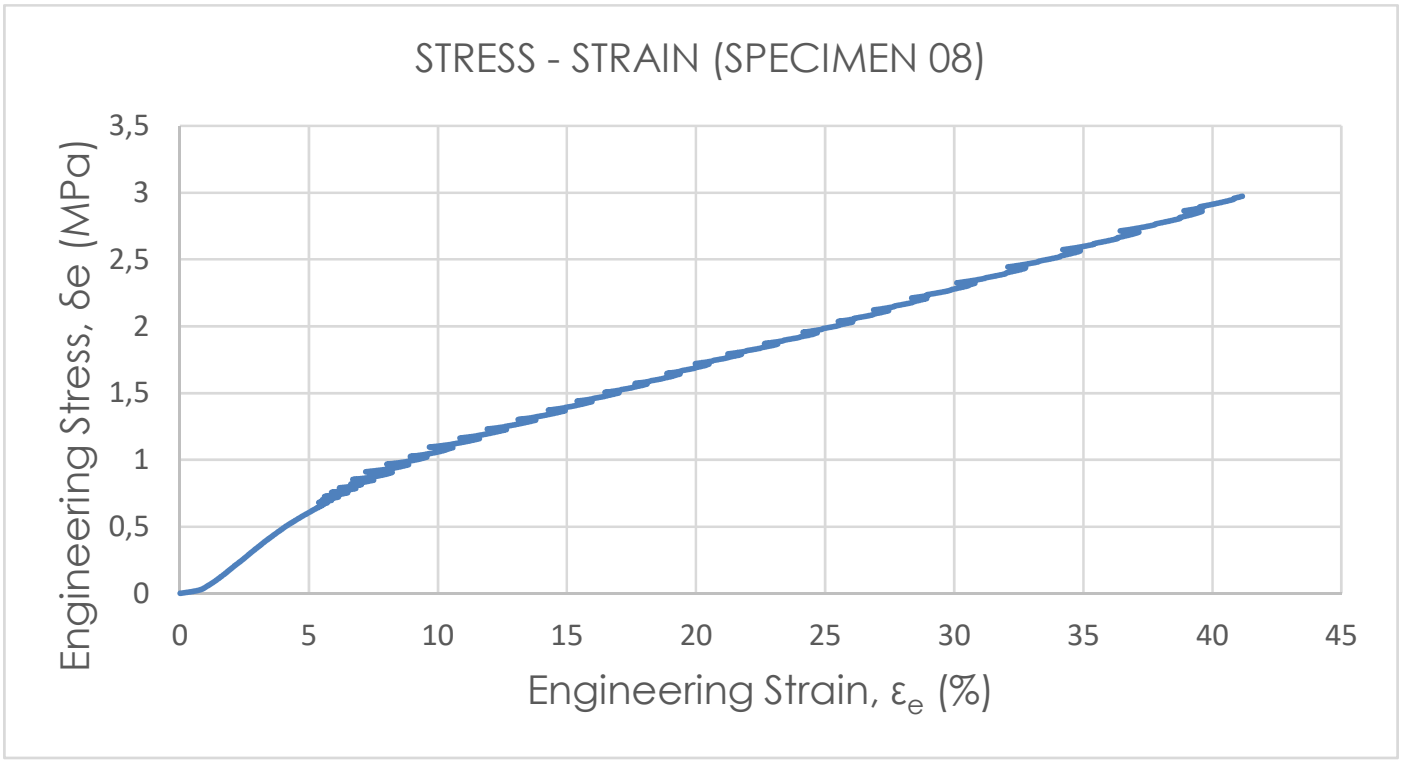
IMAGES

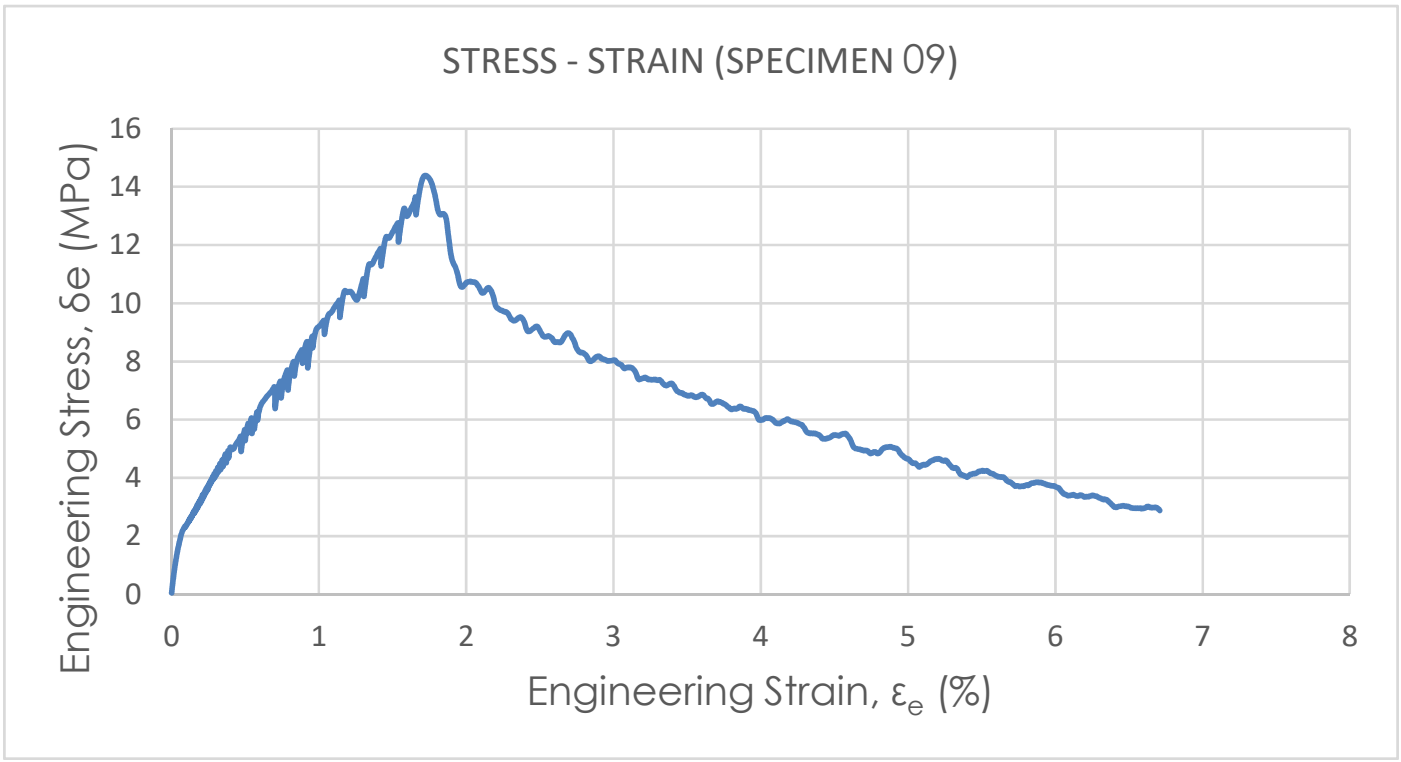
[1] G. Minke, "Working with Earthen Blocks," in Building with Earth, Birkhaeuser, 2006.
[2] E. Zakharov, Transparent glass with clean mineral water isolated on white background. 2019.
[3] Samsung CTR164NC01 57.5cm 4 Zone Electric Cooktop. .

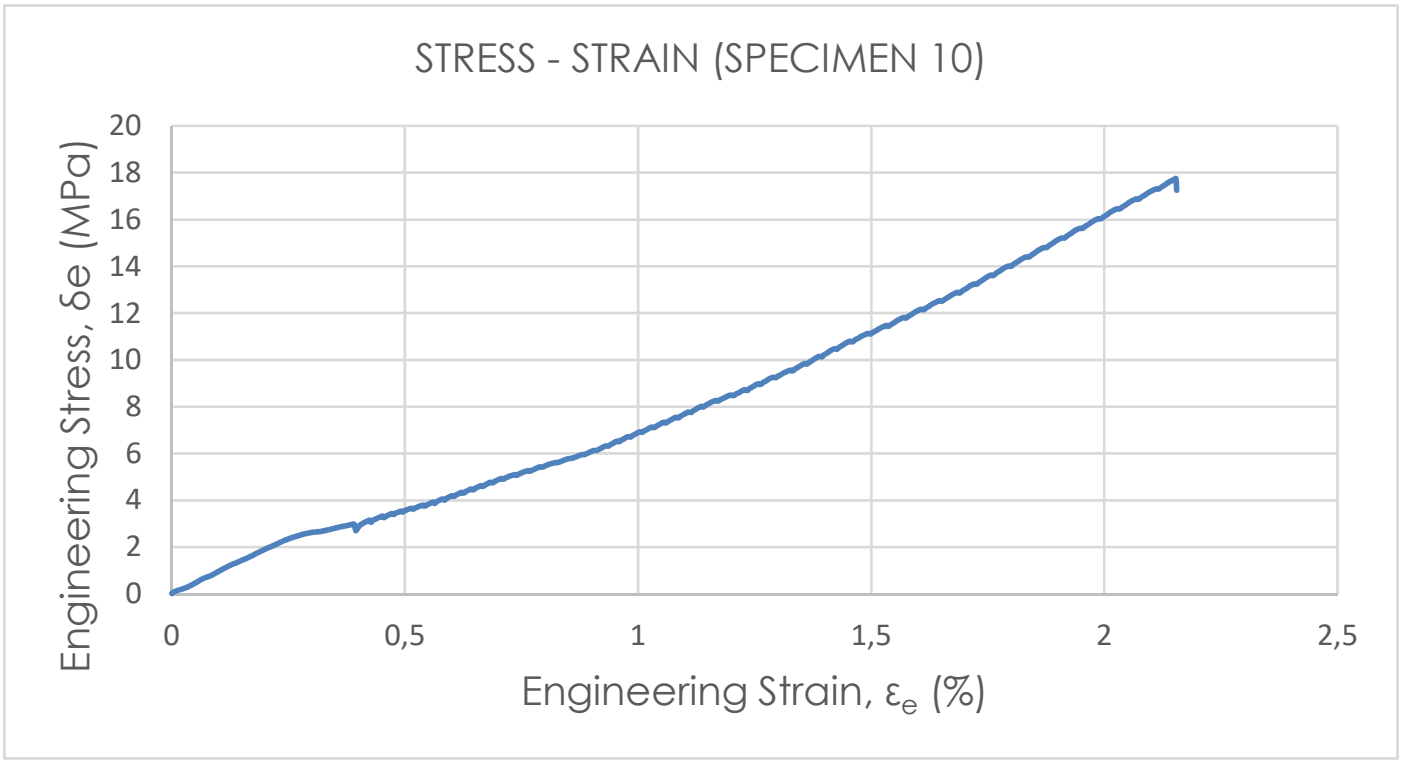
VIII. LIST OF FIGURES

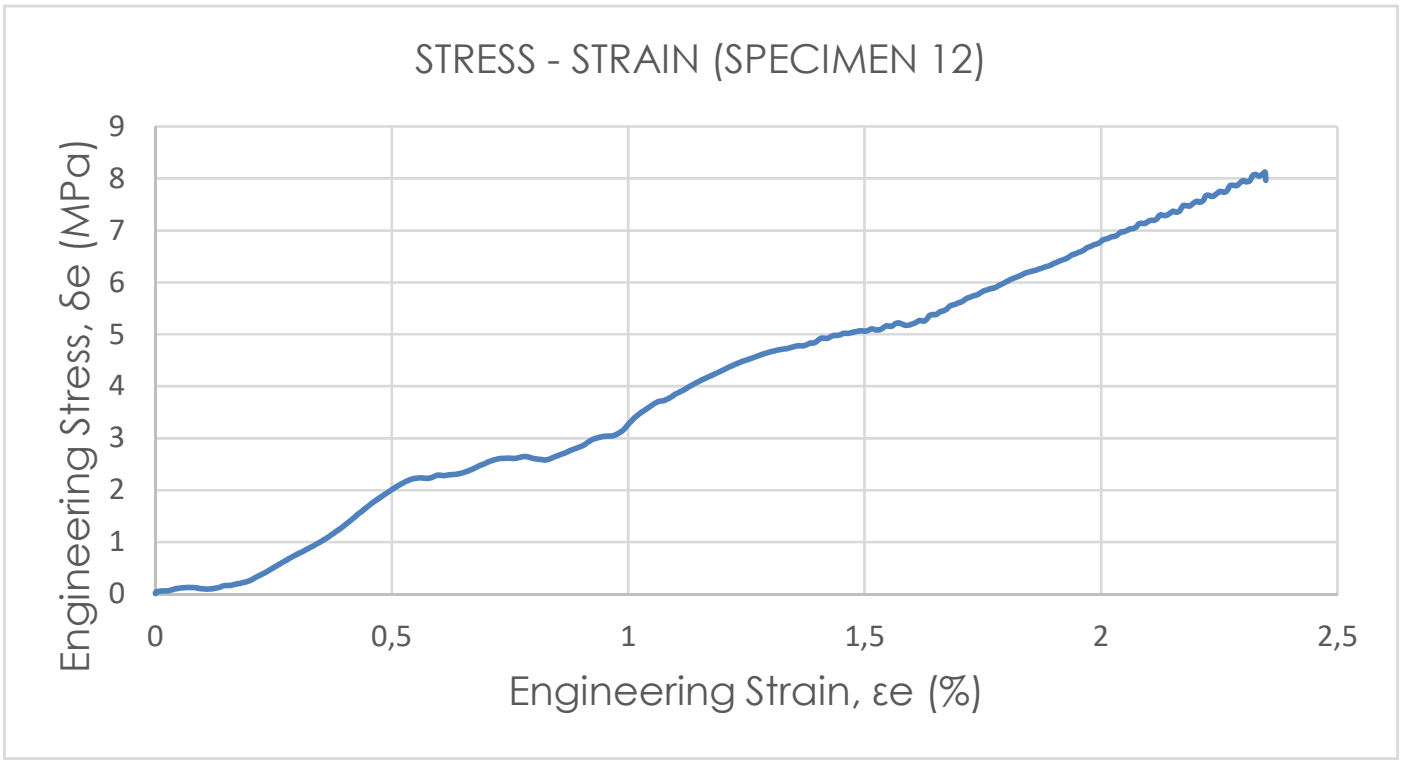
IX. APPENDIX A: STRESS-STRAIN

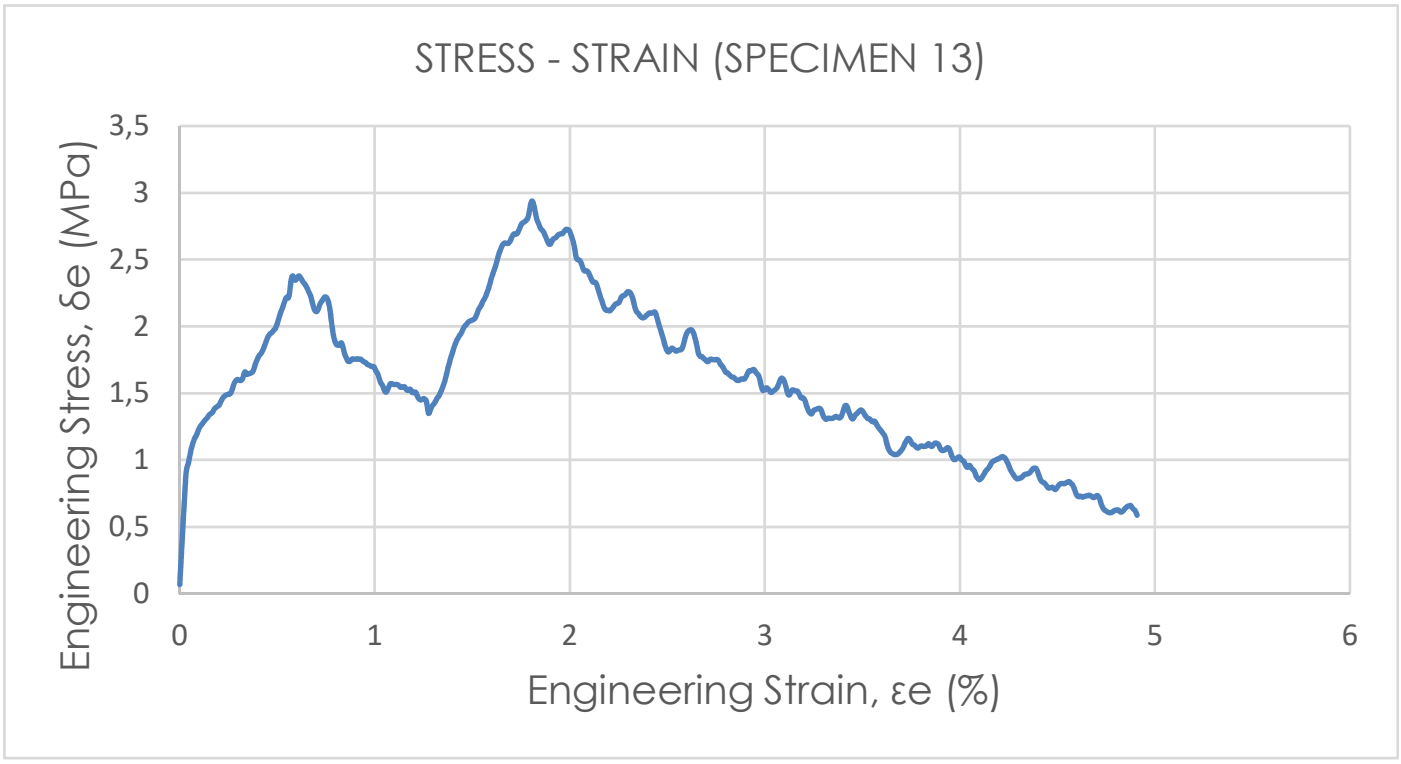


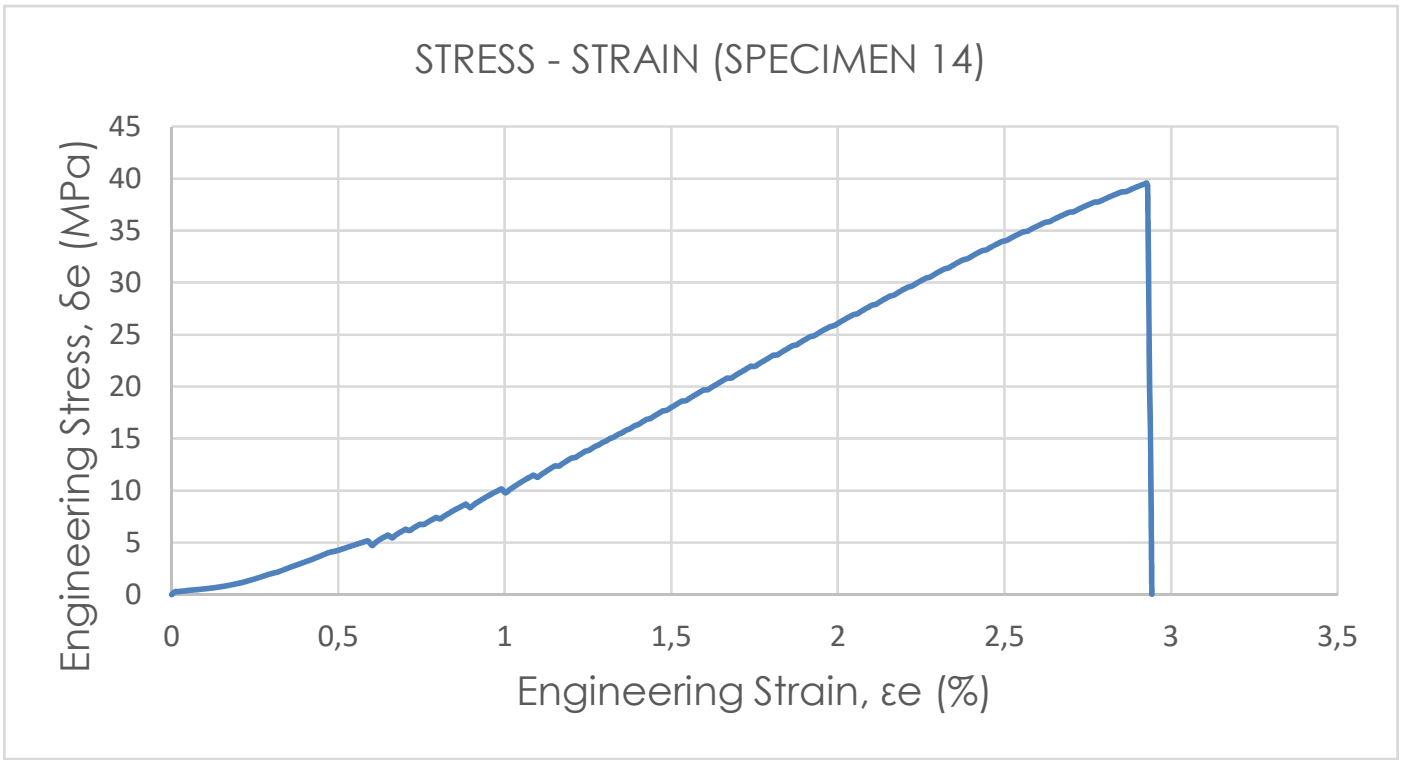


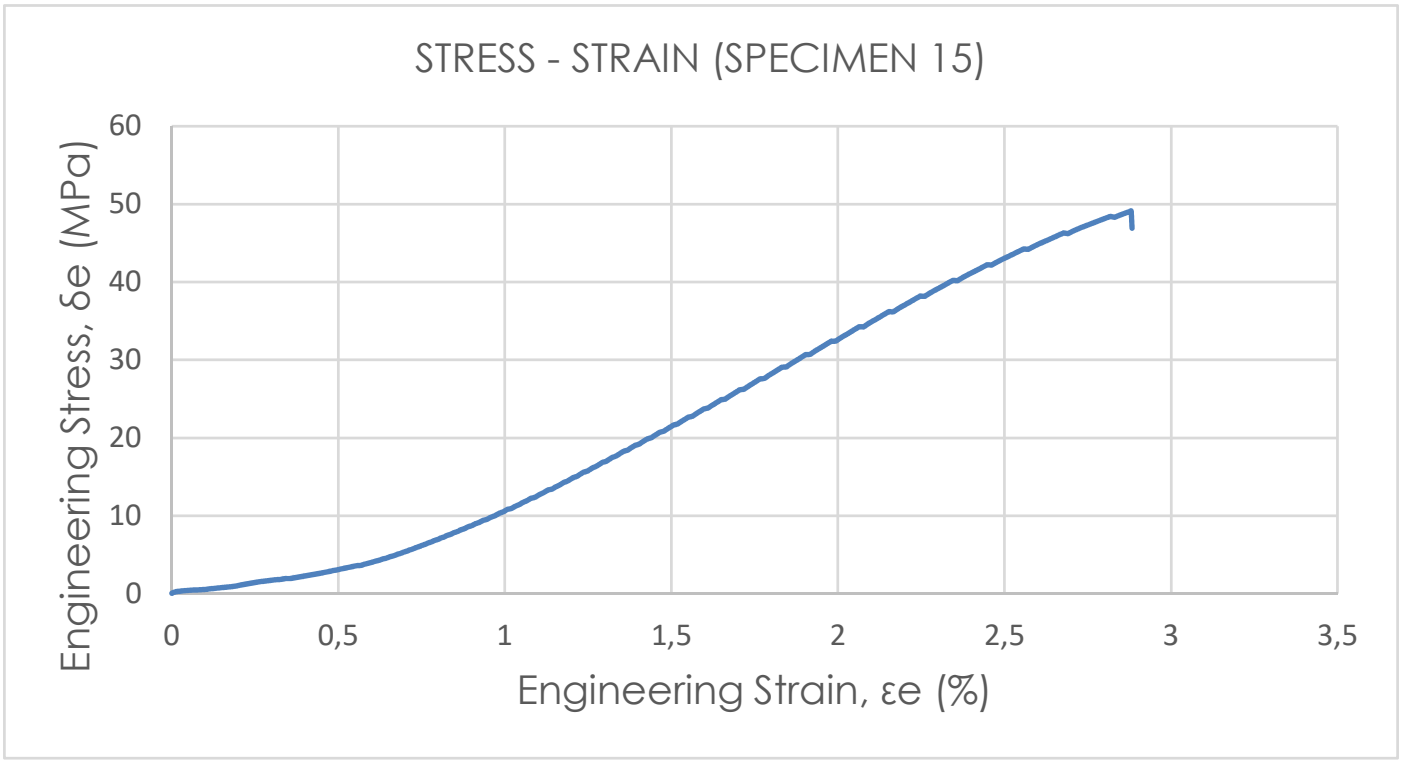

STRESS - STRAIN (SPECIMEN 09)
Engineering Stress, δe (MPa)
16
14
12
10
8
6
4
2
0
0
1
2
3
4
5
6
7
8
Engineering Strain, εe (%)

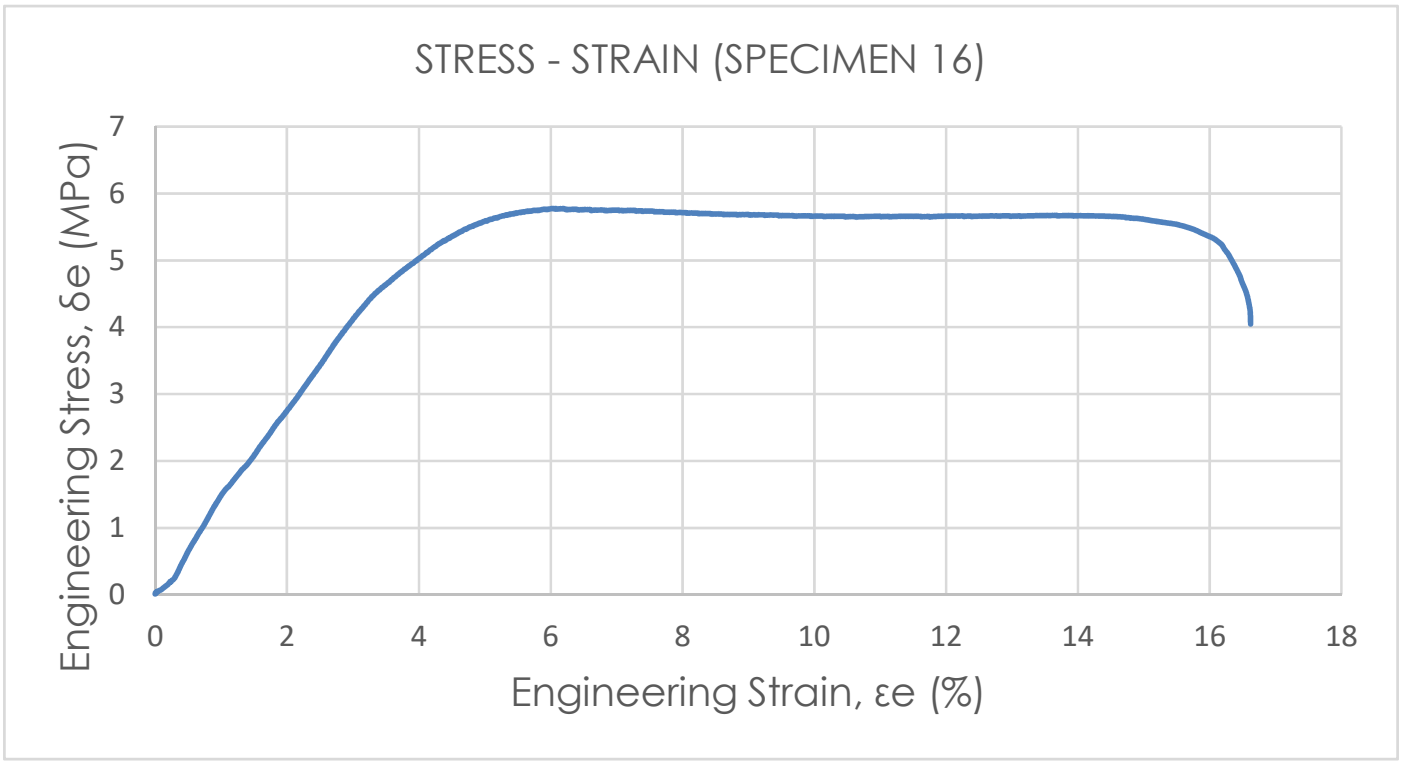

STRESS - STRAIN (SPECIMEN 10)
Engineering Stress, δe (MPa)
20
18
16
14
12
10
8
6
4
2
0
0
0,5
1
1,5
2
2,5
Engineering Strain, εe (%)

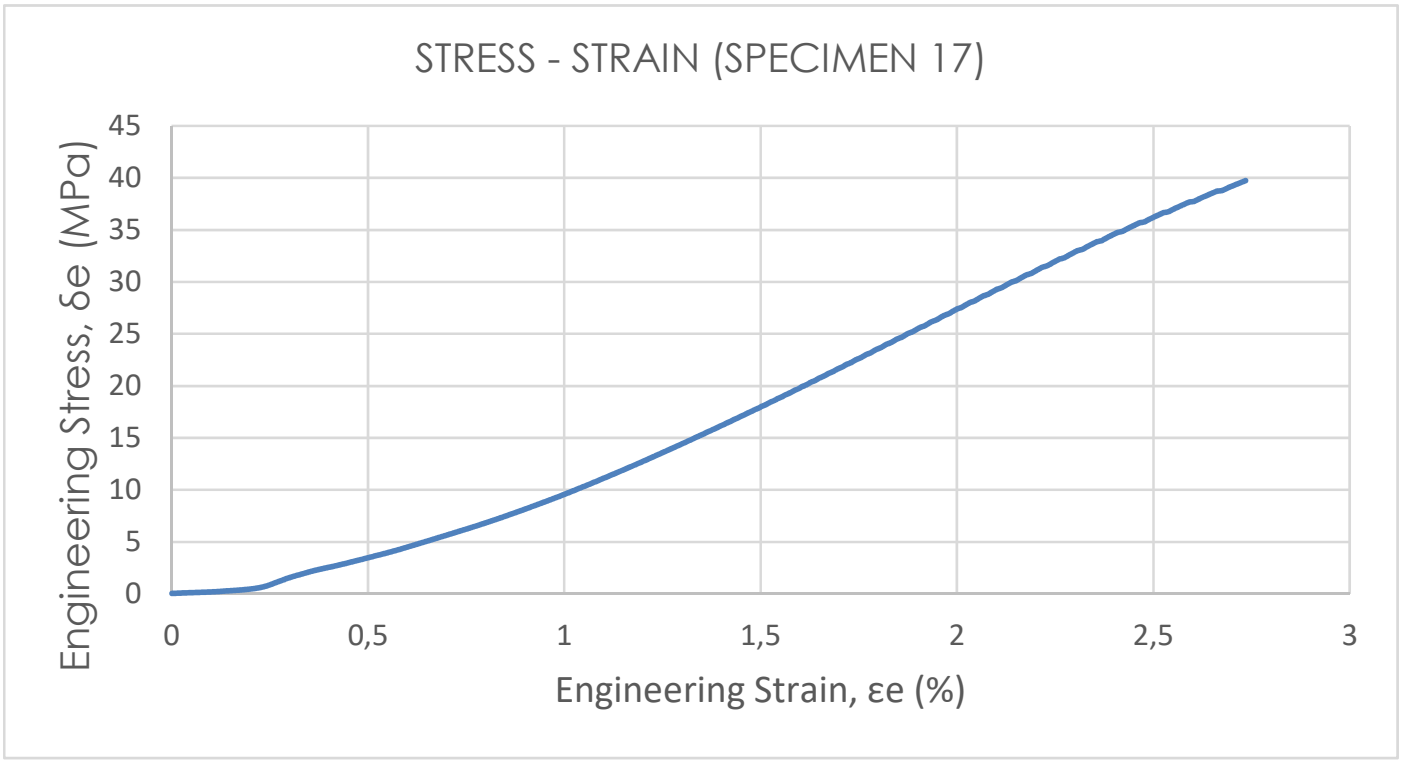

STRESS - STRAIN (SPECIMEN 12)
Engineering Stress, δe (MPa)
9
8
7
6
5
4
3
2
1
0
0
0,5
1
1,5
2
2,5
Engineering Strain, εe (%)

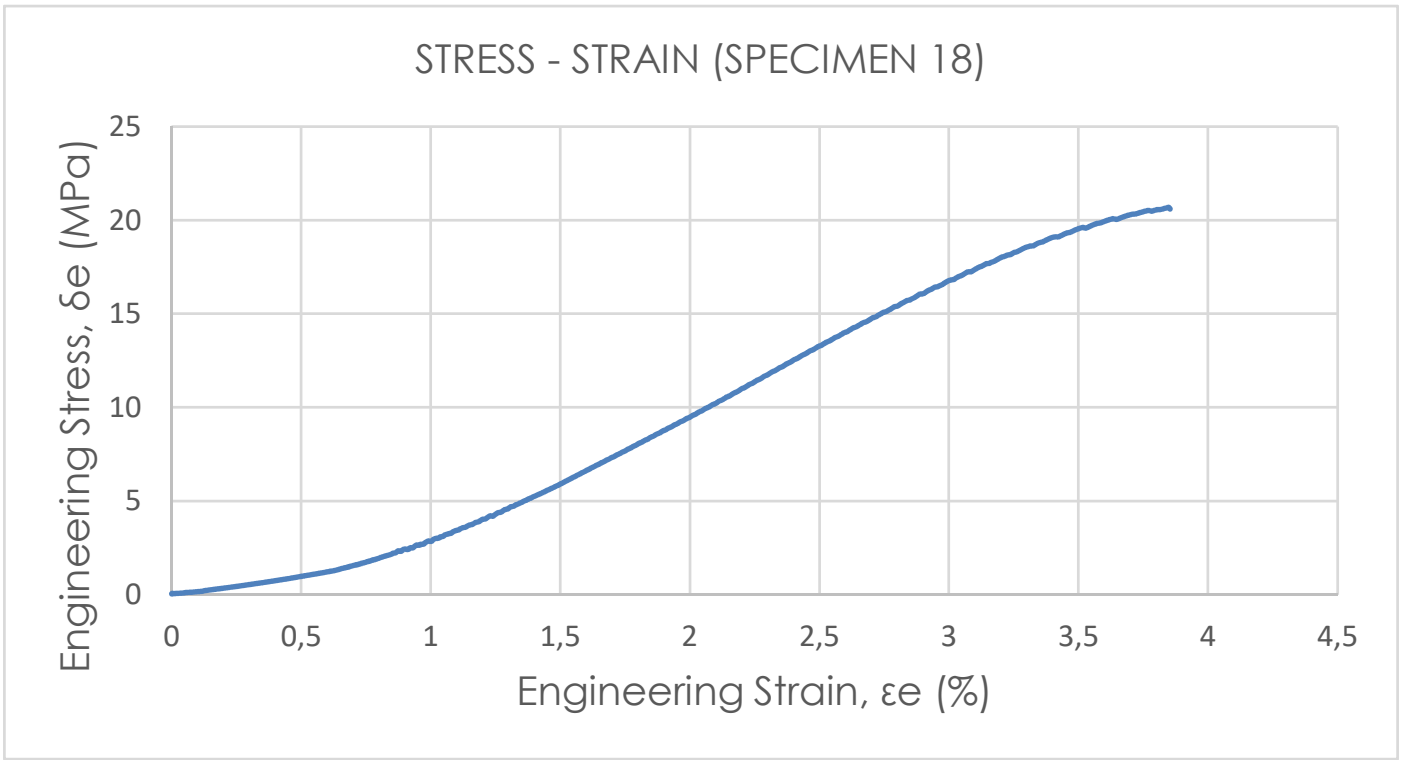

STRESS - STRAIN (SPECIMEN 13)
Engineering Stress, δe (MPa)
3,5
3
2,5
2
1,5
1
0,5
0
0
1
2
3
4
5
6
Engineering Strain, εe (%)

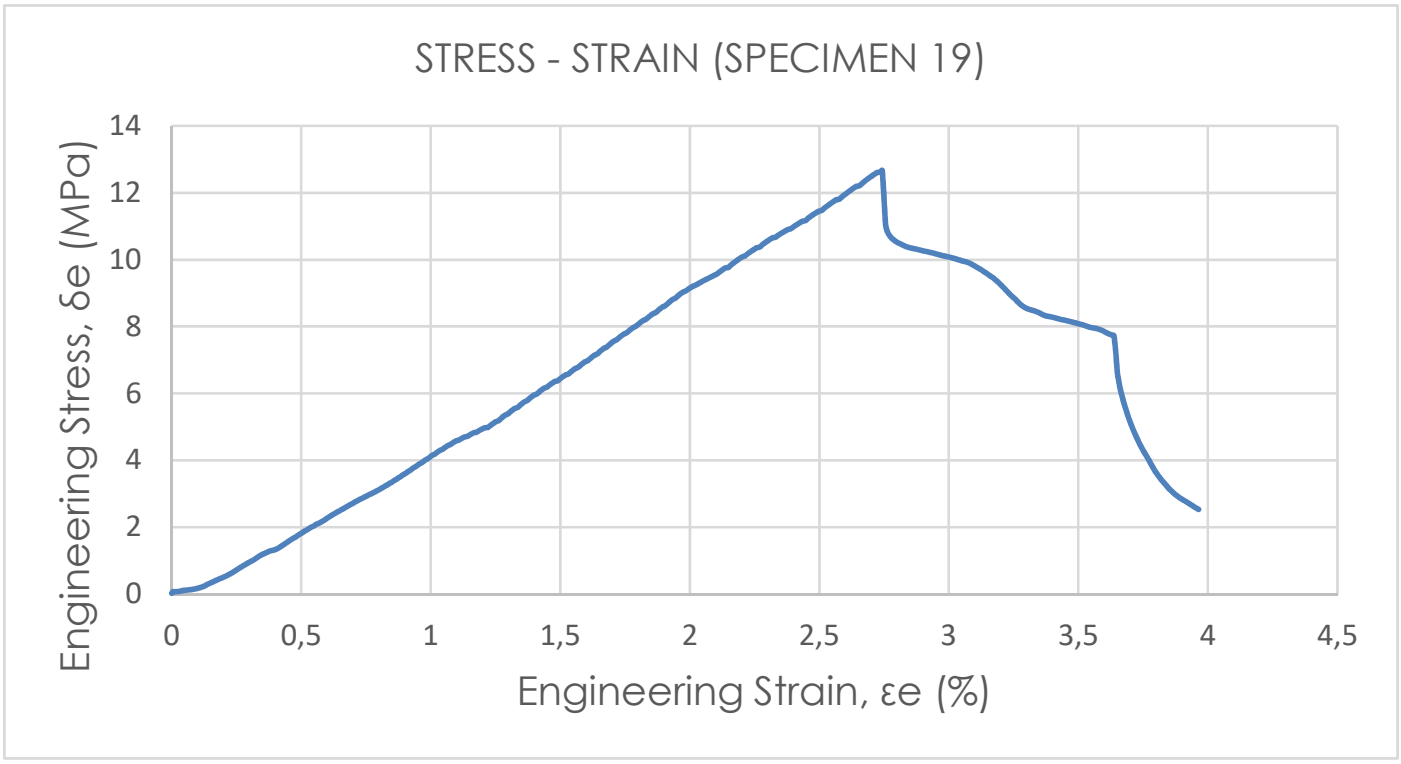

STRESS - STRAIN (SPECIMEN 14)
Engineering Stress, δe (MPa)
45
40
35
30
25
20
15
10
5
0
0
0,5
1
1,5
2
2,5
3
3,5
Engineering Strain, εe (%)

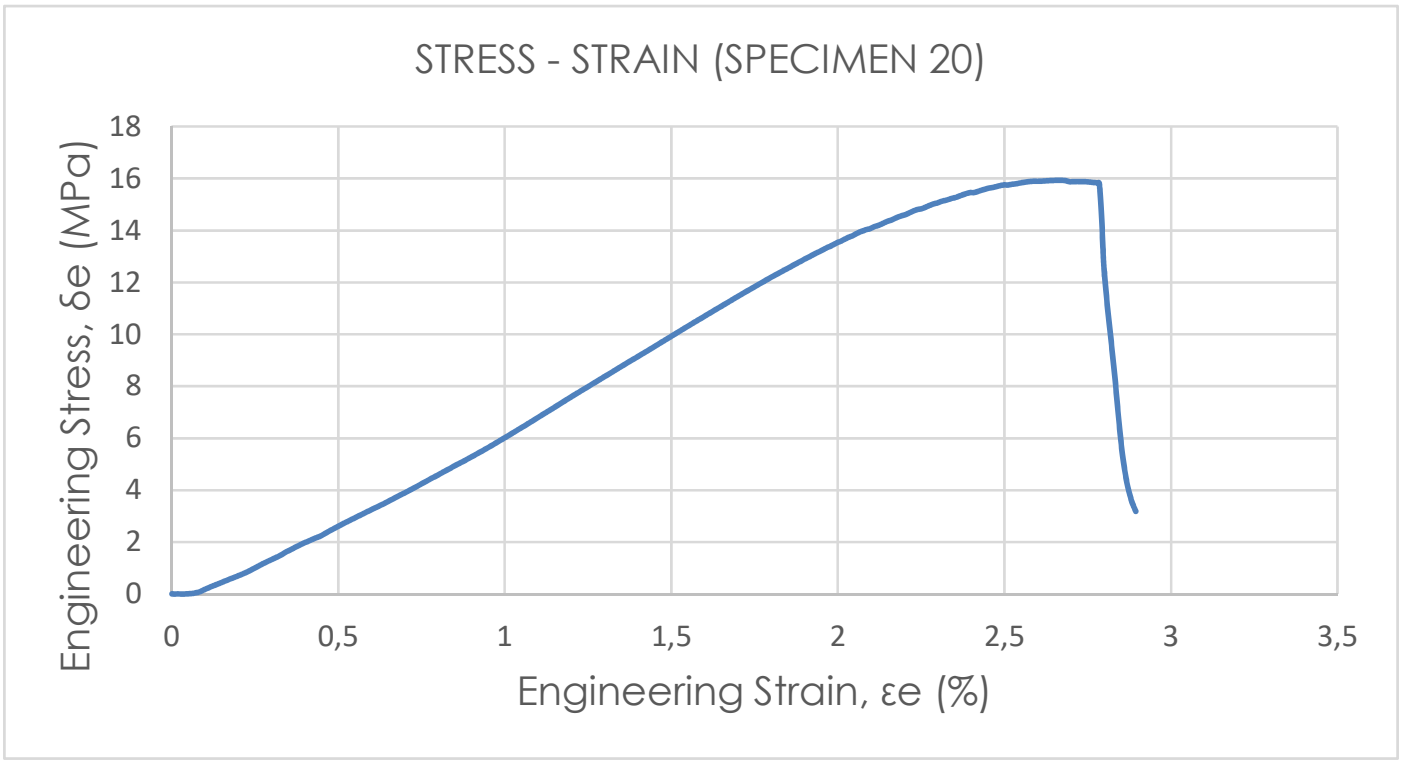

STRESS - STRAIN (SPECIMEN 15)
Engineering Stress, δe (MPa)
60
50
40
30
20
10
0
0
0,5
1
1,5
2
2,5
3
3,5
Engineering Strain, εe (%)

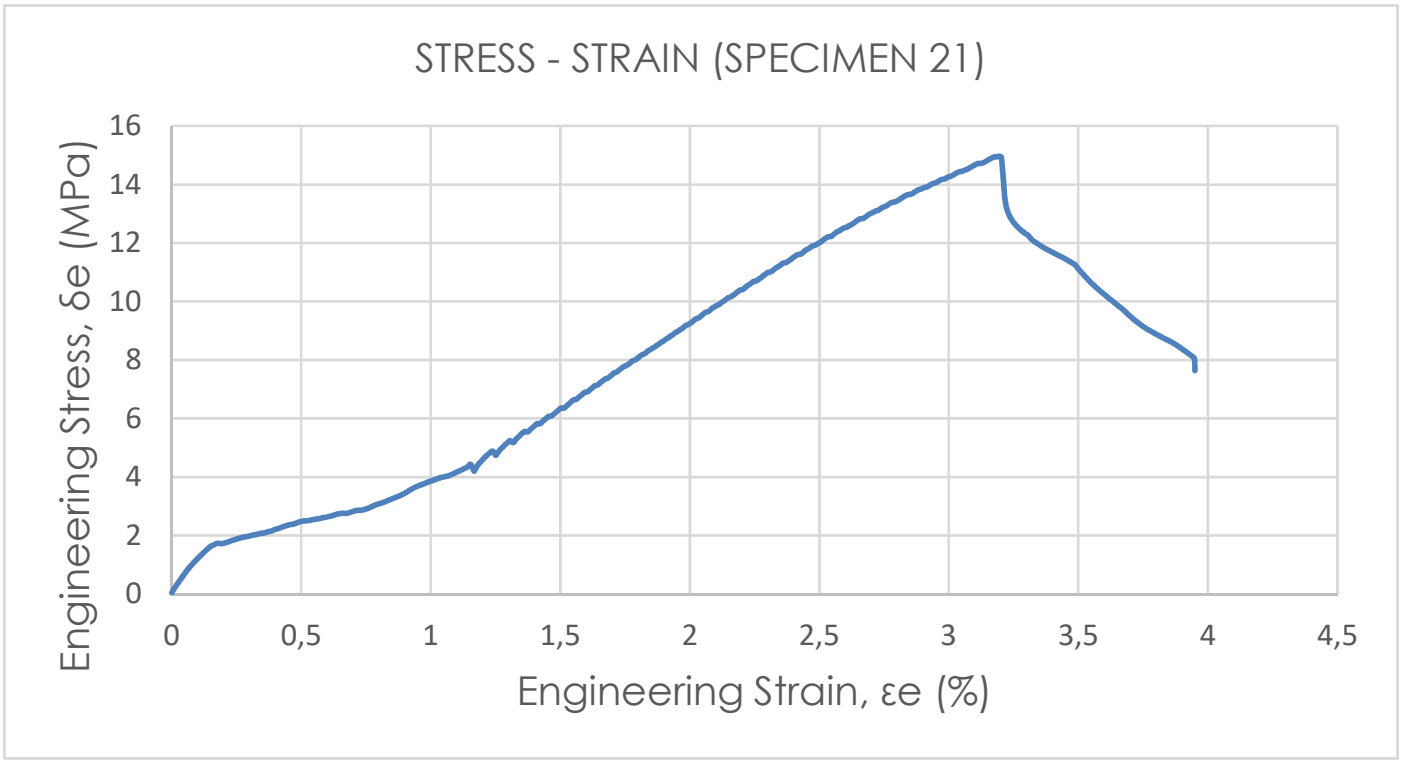

STRESS - STRAIN (SPECIMEN 16)
Engineering Stress, δe (MPa)
0
1
2
3
4
5
6
7
0
2
4
6
8
10
12
14
16
18
Engineering Strain, εe (%)

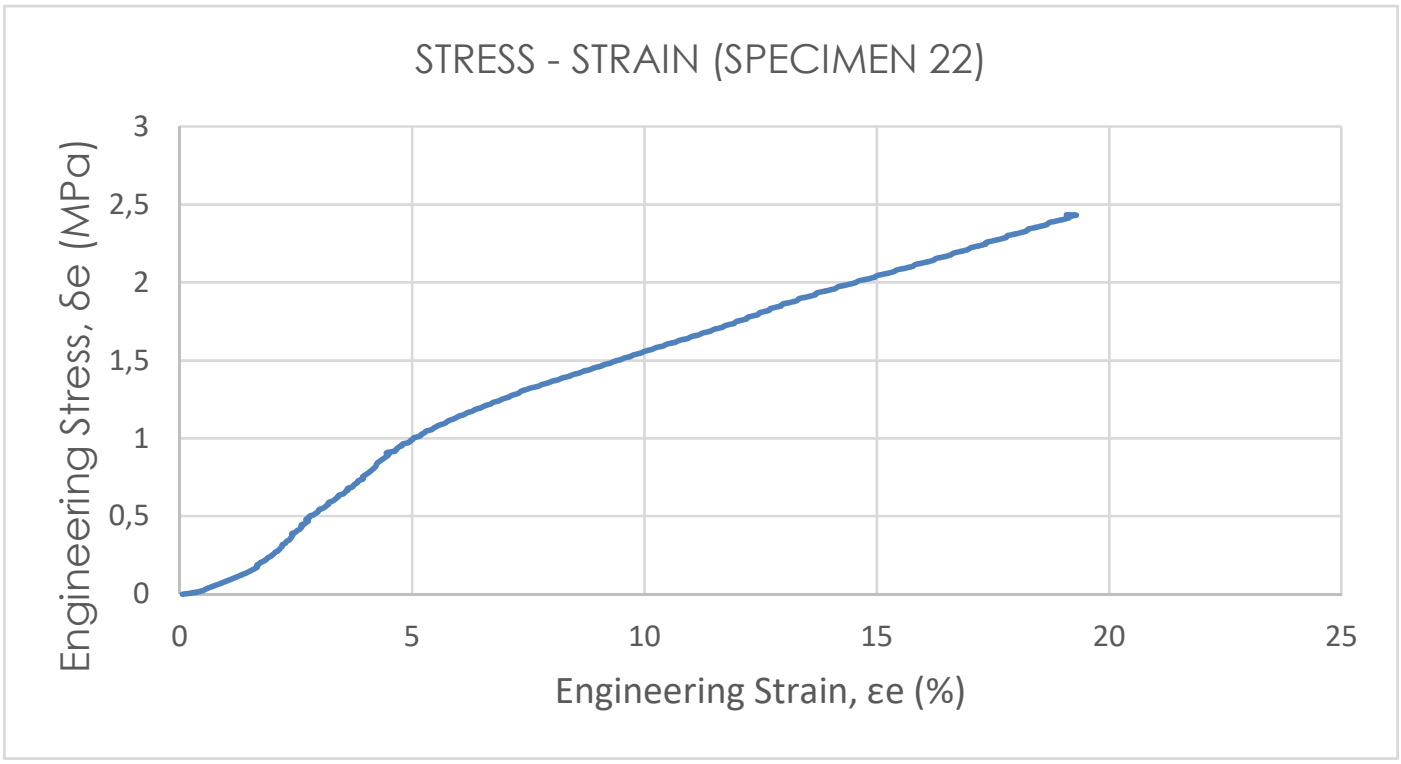

STRESS - STRAIN (SPECIMEN 17)
Engineering Stress, δe (MPa)
0
5
10
15
20
25
30
35
40
45
0
0,5
1
1,5
2
2,5
3
Engineering Strain, εe (%)

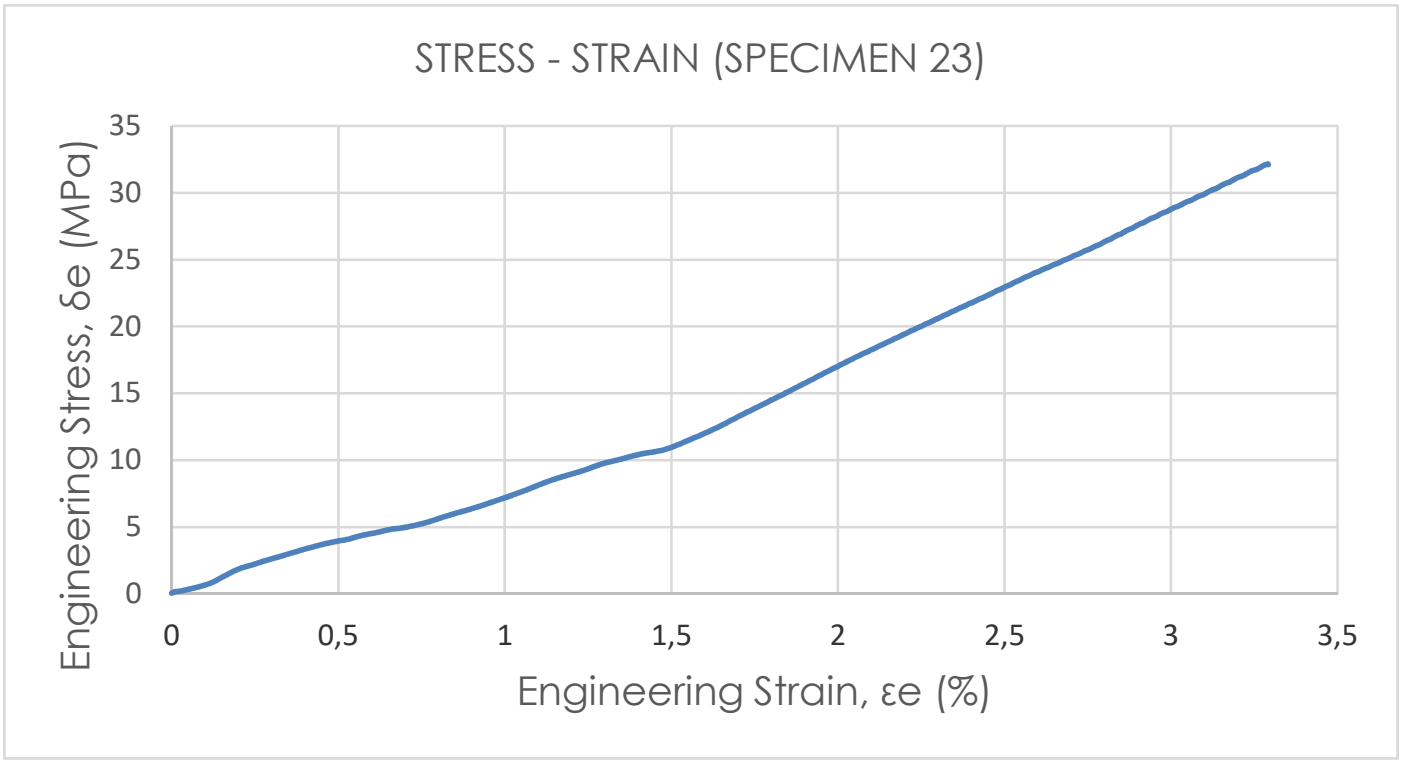

STRESS - STRAIN (SPECIMEN 18)
Engineering Stress, δe (MPa)
25
20
15
10
5
0
0
0,5
1
1,5
2
2,5
3
3,5
4
4,5
Engineering Strain, εe (%)

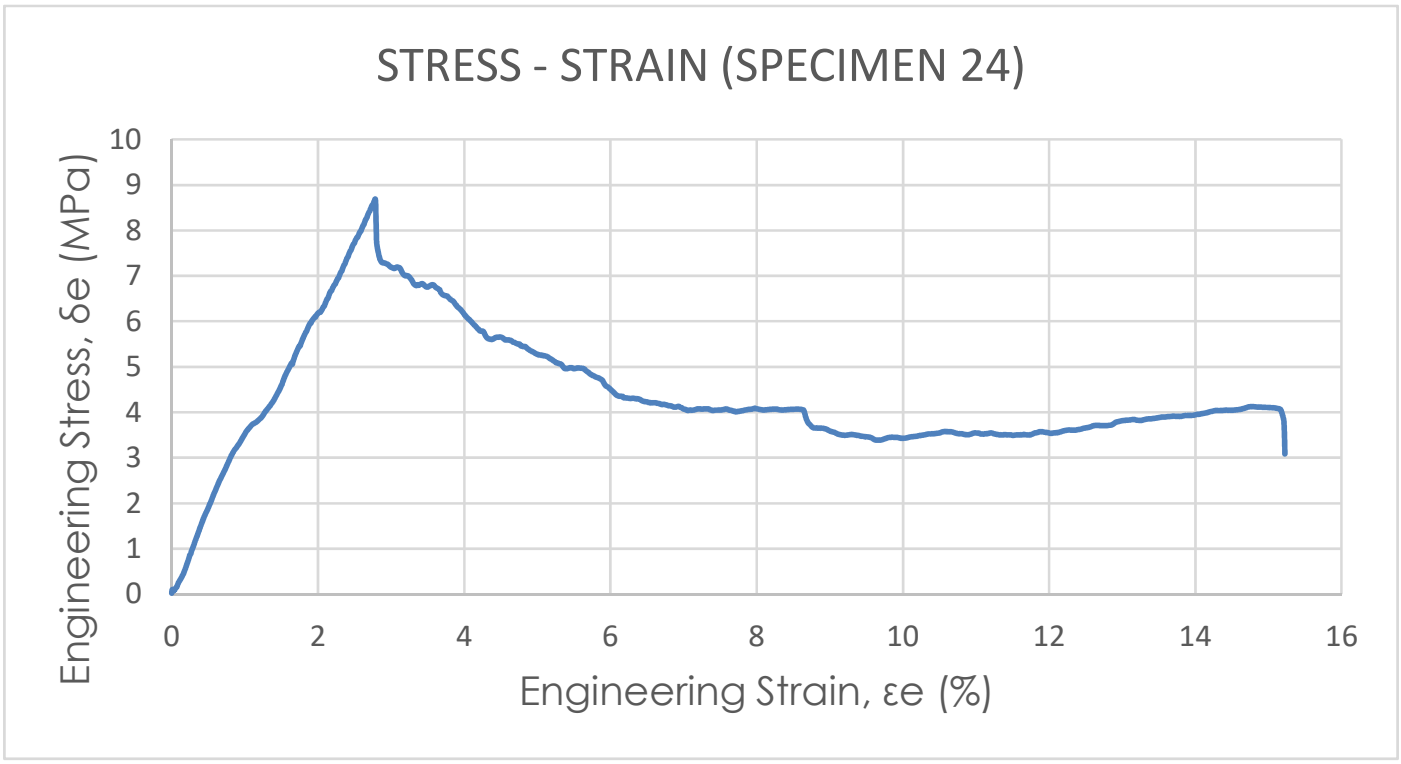

STRESS - STRAIN (SPECIMEN 19)
Engineering Stress, δe (MPa)
14
12
10
8
6
4
2
0
0
0,5
1
1,5
2
2,5
3
3,5
4
4,5
Engineering Strain, εe (%)

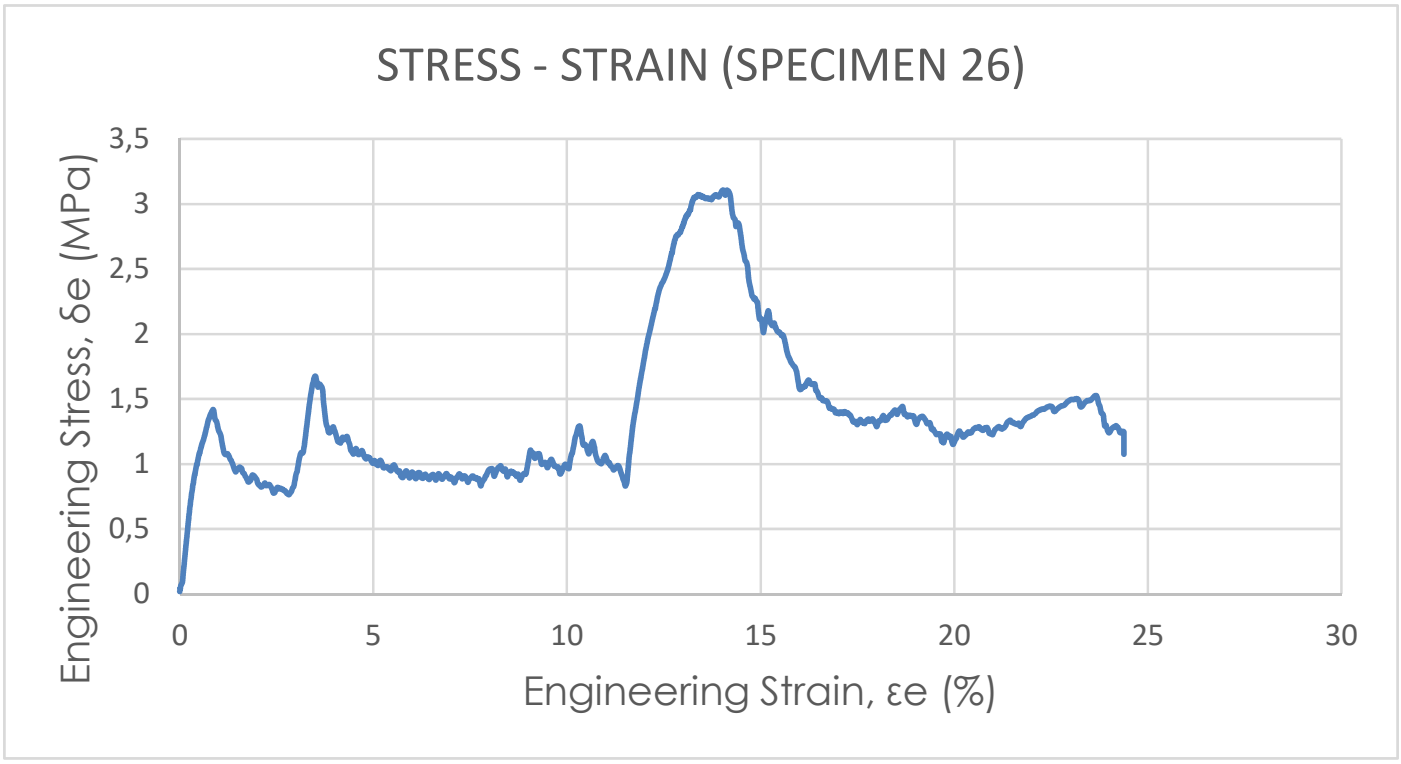

STRESS - STRAIN (SPECIMEN 20)
Engineering Stress, δe (MPa)
18
16
14
12
10
8
6
4
2
0
0
0,5
1
1,5
2
2,5
3
3,5
Engineering Strain, εe (%)

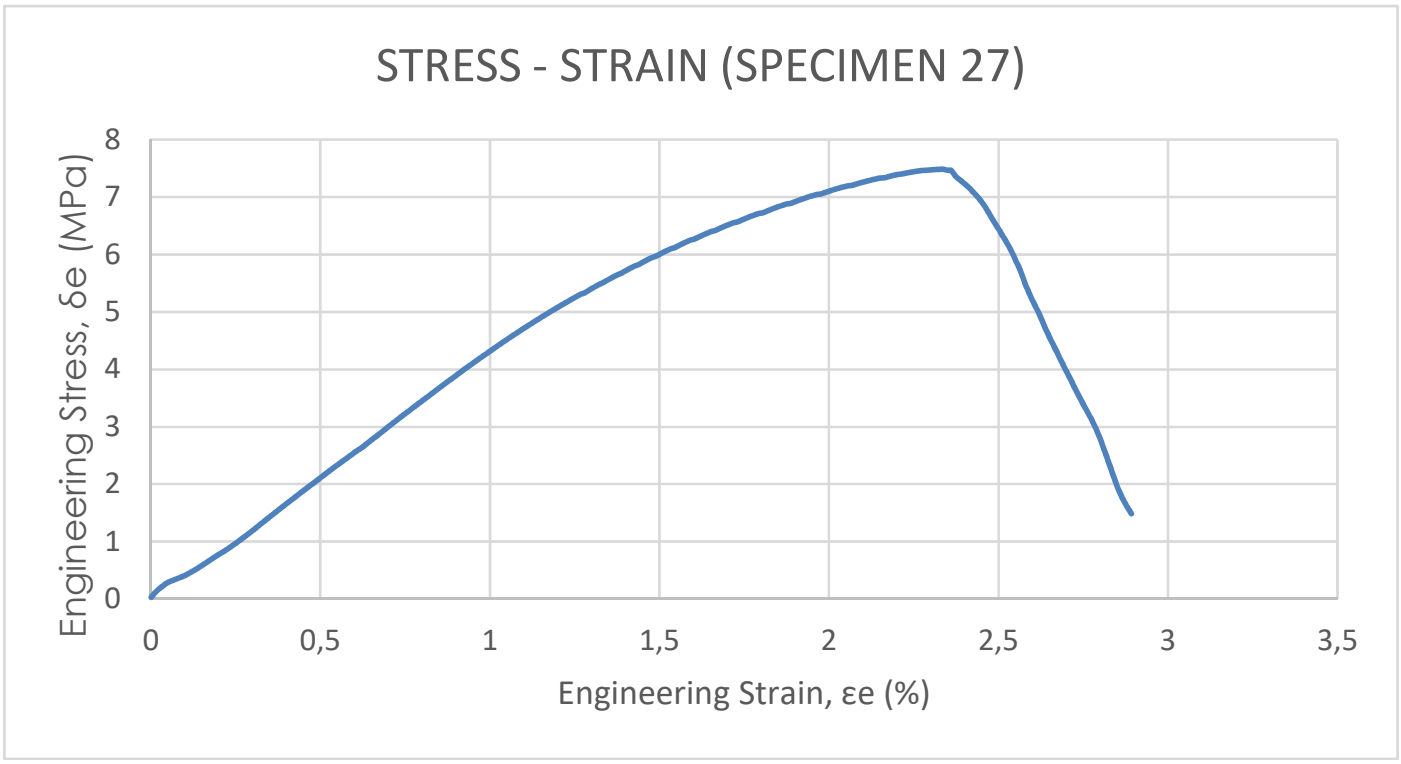

STRESS - STRAIN (SPECIMEN 21)
Engineering Stress, δe (MPa)
16
14
12
10
8
6
4
2
0
0
0,5
1
1,5
2
2,5
3
3,5
4
4,5
Engineering Strain, εe (%)

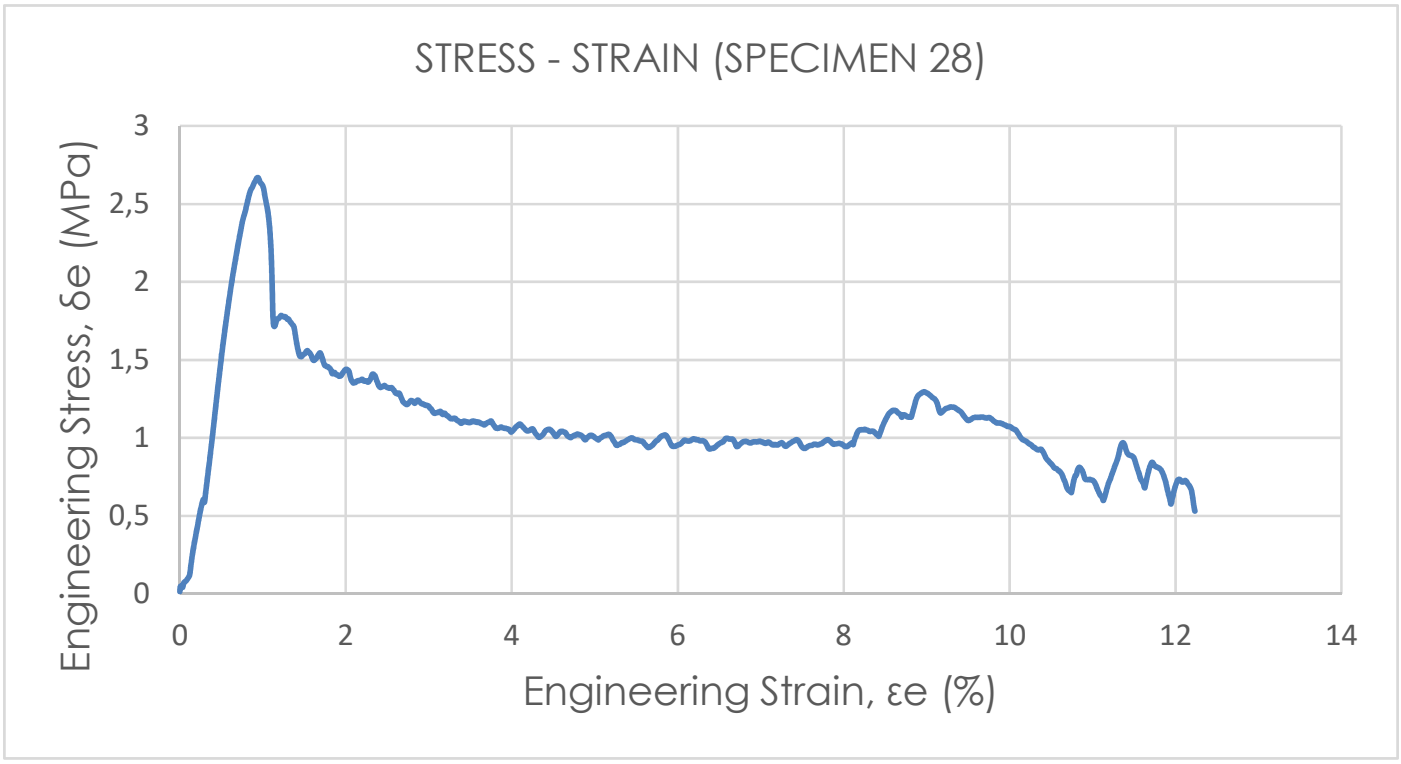

STRESS - STRAIN (SPECIMEN 22)
Engineering Stress, δe (MPa)
3
2,5
2
1,5
1
0,5
0
0
5
10
15
20
25
Engineering Strain, εe (%)

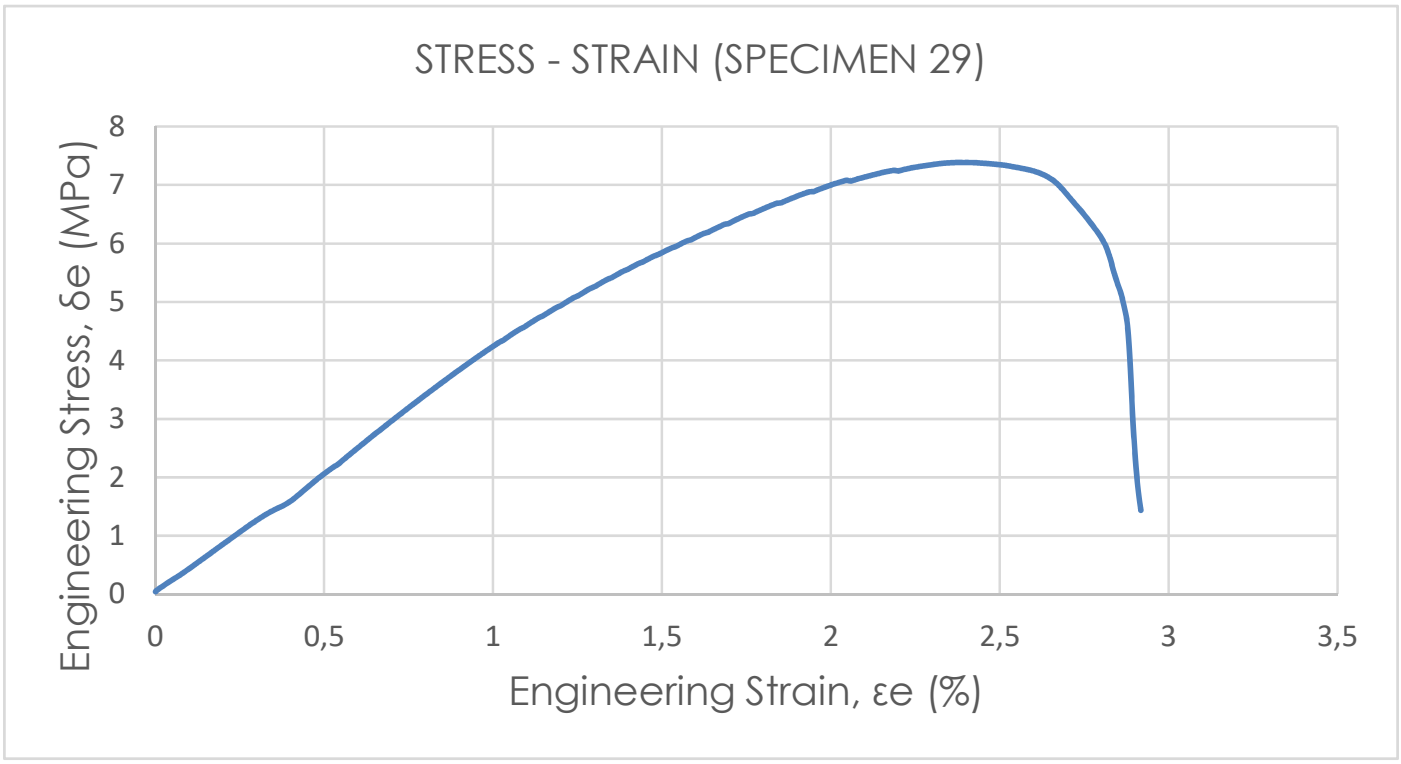

STRESS - STRAIN (SPECIMEN 23)
Engineering Stress, δe (MPa)
35
30
25
20
15
10
5
0
0
0,5
1
1,5
2
2,5
3
3,5
Engineering Strain, εe (%)


STRESS - STRAIN (SPECIMEN 24)
Engineering Stress, δe (MPa)
10
9
8
7
6
5
4
3
2
1
0
0
2
4
6
8
10
12
14
16
Engineering Strain, εe (%)


STRESS - STRAIN (SPECIMEN 26)
Engineering Stress, δe (MPa)
3,5
3
2,5
2
1,5
1
0,5
0
0
5
10
15
20
25
30
Engineering Strain, εe (%)


STRESS - STRAIN (SPECIMEN 27)
Engineering Stress, δe (MPa)
0
1
2
3
4
5
6
7
8
0
0,5
1
1,5
2
2,5
3
3,5
Engineering Strain, εe (%)


STRESS - STRAIN (SPECIMEN 28)
Engineering Stress, δe (MPa)
0
0,5
1
1,5
2
2,5
3
0
2
4
6
8
10
12
14
Engineering Strain, εe (%)


STRESS - STRAIN (SPECIMEN 29)
Engineering Stress, δe (MPa)
0
1
2
3
4
5
6
7
8
0
0,5
1
1,5
2
2,5
3
3,5
Engineering Strain, εe (%)

XI. APPENDIX B: FIRST APPROACH CODE
MESH CODE

```
###CREATE MESH###
import rhinoscriptsyntax as rs
import Rhino.Geometry as rg
import math

#create class for vertex points
class Vert:
    def __init__(self):
        self.kx = None
        self.ky = None
        self.k = None
        self.p = None
    def setKey(self,x,y):
        self.kx = x
        self.ky = y
        self.k = (self.kx, self.ky)
    def setP(self,a):
        self.p = a
    def getP(self):
        return self.p
    def getKey(self):
        return self.k
    def getRevKey(self):
        return (self.ky,self.kx)
    def getKX(self):
        return self.kx
    def getKY(self):
        return self.ky

# get brep edges
brepe = brep.Curves3D

#pick the longest edge
lenl = []
for i in brepe:
    l = rg.Curve.GetLength(i)
    lenl.append(l)
lenl.sort()
long = lenl[len(lenl)-1]
for i in brepe:
    if i.GetLength() == long:
        lngst = i

#find the support points
a = brep.Edges
supportp = []
for i in range(a.Count):
    x = a.Item[i]
    y = x.EndVertex
    z = y.Location
    supportp.append(z)

#create plane from support points
mpl = rg.Plane(supportp[0],supportp[1],sup-
portp[2])

#create brick locations
lenlngst = rg.Curve.GetLength(lngst)
nob = math.floor(lenlngst/(refine*bw))
eqdiv = lenlngst/nob
plocs = rg.Curve.DivideEquidistant(lngst, eqdiv)

#project points onto plane
ptproject = []
for i in plocs:
    a = rg.Point3d(i.X, i.Y, mpl.OriginZ)
    ptproject.append(a)

#create contours for bricks
bricklines = []
for i in ptproject:
    a = rg.Line(i, 2*mpl.YAxis)
    bricklines.append(a)

#project lines to brep
procrvs = []
for i in bricklines:
    icrv = rg.Line.ToNurbsCurve(i)
    procrv = rg.Curve.ProjectToBrep(icrv,brep,rg.
Vector3d(0,0,-1),0.01)
    if len(procrv) ==1:
        procrvs.append(procrv[0])

#create pt grid
plocs2 = []
verts = []
listnob = []
sup = []
for x,i in enumerate(procrvs):
    leni = rg.Curve.GetLength(i)
    nob2 = math.floor(leni/(refine*bh))
    eqdiv2= leni/nob2
    ploc2 = rg.Curve.DivideEquidistant(i,eqdiv2)
    listnob.append(nob2)
    for y,j in enumerate(ploc2):
        plocs2.append(j)
        vert = Vert()
        vert.setKey(x,len(ploc2)-1-y)
        vert.setP(j)
        verts.append(vert)
        if x == 0 and y == len(ploc2)-1:
            sup.append(vert.getP())
```

```
        if x == len(procrvs)-1 and y== len(ploc2)-1:
            sup.append(vert.getP())
        if x == math.floor(nob/2-1) and y == 0:
            sup.append(vert.getP())
        if x == math.ceil(nob/2-1) and y == 0:
            sup.append(vert.getP())

#create mesh
mesh = rg.Mesh()

for i in verts:
    x = i.getKX()
    y = i.getKY()
    a = i.getP()
    av = mesh.Vertices.Add(a)
    for j in verts:
        if j.getKX() == x+1 and j.getKY() == y:
            b = j.getP()
            bv = mesh.Vertices.Add(b)
            for k in verts:
                if k.getKX() == x+1 and k.getKY() ==y+1:
                    c = k.getP()
                    cv = mesh.Vertices.Add(c)
                    for m in verts:
                        if m.getKX() == x and m.getKY() == y+1:
                            d = m.getP()
                            dv = mesh.Vertices.Add(d)
                            face = mesh.Faces.AddFace(av,bv,cv,dv)
```

XII. APPENDIX C: SECOND APPROACH CODES
01 VOXELIZATION

```
import rhinoscriptsyntax as rs
import Rhino.Geometry as rg
import math
import Rhino
import Grasshopper

bbox = rg.Brep.GetBoundingBox(brep, rg.Plane.WorldXY)
mesh = rg.Mesh()

#find the right index
def findIndex(x,y,z):
    result = int(x*(yr)*(zr) + y*(zr) + z)
    return result

#Find the Length, Widhth, and Height of the bounding box of the Brep
w = bbox.Diagonal.X
l = bbox.Diagonal.Y
h = bbox.Diagonal.Z

#Divide the 3 axis of the boudning box by the dimensions of the voxel
#This will give you a U, V, W value along the bounding box
#Turn this value into an integer by finding the next biggest integer (roudingUp)
xr = int(math.ceil (w/voxres))
yr = int(math.ceil (l/voxres))
zr = int(math.ceil (h/voxres))

#create a container for your point list and distance list
points = []
distList = []

# create a base plane for all the geometry
bp = bbox.Min
bXV = rg.Vector3d.XAxis
bYV = rg.Vector3d.YAxis

bPlane = rg.Plane(bp, bXV, bYV)

# Due to the fact that all voxels are based on the center points, we need
# to create a shift towards the voxel's center, which is half of each dimension
xShift = (w/xr)/2
yShift = (l/yr)/2
zShift = (h/zr)/2

#create voxels
voxs = [] #container for vox class
voxsbox = [] #container for instance.vox attribute.
for i in range (0,xr):
    for j in range(0,yr):
        for k in range(0,zr):
            vox = Vox()
            x = i*(w/xr)
```

```
            y = j*(l/yr)
            z = k*(h/zr)
            vox.setVox(rg.BoundingBox(x+bbox.Min.X,y+bbox.Min.Y,z+bbox.Min.Z,x+(w/xr)+bbox.
Min.X,y+(l/yr)+bbox.Min.Y,z+(h/zr)+bbox.Min.Z))
            vox.setIndex(i,j,k)
            vox.setCenter(vox.getVox().Center)
            voxs.append(vox)
            voxsbox.append(vox.getVox())

#pick voxels in the support area
supports = []

for i in support:
    for j in voxs:
        a = rg.Brep.IsPointInside(i,j.getCenter(),0.001, False)
        if a == True:
            j.setSupport(True)
            j.setActive(True)
            supports.append(j)
        else:
            if j.getSupport() != True:
                j.setSupport(False)

#pick voxels in the clear area
for i in voxs:
    a = rg.Brep.IsPointInside(remove,i.getCenter(),0.001,False)
    if a == True:
        i.setActive(False)

voxsactive =[]
for i in voxs:
    if i.getActive() != False:
        i.setActive(True)
        voxsactive.append(i.getVox())
```

XII. APPENDIX C: SECOND APPROACH CODES
02 Lines For Beams&Points For Support

```
import rhinoscriptsyntax as rs
import Rhino.Geometry as rg
import math
import Rhino
import Grasshopper
from VoxClass import Vox

for i in voxs:
    print i.getActive()

#find the right index
def findIndex(x,y,z):
    result = int(x*(yr)*(zr) + y*(zr) + z)
    return result

#find the adjacencies
for a,i in enumerate(voxs):
    x = i.getX()
    y = i.getY()
    z = i.getZ()

    if x == 0:
        i.setAdjacency(0,False)
    if x+1 < xr:
        next = findIndex(x+1,y,z)
        if voxs[next].getActive() == False:
            i.setAdjacency(2,False)
            if i.getActive() == False:
                voxs[next].setAdjacency(0,False)
            else:
                voxs[next].setAdjacency(0,True)
        else:
            i.setAdjacency(2,True)
            if i.getActive() == False:
                voxs[next].setAdjacency(0,False)
            else:
                voxs[next].setAdjacency(0,True)
    if x == xr-1:
        i.setAdjacency(2,False)

    if y == 0:
        i.setAdjacency(1,False)
    if y+1 < yr:
        next = findIndex(x,y+1,z)
        if voxs[next].getActive() == False:
            i.setAdjacency(3,False)
            if i.getActive() == False:
                voxs[next].setAdjacency(1,False)
            else:
                voxs[next].setAdjacency(1,True)
        else:
```

```
            i.setAdjacency(3,True)
            if i.getActive() == False:
                voxs[next].setAdjacency(1,False)
            else:
                voxs[next].setAdjacency(1,True)
    if y == yr-1:
        i.setAdjacency(3,False)

    if z == 0:
        i.setAdjacency(5,False)
    if z+1 < zr:
        next = findIndex(x,y,z+1)
        if voxs[next].getActive() == False:
            i.setAdjacency(4,False)
            if i.getActive() == False:
                voxs[next].setAdjacency(5,False)
            else:
                voxs[next].setAdjacency(5,True)
        else:
            i.setAdjacency(4,True)
            if i.getActive() == False:
                voxs[next].setAdjacency(5,False)
            else:
                voxs[next].setAdjacency(5,True)
    if z == zr-1:
        i.setAdjacency(4,False)

#create lines for beams
lines = []
linesInds = []
linesPts = []
for x,i in enumerate(voxs):
    if i.getActive() == True:
        x = i.getX()
        y = i.getY()
        z = i.getZ()
        if x<xr-1 and y<yr-1 and z<zr-1:
            next1 = findIndex(x+1,y,z)
            next2 = findIndex(x,y+1,z)
            next3 = findIndex(x,y,z+1)
            if voxs[next1].getActive() !=False:
                line = rg.Line(i.getCenter(),voxs[next1].getCenter())
                lines.append(line)
                linesInds.append((i.getIndex(), voxs[next1].getIndex()))
            else:
                pass
            if voxs[next2].getActive() !=False:
                line = rg.Line(i.getCenter(),voxs[next2].getCenter())
                lines.append(line)
                linesInds.append((i.getIndex(), voxs[next2].getIndex()))
            else:
```

XII. APPENDIX C: SECOND APPROACH CODES
02 Lines For Beams&Points For Support

```
            pass
        if voxs[next3].getActive() !=False:
            line = rg.Line(i.getCenter(),voxs[next3].getCenter())
            lines.append(line)
            linesInds.append((i.getIndex(), voxs[next3].getIndex()))
        else:
            pass
    if x == xr-1 and y <yr-1 and z<zr-1:
        next2 = findIndex(x,y+1,z)
        next3 = findIndex(x,y,z+1)
        if voxs[next2].getActive() !=False:
            line = rg.Line(i.getCenter(),voxs[next2].getCenter())
            lines.append(line)
            linesInds.append((i.getIndex(), voxs[next2].getIndex()))
        else:
            pass
        if voxs[next3].getActive() !=False:
            line = rg.Line(i.getCenter(),voxs[next3].getCenter())
            lines.append(line)
            linesInds.append((i.getIndex(), voxs[next3].getIndex()))
        else:
            pass
    if x<xr-1 and y == yr-1 and z<zr-1:
        next1 = findIndex(x+1,y,z)
        next3 = findIndex(x,y,z+1)
        if voxs[next1].getActive() !=False:
            line = rg.Line(i.getCenter(),voxs[next1].getCenter())
            lines.append(line)
            linesInds.append((i.getIndex(), voxs[next1].getIndex()))
        else:
            pass
        if voxs[next3].getActive() !=False:
            line = rg.Line(i.getCenter(),voxs[next3].getCenter())
            lines.append(line)
            linesInds.append((i.getIndex(), voxs[next3].getIndex()))
        else:
            pass
    if x <xr-1 and y<yr-1 and z == zr-1:
        next1 = findIndex(x+1,y,z)
        next2 = findIndex(x,y+1,z)
        if voxs[next1].getActive() !=False:
            line = rg.Line(i.getCenter(),voxs[next1].getCenter())
            lines.append(line)
            linesInds.append((i.getIndex(), voxs[next1].getIndex()))
        else:
            pass
        if voxs[next2].getActive() !=False:
            line = rg.Line(i.getCenter(),voxs[next2].getCenter())
            lines.append(line)
            linesInds.append((i.getIndex(), voxs[next2].getIndex()))
        else:
```

```
            pass

        if x == xr-1 and y == yr-1 and z < zr-1:
            next1 = findIndex(x,y,z+1)
            if voxs[next1].getActive() != False:
                line = rg.Line(i.getCenter(),voxs[next1].getCenter())
                lines.append(line)
                linesInds.append((i.getIndex(),voxs[next1].getIndex()))
            else:
                pass

        if x == xr-1 and y < yr-1 and z == zr-1:
            next1 = findIndex(x,y+1,z)
            if voxs[next1].getActive() != False:
                line = rg.Line(i.getCenter(),voxs[next1].getCenter())
                lines.append(line)
                linesInds.append((i.getIndex(),voxs[next1].getIndex()))
            else:
                pass

        if x < xr-1 and y == yr-1 and z == zr-1:
            next1 = findIndex(x+1,y,z)
            if voxs[next1].getActive() != False:
                line = rg.Line(i.getCenter(),voxs[next1].getCenter())
                lines.append(line)
                linesInds.append((i.getIndex(),voxs[next1].getIndex()))
            else:
                pass

        if x == xr-1 and y == yr-1 and z == zr-1:
            pass
    else:
        pass

supportpts = []
for i in voxs:
    if i.getSupport() == True:
        pt = i.getCenter()
        supportpts.append(pt)
```

XII. APPENDIX C: SECOND APPROACH CODES

02 Lines For Beams&Points For Support Part2

```
def dfs(graph, start):
    visited, stack = set(), [start]
    while stack:
        vertex = stack.pop()
        if vertex not in visited:
            visited.add(vertex)
            stack.extend(graph[vertex] - visited)
    return visited

def findIndex(x,y,z):
    result = int(x*(yr)*(zr) + y*(zr) + z)
    return result

graphlist = []

for i in voxs:
    if i.getActive() == True:
        edge = []
        index = i.getIndex()
        edge.append(index)
        for j in lns:
            if index in j:
                if j[0] == index:
                    edge.append(j[1])
                else:
                    edge.append(j[0])
        graphlist.append(edge)


refpts = []
for i in voxs:
    if i.getSupport() == True:
        refpts.append(i.getIndex())
print len(refpts)

graph = {}
for i in graphlist:
    a = i.pop(0)
    graph.update({a : set(i)})

newInds = dfs(graph,refpts[1])

for i in newInds:
    x,y,z = i
    ind = findIndex(x,y,z)
    voxs[ind].setActive(True)

print (newInds)

#create lines for beams
lines = []
```

```
linesInds = []
linesPts = []
for x,i in enumerate(voxs):
    if i.getActive() == True:
        x = i.getX()
        y = i.getY()
        z = i.getZ()
        if x<xr-1 and y<yr-1 and z<zr-1:
            next1 = findIndex(x+1,y,z)
            next2 = findIndex(x,y+1,z)
            next3 = findIndex(x,y,z+1)
            if voxs[next1].getActive() !=False:
                line = rg.Line(i.getCenter(),voxs[next1].getCenter())
                lines.append(line)
                linesInds.append((i.getIndex(), voxs[next1].getIndex()))
            else:
                pass
            if voxs[next2].getActive() !=False:
                line = rg.Line(i.getCenter(),voxs[next2].getCenter())
                lines.append(line)
                linesInds.append((i.getIndex(), voxs[next2].getIndex()))
            else:
                pass
            if voxs[next3].getActive() !=False:
                line = rg.Line(i.getCenter(),voxs[next3].getCenter())
                lines.append(line)
                linesInds.append((i.getIndex(), voxs[next3].getIndex()))
            else:
                pass
        if x == xr-1 and y <yr-1 and z<zr-1:
            next2 = findIndex(x,y+1,z)
            next3 = findIndex(x,y,z+1)
            if voxs[next2].getActive() !=False:
                line = rg.Line(i.getCenter(),voxs[next2].getCenter())
                lines.append(line)
                linesInds.append((i.getIndex(), voxs[next2].getIndex()))
            else:
                pass
            if voxs[next3].getActive() !=False:
                line = rg.Line(i.getCenter(),voxs[next3].getCenter())
                lines.append(line)
                linesInds.append((i.getIndex(), voxs[next3].getIndex()))
            else:
                pass
        if x<xr-1 and y == yr-1 and z<zr-1:
            next1 = findIndex(x+1,y,z)
            next3 = findIndex(x,y,z+1)
            if voxs[next1].getActive() !=False:
                line = rg.Line(i.getCenter(),voxs[next1].getCenter())
                lines.append(line)
                linesInds.append((i.getIndex(), voxs[next1].getIndex()))
```

```
    else:
        pass
    if voxs[next3].getActive() !=False:
        line = rg.Line(i.getCenter(),voxs[next3].getCenter())
        lines.append(line)
        linesInds.append((i.getIndex(), voxs[next3].getIndex()))
    else:
        pass
if x <xr-1 and y<yr-1 and z == zr-1:
    next1 = findIndex(x+1,y,z)
    next2 = findIndex(x,y+1,z)
    if voxs[next1].getActive() !=False:
        line = rg.Line(i.getCenter(),voxs[next1].getCenter())
        lines.append(line)
        linesInds.append((i.getIndex(), voxs[next1].getIndex()))
    else:
        pass
    if voxs[next2].getActive() !=False:
        line = rg.Line(i.getCenter(),voxs[next2].getCenter())
        lines.append(line)
        linesInds.append((i.getIndex(), voxs[next2].getIndex()))
    else:
        pass

if x == xr-1 and y == yr-1 and z < zr-1:
    next1 = findIndex(x,y,z+1)
    if voxs[next1].getActive() != False:
        line = rg.Line(i.getCenter(),voxs[next1].getCenter())
        lines.append(line)
        linesInds.append((i.getIndex(),voxs[next1].getIndex()))
    else:
        pass

if x == xr-1 and y < yr-1 and z == zr-1:
    next1 = findIndex(x,y+1,z)
    if voxs[next1].getActive() != False:
        line = rg.Line(i.getCenter(),voxs[next1].getCenter())
        lines.append(line)
        linesInds.append((i.getIndex(),voxs[next1].getIndex()))
    else:
        pass

if x < xr-1 and y == yr-1 and z == zr-1:
    next1 = findIndex(x+1,y,z)
    if voxs[next1].getActive() != False:
        line = rg.Line(i.getCenter(),voxs[next1].getCenter())
        lines.append(line)
        linesInds.append((i.getIndex(),voxs[next1].getIndex()))
    else:
        pass
```

```
        if x == xr-1 and y == yr-1 and z == zr-1:
            pass
    else:
        pass

print len(newInds)
for i in voxs:
    i.setActive(False)
voxsactive = []
for i in newInds:
    x,y,z = i
    newInd = findIndex(x,y,z)
    voxs[newInd].setActive(True)
    voxsactive.append(voxs[newInd].getVox())
```

XII. APPENDIX C: SECOND APPROACH CODES
03 Locating Distributed Load

```
import rhinoscriptsyntax as rs
import Rhino.Geometry as rg
import math
import Rhino
import Grasshopper
import sys
from VoxClass import Vox
import clr

#find the right index
def findIndex(x,y,z):
    result = int(x*(yr)*(zr) + y*(zr) + z)
    return result

#find the adjacencies
def findAdjacency(voxs):
    for a,i in enumerate(voxs):
        x = i.getX()
        y = i.getY()
        z = i.getZ()
        if x == 0:
            i.setAdjacency(0,False)
        if x+1 < xr:
            next = findIndex(x+1,y,z)
            if voxs[next].getActive() == False:
                i.setAdjacency(2,False)
                if i.getActive() == False:
                    voxs[next].setAdjacency(0,False)
                else:
                    voxs[next].setAdjacency(0,True)
            else:
                i.setAdjacency(2,True)
                if i.getActive() == False:
                    voxs[next].setAdjacency(0,False)
                else:
                    voxs[next].setAdjacency(0,True)
        if x == xr-1:
            i.setAdjacency(2,False)
        if y == 0:
            i.setAdjacency(1,False)
        if y+1 < yr:
            next = findIndex(x,y+1,z)
            if voxs[next].getActive() == False:
                i.setAdjacency(3,False)
                if i.getActive() == False:
                    voxs[next].setAdjacency(1,False)
                else:
                    voxs[next].setAdjacency(1,True)
            else:
                i.setAdjacency(3,True)
                if i.getActive() == False:
```

```
                voxs[next].setAdjacency(1,False)
            else:
                voxs[next].setAdjacency(1,True)
        if y == yr-1:
            i.setAdjacency(3,False)
        if z == 0:
            i.setAdjacency(5,False)
        if z+1 < zr:
            next = findIndex(x,y,z+1)
            if voxs[next].getActive() == False:
                i.setAdjacency(4,False)
                if i.getActive() == False:
                    voxs[next].setAdjacency(5,False)
                else:
                    voxs[next].setAdjacency(5,True)
            else:
                i.setAdjacency(4,True)
                if i.getActive() == False:
                    voxs[next].setAdjacency(5,False)
                else:
                    voxs[next].setAdjacency(5,True)
        if z == zr-1:
            i.setAdjacency(4,False)

findAdjacency(voxs)
loadvox = []
voxBox = []
for i in voxs:
    if i.getActive() == True:
        if i.getAdjacency()[4] == False:
            loadvox.append(i)
            voxBox.append(i.getVox())

loadedbeams = []
loadedbeamsId = []
for i in loadvox:
    for n,j in enumerate(linesInds):
        x,y,z = i.getIndex()
        xj0, yj0, zj0 = j[0]
        xj1, yj1, zj1 = j[1]
        if i.getIndex() == j[0] and z == zj1:
            print i.getIndex()
            print j
            loadedbeams.append(lines[n])
            loadedbeamsId.append(linesInds[n])
        elif i.getIndex() == j[1] and z == zj0:
            loadedbeams.append(lines[n])
            loadedbeamsId.append(linesInds[n])
        else:
            pass
```

XII. APPENDIX C: SECOND APPROACH CODES
04 Force To Voxel

```
import rhinoscriptsyntax as rs
import Rhino.Geometry as rg
import math
from VoxClass import Vox
import clr
clr.AddReferenceToFileAndPath("C:\Program Files\Rhino 6\Plug-ins\karamba.gha")
import operator

class Force:
    def __init__(self, force, vox0,vox1):
        self.force = force
        self.vox = vox0
        self.direction = vox1
        self.voxInd = (vox0,vox1)
        self.index = None
    def getForce(self):
        return self.force
    def getVox(self):
        return self.vox
    def getDirection(self):
        return self.direction
    def getVoxInd(self):
        return self.voxInd
    def setIndex(self,x):
        self.index = x
    def getIndex(self):
        return self.index

axialForces0 = axialForces[:len(axialForces)//2]
axialForces1 = axialForces[len(axialForces)//2:]

axialFs0 = [axialForces0[x:x+2] for x in xrange(0,len(axialForces0),2)]
axialFs1 = [axialForces1[x:x+2] for x in xrange(0,len(axialForces1),2)]

axialFs = []
for x,i in enumerate(axialFs0):
    axialF = i+axialFs1[x]
    axialFs.append(axialF)

forces = []
for x,force in enumerate(axialFs):
    a = Force(force[0],lineInds[x][0],lineInds[x][1])
    a.setIndex(x)
    b = Force(force[1],lineInds[x][1],lineInds[x][0])
    b.setIndex(x)
    forces.append(a)
    forces.append(b)

voxsActive = []
for i in voxs:
    if i.getActive() != False:
```

```
        voxsActive.append(i)

forcesActive = []
for i in forces:
    for j in voxsActive:
        if i.getVox() == j.getIndex():
            forcesActive.append(i)

voxs_sorted = sorted(voxsActive,key = operator.attrgetter('index'))
forces_sorted = sorted(forcesActive,key=operator.attrgetter('vox'))

voxForces= []
for i in range(len(voxs_sorted)):
    a = []
    for x,j in enumerate(forces_sorted):
        if j.getVox() == voxs_sorted[i].getIndex():
            a.append(j)
    voxForces.append(a)

def findIndex(x,y,z):
    result = int(x*(yr)*(zr) + y*(zr) + z)
    return result

totdirs = []
for i in voxForces:
    for j in i:
        xdir = []
        ydir = []
        zdir = []
        totxdir = 0
        totydir = 0
        totzdir = 0
        xa,ya,za = j.getVox()
        xb,yb,zb = j.getDirection()
        index = findIndex(xa,ya,za)
        if ya == yb and za == zb:
            xdir.append(j.getForce())
            if len(xdir) == 1:
                totxdir = -xdir[0]
            elif len(xdir) == 2:
                totxdir = xdir[0] + xdir[1]
            else:
                totxdir = 0
            voxs[index].setTotalForceX(totxdir)
            voxs[index].setTotalStressX(math.pow(crosec,2))
        elif xa == xb and za == zb:
            ydir.append(j.getForce())
            if len(ydir) == 1:
                totydir = -ydir[0]
            elif len(ydir) == 2:
                totydir = ydir[0] + ydir[1]
```

XII. APPENDIX C: SECOND APPROACH CODES
04 Force To Voxel

```
            else:
                totydir = 0
            voxs[index].setTotalForceY(totydir)
            voxs[index].setTotalStressY(math.pow(crosec,2))
        elif xa == xb and ya == yb:
            zdir.append(j.getForce())
            if len(zdir) == 1:
                totzdir = -zdir[0]
            elif len(zdir) == 2:
                totzdir = zdir[0] + zdir[1]
            else:
                totzdir = 0
            voxs[index].setTotalForceZ(totzdir)
            voxs[index].setTotalStressZ(math.pow(crosec,2))


for i in voxs:
    if i.getActive() !=False:
        if i.getTotalForceX() == None:
            i.setTotalForceX(0)
        if i.getTotalForceY() == None:
            i.setTotalForceY(0)
        if i.getTotalForceZ() == None:
            i.setTotalForceZ(0)
        i.setTotalForce()       elif xa == xb and ya == yb:
            zdir.append(j.getForce())
            if len(zdir) == 1:
                totzdir = -zdir[0]
            elif len(zdir) == 2:
                totzdir = zdir[0] + zdir[1]
            else:
                totzdir = 0
            voxs[index].setTotalForceZ(totzdir)
            voxs[index].setTotalStressZ(math.pow(crosec,2))


for i in voxs:
    if i.getActive() !=False:
        if i.getTotalForceX() == None:
            i.setTotalForceX(0)
        if i.getTotalForceY() == None:
            i.setTotalForceY(0)
        if i.getTotalForceZ() == None:
            i.setTotalForceZ(0)
        i.setTotalForce()
```

XII. APPENDIX C: SECOND APPROACH CODES
05 Voxel Remove

```
import rhinoscriptsyntax as rs
import Rhino.Geometry as rg
import math
import clr
clr.AddReferenceToFileAndPath("C:\Program Files\Rhino 6\Plug-ins\karamba.gha")
import operator
from ForceClass import Force
from VoxClass import Vox

def findIndex(x,y,z):
    result = int(x*(yr)*(zr) + y*(zr) + z)
    return result

for i in voxs:
    print i.getActive()

actVox = []
for i in voxs:
    if i.getActive() == True and i.getSupport() == False:
        actVox.append(i)

comp_sort = []
comp_sorted = []
comp_sorted0 = []
comp_sorted1 = []
comp_sorted2 = []
comp_sorted3 = []
comp_sorted4 = []

for i in actVox:
    x,y,z = i.getTotalForce()
    if z == 0:
        comp_sorted.append(i)
    if z > 0:
        comp_sorted0.append(i)
    elif x > 0 and y > 0 and z<0:
        comp_sorted1.append(i)
    elif x > 0 or y > 0 and z<0:
        if x > 0:
            comp_sorted2.append(i)
        elif y > 0:
            comp_sorted3.append(i)
    elif x <= 0 and y<=0 and z<=0:
        comp_sorted4.append(i)

comp = sorted(comp_sorted,key = operator.attrgetter('totalForceXY'))
comp0 = sorted(comp_sorted0,key = operator.attrgetter('totalForceZ'))
comp1 = sorted(comp_sorted1,key = operator.attrgetter('totalForceXY'))
comp2 = sorted(comp_sorted2,key = operator.attrgetter('totalForceX'))
comp3 = sorted(comp_sorted3,key = operator.attrgetter('totalForceY'))
comp4 = sorted(comp_sorted4,key = operator.attrgetter('totalForceVect'))
```

```
comp_sort.append(comp)
comp_sort.append(comp0)
comp_sort.append(comp1)
comp_sort.append(comp2)
comp_sort.append(comp3)
comp_sort.append(comp4)

voxRemoved = []
count = 0
if count < (NRemove):
    if len(comp_sort[0]) > NRemove:
        for j in range(NRemove):
            comp_sort[0][j].setActive(False)
            voxRemoved.append(comp_sort[0][j])
            count+=1
    elif len(comp_sort[0]) > 0:
        for i in comp_sort[0]:
            i.setActive(False)
            voxRemoved.append(i)
            count+=1
        for j in range(NRemove-len(comp_sort[0])):
            comp_sort[1][j].setActive(False)
            voxRemoved.append(comp_sort[1][j])
            count+=1
    elif len(comp_sort[1]) > NRemove:
        for j in range(NRemove):
            comp_sort[1][j].setActive(False)
            voxRemoved.append(comp_sort[1][j])
            count+=1
    elif len(comp_sort[1]) > 0:
        for i in comp_sort[1]:
            i.setActive(False)
            count+=1
            voxRemoved.append(i)
        for j in range(NRemove-len(comp_sort[1])):
            comp_sort[2][j].setActive(False)
            count+=1
            voxRemoved.append(comp_sort[2][j])
    elif len(comp_sort[2]) > NRemove:
        for j in range(NRemove):
            comp_sort[2][j].setActive(False)
            count+=1
            voxRemoved.append(comp_sort[2][j])
    elif len(comp_sort[2]) > 0:
        for i in comp_sort[2]:
            i.setActive(False)
            count+=1
            voxRemoved.append(i)
        for j in range(NRemove-len(comp_sort[2])):
            comp_sort[3][j].setActive(False)
```

XII. APPENDIX C: SECOND APPROACH CODES
05 Voxel Remove

```
            count+=1
            voxRemoved.append(comp_sort[3][j])
    elif len(comp_sort[3]) > NRemove:
        for j in range(NRemove):
            comp_sort[3][j].setActive(False)
            count+=1
            voxRemoved.append(comp_sort[3][j])
    elif len(comp_sort[3]) > 0:
        for i in comp_sort[3]:
            i.setActive(False)
            count+=1
            voxRemoved.append(i)
        for j in range(NRemove-len(comp_sort[3])):
            comp_sort[4][j].setActive(False)
            count+=1
            voxRemoved.append(comp_sort[4][j])
    elif len(comp_sort[4]) > 0:
        for i in comp_sort[4]:
            i.setActive(False)
            count+=1
            voxRemoved.append(i)
        for j in range(NRemove-len(comp_sort[4])):
            comp_sort[5][j].setActive(False)
            count+=1
            voxRemoved.append(comp_sort[5][j])

compressive = []
compVoxs = []
compInds = []

for i in actVox:
    if i.getActive() == True:
        x,y,z = i.getIndex()
        voxs[findIndex(x,y,z)].setActive(True)
    else:
        x,y,z = i.getIndex()
        voxs[findIndex(x,y,z)].setActive(False)

for i in voxs:
    if i.getActive() == True:
        compressive.append(True)
        compVoxs.append(i.getVox())
        compInds.append(i.getIndex())
    else:
        compressive.append(False)
```

XII. APPENDIX C: SECOND APPROACH CODES
06 Voxel To Mesh

```
#find the right index
def findIndex(x,y,z):
    result = int(x*(yr)*(zr) + y*(zr) + z)
    return result

mesh = rg.Mesh()


#find the adjacencies
for a,i in enumerate(voxs):
    x = i.getX()
    y = i.getY()
    z = i.getZ()
    print "x"
    print x
    print "y"
    print y
    print "Z"
    print z

    if x == 0:
        i.setAdjacency(0,False)
    if x+1 < xr:
        next = findIndex(x+1,y,z)
        if voxs[next].getActive() == False:
            i.setAdjacency(2,False)
            if i.getActive() == False:
                voxs[next].setAdjacency(0,False)
            else:
                voxs[next].setAdjacency(0,True)
        else:
            i.setAdjacency(2,True)
            if i.getActive() == False:
                voxs[next].setAdjacency(0,False)
            else:
                voxs[next].setAdjacency(0,True)
    if x == xr-1:
        i.setAdjacency(2,False)

    if y == 0:
        i.setAdjacency(1,False)
    if y+1 < yr:
        next = findIndex(x,y+1,z)
        if voxs[next].getActive() == False:
            i.setAdjacency(3,False)
            if i.getActive() == False:
                voxs[next].setAdjacency(1,False)
            else:
                voxs[next].setAdjacency(1,True)
        else:
            i.setAdjacency(3,True)
```

```
            if i.getActive() == False:
                voxs[next].setAdjacency(1,False)
            else:
                voxs[next].setAdjacency(1,True)
    if y == yr-1:
        i.setAdjacency(3,False)

    if z == 0:
        i.setAdjacency(5,False)
    if z+1 < zr:
        next = findIndex(x,y,z+1)
        if voxs[next].getActive() == False:
            i.setAdjacency(4,False)
            if i.getActive() == False:
                voxs[next].setAdjacency(5,False)
            else:
                voxs[next].setAdjacency(5,True)
        else:
            i.setAdjacency(4,True)
            if i.getActive() == False:
                voxs[next].setAdjacency(5,False)
            else:
                voxs[next].setAdjacency(5,True)
    if z == zr-1:
        i.setAdjacency(4,False)

#find boundary voxels
boundaryVoxs = []
for i in voxs:
    if i.getActive() != False:
        count = 0
        for x,j in enumerate(i.getAdjacency()):
            if j == False:
                count+=1
            if count > 0:
                i.setIsEdge(True)
                boundaryVoxs.append(i.getVox())

meshVerts = []
for i in voxs:
    if i.getActive() != False:
        for x,j in enumerate(i.getAdjacency()):
            if j == False:
                if x == 0:
                    boxCorners = rg.BoundingBox.GetCorners(i.getVox())
                    meshVerts.append(boxCorners[0])
                    meshVerts.append(boxCorners[3])
                    meshVerts.append(boxCorners[7])
                    meshVerts.append(boxCorners[4])
                if x == 1:
                    boxCorners = rg.BoundingBox.GetCorners(i.getVox())
```

XII. APPENDIX C: SECOND APPROACH CODES
06 Voxel To Mesh

```
            meshVerts.append(boxCorners[1])
            meshVerts.append(boxCorners[0])
            meshVerts.append(boxCorners[4])
            meshVerts.append(boxCorners[5])
        if x == 2:
            boxCorners = rg.BoundingBox.GetCorners(i.getVox())
            meshVerts.append(boxCorners[2])
            meshVerts.append(boxCorners[1])
            meshVerts.append(boxCorners[5])
            meshVerts.append(boxCorners[6])
        if x == 3:
            boxCorners = rg.BoundingBox.GetCorners(i.getVox())
            meshVerts.append(boxCorners[3])
            meshVerts.append(boxCorners[2])
            meshVerts.append(boxCorners[6])
            meshVerts.append(boxCorners[7])
        if x == 4:
            boxCorners = rg.BoundingBox.GetCorners(i.getVox())
            meshVerts.append(boxCorners[4])
            meshVerts.append(boxCorners[7])
            meshVerts.append(boxCorners[6])
            meshVerts.append(boxCorners[5])
        if x == 5:
            boxCorners = rg.BoundingBox.GetCorners(i.getVox())
            meshVerts.append(boxCorners[0])
            meshVerts.append(boxCorners[3])
            meshVerts.append(boxCorners[2])
            meshVerts.append(boxCorners[1])

faceVerts = [meshVerts[x:x+4] for x in xrange(0, len(meshVerts), 4)]

for i in faceVerts:
    a = mesh.Vertices.Add(i[0])
    b = mesh.Vertices.Add(i[1])
    c = mesh.Vertices.Add(i[2])
    d = mesh.Vertices.Add(i[3])
    face = mesh.Faces.AddFace(a,b,c,d)
```