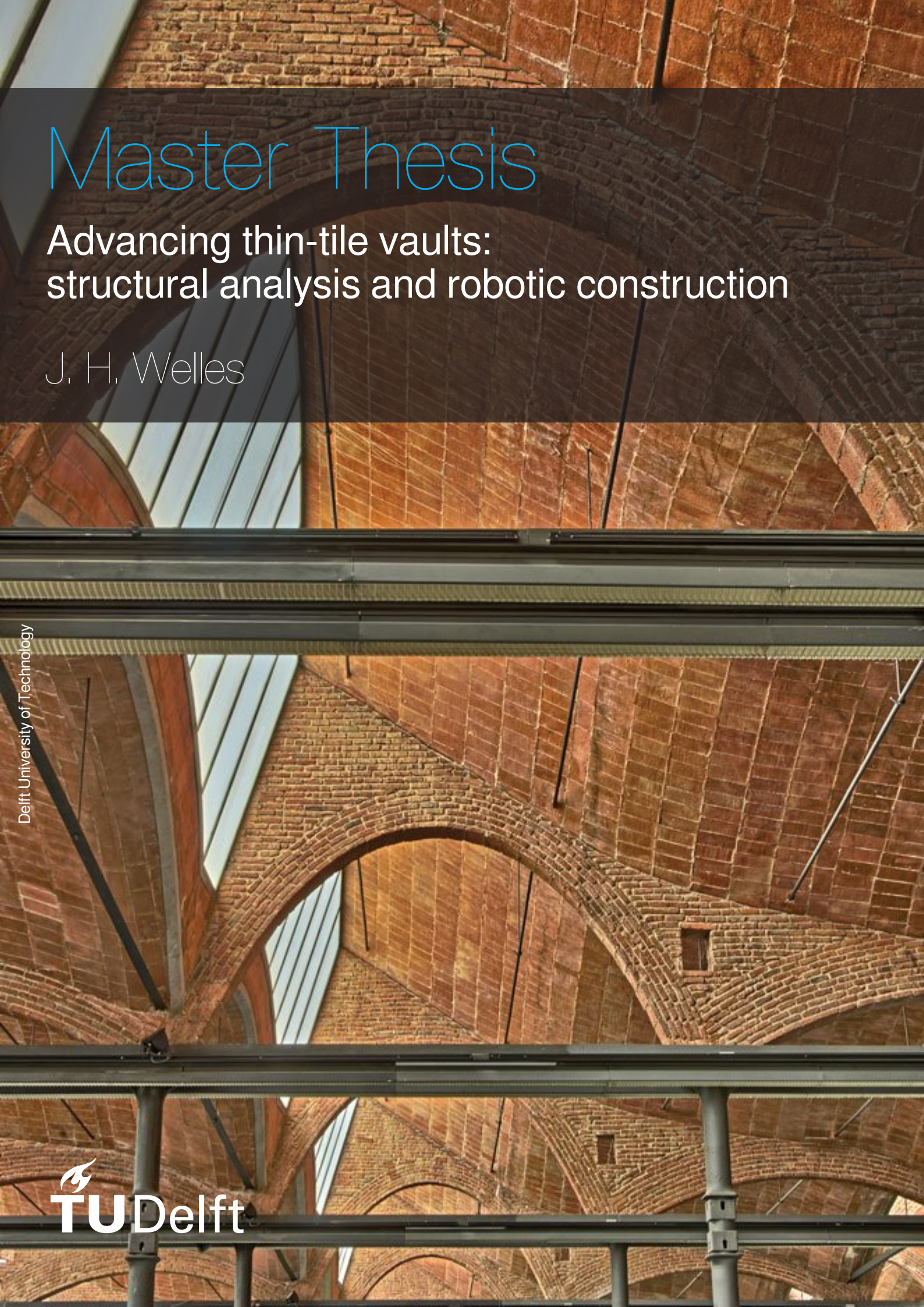


Master Thesis

Advancing thin-tile vaults: structural analysis and robotic construction

J. H. Welles

Delft University of Technology

TU Delft

Master Thesis

Advancing thin-tile vaults: structural analysis and robotic construction

by

J. H. Welles

in partial fulfilment of the requirements for the degree of

Master of Science

in Civil Engineering (Building Engineering)

at the Delft University of Technology,

to be defended publicly on Tuesday, second of November, 2021

Student Number:	4270592		
Project Duration:	December, 2020 - October, 2021		
Faculty:	Civil Engineering and Geosciences (CiTG)		
Thesis Committee:	prof. dr. ir. J.G. Rots	3MD (CiTG)	(chair)
	dr. S. Asut	AE+T (BK)	
	ir. L.P.L. van der Linden	3MD (CiTG)	
	ir. J. Driessen	WSP	(daily supervisor)

This thesis can be found online on https://repository.tudelft.nl

Cover Image: Roof mNACTEC, The Art of Structural Tile by J. Ochsendorf

TU Delft

Preface

It is hard to imagine to have reached this point, where I can hand in my last work as student. With the completion of this report, a new phase in my life will start. I have completed my Masters, and with that the complete study of Civil Engineering at the TU Delft. I see getting to this point as one of hard work but also one of coincidences. For instance, besides the minimal curriculum for the specialisation Structural Design in the master track Building Engineering, I have completed the course on Structural Dynamics. My motivation for doing this was something the supervisor at my internship said: if I had done anything with dynamics, and my answer was no. Another example of these coincidences is the topic of this thesis. One day, while finishing the report for one of my courses, I needed to find a representative value for the thickness of a shell structure. I came across the website of Pierre Hoogenboom. He had a video link on his website to something called 'Catalan vaults'. Intrigued by the video I started looking into this topic and after a long journey, this is the end product of that spark.

I would like to express my gratitude to Jeroen Driessen, who has helped me for the past months immensely with keeping me focused on the topic and improving my output. I've enjoyed our meetings very much and I am grateful I had the chance to work with you, also at the office of WSP. I am very grateful to the rest of my committee as well. Jan has shown interest in my topic from the start, even relating it to projects with the *Utrechtse werfkelders*. Lennert has given me the assurance and provided me sources to tackle the reporting of my research. With Serdar I have teamed up to make the most out of my robotic simulation. I think I have taught you something as well, in return. For your guidance, knowledge and company, I am ever grateful. A special mention for Paul Korswagen and Telesilla Bristogianni, who helped me with DIANA and epoxies, respectively.

I would also like to thank all those that have stood by me during my thesis. First of all I would like to thank Kostas, with whom I've started and ended this process simultaneously. Our cooperation, discussions and lunch breaks have been a part I would not have wanted to miss at all. I am glad we have experienced this process together. Secondly, I would like to thank Bas, who has helped me with my brick experiment. If you want you can share in this memory by taking some bricks with you, Bas. Furthermore, I would like to thank a couple of students with whom I've had the pleasure to enjoy the master thesis with, even if it was for only a part. Thank you, Thijs, Thijs and Laura. Additionally, all the other people at U-BASE and at the TU Delft with whom I've had the pleasure to share drinks, laughs, ideas and frustrations with. Lastly, I would like to thank my family and (non TU-Delft) friends, who have stood by me and supported me in their own way.

Nothing else can be said than to wish the reader a joyful read, and good luck!

J. H. Welles
Delft, November 2021

Summary

Thin-tile vaults are a a type of vaults that went out of fashion in the early twentieth century. Its origins are around the Mediterranean, but modern interest is mostly due to Guastavino, and the research done at MIT and ETH. The thin-tile vaults have a unique construction method without any temporary support. Eventually the increase in labour costs and the advancements in concrete and steel made the structure non-competitive.

Robotics are a type of machines that can perform (semi)-automated tasks. In the past decades the development of robots have led to their implementation in the construction industry. Robots developed specifically for masonry show a high promise where they're able to lay much more bricks than even the most skilled mason.

This research aims to investigate the time it takes for a robot to construct a thin-tile vault, and thereafter to advance the possibilities of research into and construction of the thin-tile vault. This is researched by answering the following question:

How does a robotic construction of a parametrically designed thin-tile vault perform based on step-wise structural analyses?

To answer this question a parametric model has been made to include the design of the thin-tile vault, the structural analysis and the robotic construction. This model is made in Grasshopper, but is supported by Python and RoboDK. Python performs any calculation necessary and analysis of this output. RoboDK is a robotic simulation program aimed to give users a tool to translate their design to robotic instructions. The structure of this report is based on the three aspects of the parametric model.

In the first part the state of the art and the relevance of this research is stated. Thereafter the objectives, questions and methodology are noted.

The first part of the model is related to the design. In the first chapter a literature review is provided on thin-tile vaults and similar vault structures. The second chapter of this part describes how the design model has been made. First the global shape of the vault is set as a barrel vault formed as a catenary arch. Then the bricks are placed on this vault surface using a similar approach as map projection. This is done with the introduction of the centre point approach. This ensures that proper information is maintained, like the course and the position within the course per brick. With this approach it is possible to fill this surface even with courses misaligned to the primary curvature. Bricks at the edge of the vaults surface are cut to fit within its domain. The last step is to thicken the bricks, which thus far had been represented as surfaces, into volumes.

The structural analysis consists of three major parts. In the first chapter literature is shown to justify the use of a linear elastic analysis on masonry. Additionally the behaviour of this uncommon type of masonry is tried to be found in literature. The materials brick, mortar and epoxy are looked into as well. Bricks have time-independent properties, making these similar as found in the Eurocode, by manufacturers and in research. Mortar and gypsum plaster especially (also known as Plaster of Paris), are described in their material properties as well. Due to uncertainty in these properties, epoxy is investigated as well and found to have more theoretical values useful for this research. A small-scale experiment with bricks and epoxy has been done to verify what values from epoxy should be used.

After the literature review on material properties, the structural analysis is done. The flow of this chapter starts with the form from the design model, through the forces of a cantilever, to the stresses occurring in the structure. The vault has been simplified to an arch during construction. Thrust is not present. The sectional forces are found by first considering the (partial) arch as a cantilever with a vertical shear force and rotational moment. Next this shear force is converted into the normal and shear force in the cross-section of the arch. This calculation has been worked out in a Python script.

In the last chapter a possible stress distribution is shown. The analysis is based on a phased structural analysis. Each time step is equivalent to the placement of one row of bricks, which have a similar cantilever length. The stresses of the bottom and top side are shown, as well as the stresses between the wythes of the vault. In a DIANA model it is shown these wythes behave more like a monolithic material than a layered one.

The last part is related to the robotic construction. First literature is shown related to masonry robotics. In the second chapter the stations the robot visits, are described. At the pallet station three instructions are described. The adhesive station has a number of instructions which is dividable by four. The vault station again has three instructions. This is similar to a pick&place work order, but with the adhesive station in between. Two configurations are considered. The first is when the robot is outside the vault, where it is closer to the support than to the apex. In the second configuration the robot is placed within the vault, where it is positioned close to the apex. In the last chapter the robot is chosen. Additionally the path is described with boundary conditions.

The results are shown for 8 computations: a basis computation; one with the robot changed; one with the shape of the vault changed; one with the orientations of the wythes changed; one with an increase of the adhesive hardening time; one with a small course length of the vault; one with a different work space configuration; and lastly one with a different preferred construction sequence.

In conclusion the models and results show that the construction of a thin-tile vault is similar to other masonry robotics, if the hardening time is excluded. Included, the number of bricks built per day is similar to two masons building a thin-tile vault prototype. The hardening time is the primary influence of the total construction time and any reduction here is advised. Further optimisation of the movement of the robots is possible with a more in-depth optimization of the work space configuration and an optimization where the shortest movements are considered across instructions. The linear-elastic structural analysis has been found to be possible to use for the calculation of the thin-tile vault during construction. Additionally, two construction sequence preferences work well with the multiple wythes, but one preference shows better results when more bricks with more cantilever length need to be placed, due to reducing the stresses on the edge of the extrados wythe.

Further research into thin-tile vaults is still required to fully understand and model the thin-tile vault. Related to the three models (design, engineering and robotics) each have been improved in the models made for this research, but are still a long way away from full implementation.

Contents

List of Figures

List of Tables

Part I

Introduction

Figure 1: Example of thin-tile vaults: Droneport.

The building industry has a long and diverse history. In the past centuries the knowledge of structural mechanics, building materials and construction techniques has increased significantly. Machinery is now a necessity on and off the construction site, whether it being a lorry or a crane. With the addition of digital advances, combining machine work with offsite oversight from experts has become a possibility as well. While these modern technologies expand the realm of possibilities for the building industry, old techniques and applications have gathered interest as well. Vernacular architecture has seen a resurgence now the standard palette of the structural engineer has expanded beyond steel and concrete structures. Although both directions should be encouraged, this research will focus on combining an old technique with new innovations.

One vernacular construction technique that has sparked more interest, is that of the Catalan vault. Also known as the timbrel vault and bóveda tabicada, the Catalan vault is a thin-shell structure with, traditionally, alternating layers of (very) thin bricks and mortar (Benfratello et al., 2012). The timbrel vault has advantages compared to other vaults and spanning (floor) structures. Due to the low self-weight of the structure, the lateral thrust on its foundation (most likely vertical elements) is reduced. The vault's design also allows for a better rise/span ratio compared to the better-known barrel vault, Roman arch and pointed arch. However, due to the properties of a well-designed arch, it requires no or minimal (steel) reinforcement, unlike most modern beams and slabs. One more advantage is the construction technique. Whereas Roman arches need a centring as support until the keystone is put in place and prefabricated slabs need falsework until the in-situ concrete has dried, the Catalan vault can be constructed without the need of any falsework, besides guide strings. This makes the timbrel vault an interesting structural design to improve for further use.

The timbrel vault did go out of fashion in the twentieth century. One of the reasons is the labour-intensive construction method. With modern technologies, the Catalan vault can once again become a feasible construction technique. In the context of this research the modern technology of robotics can be the solution to reduce the labour costs and in return make the Catalan vault more desirable. Robotics boils down to the use of robots for tasks previously done or impossible to accomplish by humans. Since the 1980's robotics have been implemented in the building industry to increase the quality and quantity of the products and processes, while reducing costs and time on others (Bock, 2007). A variety of robotic applications has been developed to cater to the needs of the building industry. With the complex structure of the timbrel vault the range of applications for robotics may be expanded further.

1

Research motivation

This research sets out to combine the vernacular architecture of the Catalan vault and the modern technology of robotics. This chapter shows the relevance and development into both areas by giving an historic overview, an overview of applications and recent research and interest in both. The third part of this chapter focuses on the overlap of both areas. Here, an overview of the current research and possibilities of the digital construction of brick structures is given.

1.1. The timbrel vault

The use of bricks as a building material has been common place for millennia. To see the origins of the Catalan vault, the use of its main visual characteristics throughout history has to be found. As set out by Benfratello et al., 2012 the characteristics of the Catalan vault are the flat-oriented bricks, the gypsum and the absence of ribs. These three, combined or individually, are found throughout the Mesopotamian and Mediterranean regions, where the latter followed the first. The period of use in Spanish and Italian regions can be seen from the end of the Middle Ages up to the twentieth century. It is clear from documents at the time that the main advantage to other arch-like structures was known. The reduced weight due to the material choice and orientation results in a lower thrust on its foundation.

In the eighteenth-century further research into the Catalan vault resulted in a broader understanding and a further spread of knowledge. In this age the fire resistance and monolithic behaviour of the vault has been studied. Starting in France the structure gathered more interest in Europe with an overview of the state of the art of the Catalan vault published in French, Italian and German.

The most well-known buildings with the bovéda tabicada are built around the turn of the twentieth century. One company in particular used the structural and safety properties of the Catalan vault to create a business. The Catalan architect Rafael Guastavino Moreno used his expertise to construct thousands of buildings in the United States with the use of the Catalan vault. The Guastavino room in the Boston Public Library, parts of Carnegie Hall, the registration hall of Ellis Island and the Oyster bar in the Grand Central Station of New York (Ochsendorf & Freeman, 2010), which is also a whispering gallery, are some examples of of his work.

In order to proof the structural safety of the Catalan vault, or Guastavino vault in the US, Guastavino developed various experiments and studies to provide empirical and theoretical evidence. He attributed the structural performance of the vault primarily to the multiple layers combined with the mortar. Benfratello et al., 2012 disputes that claim and points out that the cohesion between the layers is mostly beneficial for the constructional performance. The mortar makes it possible to create angled

Figure 1.1: Guastavino on top of the vaults for the Boston Public Library.

Figure 1.2: Grand Hall on Ellis Island.

gaps between the bricks, whereas a joint due to the gravitational friction is only as good as where the elements touch. Guastavino compiled his theoretical works in Essay on the Theory and History of Cohesive Construction Applied Especially to the Timbrel Vault (1892). Additional research has shown as well that Guastavino's thinking was not completely scientific, but more of experimental nature (Dugum, 2013).

His son continued with his fathers' work and used the membrane theory to get to the internal stresses of the vault. He and his father issued a number of patents, with one about the reinforcement of the brick structure.

Meanwhile in other parts of the world architects like Gaudi and Dieste were exploring geometric variations. Gaudi, known for his work in Barcelona and Catalonia, explored different geometric and spatial designs, while adhering to the technique of the Catalan vault. Dieste worked in an environment where materials like timber and steel were lacking. Thus, he relied on the principles of the Catalan vault to span large spaces. Nowadays the existing structures are in need of repair and maintenance. To improve these operations, various institutions across the world have been exploring and researching this vernacular architecture. John Ochsendorf at MIT, Boston, is one of the leading experts on the Catalan vault and the tabicada technique. He has published a book about Guastavino: Guastavino Vaulting: The Art of Structural Tile.

Research in the USA related to this topic is concentrated around MIT. This is not surprising, given the location of these structures. In Europe the research is more spread out. In this regard a major distinction can be made between the research into existing structures and the exploration of building and designing the structure. At Italian and Spanish universities, again regions where the thin-tile vault is common place, the focus is more on this first category. At the ETH in Zürich, especially by the Block Research Group, the focus has been on producing the structure. Their latest research in thin-tile vaults is from 2016 and focuses on the state of the art of thin-tile vaults. In recent years their focus has been on failure of unreinforced masonry structures, with extra attention for gothic structures, and on concrete shell structures combined with its formwork, integration of the thin-tile vault and the double-curved shell.

1.2. Robotics

The definition of a robot, according to the Cambridge dictionary, is a machine controlled by a computer that is used to perform jobs automatically.

Within the building industry, the use of robotics started in the 1980's. Research and application have been motivated with various reasons, of which the trends in workforce; concern for health & safety; the physical environment; and progress in robotics technology are some examples (Haas et al., 1995). In the early years of robotics in civil engineering research has been in a wide range of applications and phases of the building industry, with siteworks being the largest group. These years also saw a regional difference, with Japan being the clear leader in the development of robotics in the building industry. Over these years, a relative shift has occurred in the papers from conceptual to prototype systems.

At the end of the twentieth century, advancements has been made in various areas, of which the most important have been in the motion control of these robots, the automated levelling and grading and the tipping- and proximity sensing. The paper from 1995 already predicts machines that can lay bricks robotically.

Li, 2018 defines three different types of robotics used in the construction industry. These types are the wearable robots, the robotic arm and the traditional robot which resembles the human form (with arms and legs). The trend described by this paper comes from Haidegger et al., 2013 and shows that the intelligence of the robots is evolving. The distinction shown is split in industrial robots and service robots. Here, industrial robots follow set forth procedures without the ability to adapt to a changing environment, while service robots can interact with their surroundings.

From this paper comes figure 1.3, where the

description makes it clear that a bricklaying robot is most likely to be between the two categories. Although this is dependent on the fabrication and construction process. If the fabrication is done off-site in a factory, a changing environment or interaction with humans is limited. When the robot is on an operating building site, the situation will change often, with changing weather and human movements as examples.

As Bloss, 2014a describes, architectural projects and research related to robotics are executed with three software packages: Rhino, Grasshopper and YOUR. As the paper describes:

> "Grasshopper is a visual programming language used in conjunction with Rhino. It allows the programmer to drag and drop components to construct three-dimensional (3-D) model design. Rhino (Rhinoceros 3-D) is a graphic modeling tool software package widely used in architectural robotics projects. YOUR is a software package used to create a 3-D interface from the computer. Together, these three software tools are used by many of the research teams."

The paper also shows the application of a robotic arm to make a small-scale brick wall.

Besides the earlier mentioned robotic types, other types of robots are now in development. These include the use of flying robotics, like drones, or swarm-operated robots.

1.3. Digital construction of brick structures

The use of robotically constructed masonry structures can be split into three categories. The first category is the architectural exploration of brick-laying. The primary focus of this research is the expansion on design alternatives, mainly for brick walls. The main alterations to the more convenltional brick wall designs are on the shape of the wall overall and of the brick pattern itself. The ETH in Zürich has projects related to DFAB and NCCR. The focus of this organisation is in the digital fabrication and it has had some cooperation with the Block Research Group in the past. One of their projects can be seen in figure 1.4.

The second category is the development of brick-laying robots to be used in on-site construction. This robotic development is different from the

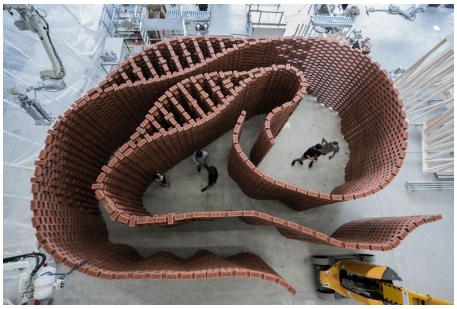


Figure 1.4: Brick Labyrinth, in Zurich 2017.

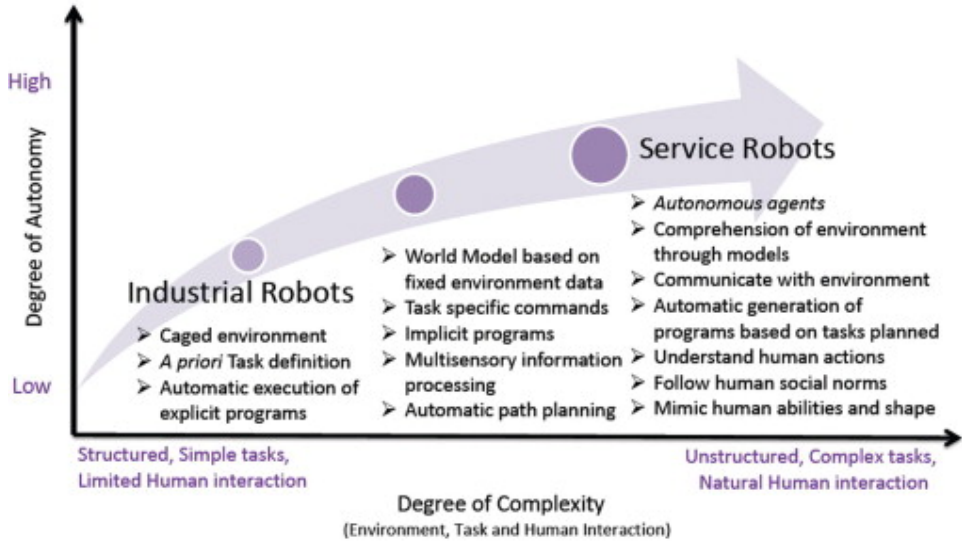


Figure 1.3: Evolution of robotics. Current trends are leading towards more complex, more personalized systems and robot services. This implies flexible systems that are able to perform tasks in an unconstrained, human-centered environment.

(a) SAM100

(b) MULE135

Figure 1.5: Masonry robotics

previous category in two ways. The first is that the robots are to be used on the construction site and the development of these robots has a direct driver to be used by contractors. The second difference can already be seen in figure 1.4. The architectural exploration has a focus on the placing of bricks, but masonry structures also include the application of mortar between the bricks. Thus, this category develops robots that can apply mortar to bricks as well. The main output of these robots is their speed. With the procedure and quality of the construction set, the main development is to improve the speed of the brick-laying. However, the architectural exploration allows for differently orientated bricks, these quick brick-laying robots are limited in the designs they can be applied for. Two examples of commercially available robots aiding in the brick laying are MULE135 and SAM100 from Construction Robotics.

The third category is those structures that are not a wall. This research is related most to this category, with some overlap to the second. The use of robots for masonry structures besides walls is somewhat limited. Recently a vault of funicular shape has been built with the help of two robotic arms (Parascho et al., 2020). This vault was made using the glass brick expertise from the TU Delft. These glass bricks were placed in a fishbone pattern by the robots, after an initial, stable arch was completed.

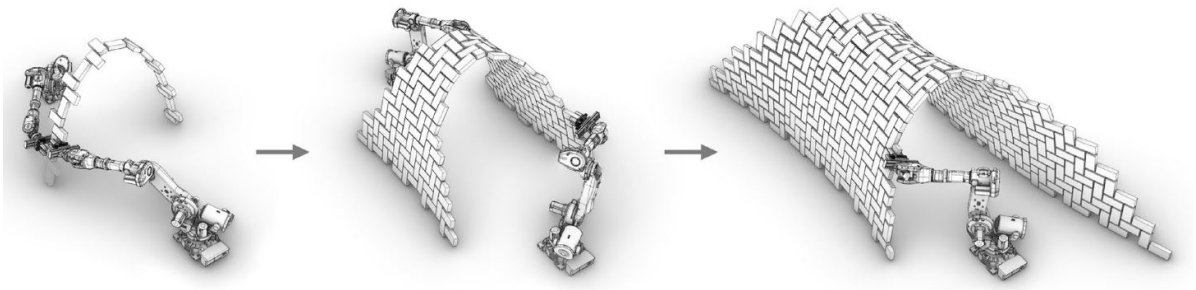


Figure 1.6: Cooperative assembly method. First phase: the middle arch is built by alternating the robots used to place and then support the structure. Second phase: the construction is continued on either side of the middle arch

2

Research definition

2.1. Aim & objectives

This research aims to investigate the time it takes for a robot to construct the vault. This may result in an advance of the possibilities of research into and construction of the thin-tile vault.

A couple of terms are used to describe the investigated structure, some adequately, others confusingly. Common terms are the 'timbrel vault', the 'Catalan vault', the 'Guastavino vault' and the 'thin-tile vault'. In this research the thin-tile vault is used as the structure. However, it is necessary to provide the relation between this term and the other three. Therefore, a review of the literature will show each definition.

The typology of the structure is the thin-tile vault and this defines some geometric and structural properties. The geometrical properties narrow down the scope of the research. This results in a type of the thin-tile vault, where a set of parameters creates design variations.

The building envelope is of little use for the construction phase. The construction of a masonry structure involves the placement of elements (bricks) with the help of an adhesive. Therefore, it is important to know the position and orientation of the bricks of the thin-tile vault. Furthermore, the order of construction of these bricks is required. Although the design plays a trivial role in this determination, a proper set-up of the design definition of these bricks and the thin-tile vault eases the determining process in other stages.

The typology of the structure is the thin-tile vault and this defines some geometric and structural properties. The structural properties of the thin-tile vault justify the used structural analysis. The major interest is the structural working of the thin-tile vault compared to traditional masonry (vaults). These properties of the thin-tile vault result in a loading scheme for the element to withstand.

In a time-independent structure, as is considered with any structural analysis on the structure after construction, the maximum stresses of a vault can be found at the base of the structure, where the normal force is at its maximum. However, this structure is time-dependent. This results in a different structural scheme as long as the structure does not encompass the thrust line of the vault. Additionally, the adhesive will go through a change of its properties. Before hardening it can be moulded, while after hardening it is able to withstand stresses. This process takes time. This means not only will the maximum occurring stresses be of importance, but also those positions where the strength has not yet developed properly.

During construction the structural scheme of the thin-tile vault is different than after completion. The most important property of vaults is the lack or insignificant amount of bending in the structure. This property is not present until a section of the structure spans the vault. Thus, bending is present in the structure. This results in different requirements for the materials in masonry. Even a different material may be required.

Some materials show a change in their properties during construction. Other materials remain the same after construction as they were before construction. Timber and steel elements are examples of this last category. (In-situ) concrete and paint are of the first. In masonry, the properties of bricks remain unchanged, while the mortar hardens, changing from moldable to strength resisting. This means that as the construction progresses, the material properties change. This has to be taken into account when choosing the appropriate materials. When how much the structure can be loaded at which locations determines the size of the structure and

the speed of construction.

A wide range of robots exist. However, not all robots will be suited as construction robot. The robot should be able to perform tasks similar to a mason. For some tasks assistance would be the better option, cooperating with robots. Furthermore, the usability and limitations of the robots is required as well. A look into the (Dutch) construction industry may reduce the number of possible robots even further. This should lead to a (small or limited) stock of robots where from the robots are chosen that are used in this research.

The design process is mostly done digitally nowadays. From basic (technical) drawings and sketches to complete 3D-, or even 5D-models, the model is on the computer. This is no different for this research. The design model and engineering model are both in a digital environment. The same applies to the model for the robotic construction simulation. However, both operate differently. The first two models are done from the principles of CAD, while the robotics is done from CAM. CAD is Computer-Aided Design and has resulted in structural analysis and design software. Here input from reality and designers or engineers create a design model and results of the structural analysis. CAM is Computer-Aided Manufacturing and uses as input a design model or work station and outputs a code to operate the robot with. The translation from CAD to CAM may be as simple as using the design model for CAM-software, or it may require a specific output.

2.2. Research question

Based on the objectives described in the previous section, the following question can be stated.

How does a robotic construction of a parametrically designed thin-tile vault perform based on step-wise structural analyses?

This results in the following sub-questions:

1. What is the parametric model of a thin-tile vault?
2. What stresses (can) develop in the construction of the thin-tile vault?
3. How does a robot construct a thin-tile vault?

These sub-questions are reflected in the outline of this report, with each its own part.

2.3. Methodology

The question requires a method of three major steps. The first step is to create a design model of the thin-tile vault. The second step is to calculate the stresses occurring within the structure during construction, which is done in an engineering model. The last step is to use the design model and the results from the engineering model to provide a simulation of the construction by a robot.

2.3.1. Design model

The design model needs to provide the information for the engineering and robotics model. To produce the design model, three major steps have to be undertaken, as described in the previous section. The first step is to provide literature which defines the thin-tile vault. The second step is to find the characteristics of the thin-tile vault and define these in the design model. The last step is to define the bricks in the thin-tile vault.

The first step places this research within context. As the introduction already provides, a starting point is the research done at both MIT and ETH. The first is to define the thin-tile vault in historic context, while the second provides starting conditions which aligns with recent research.

With the definition of the thin-tile vault the main part of the design model can be made. The design of the thin-tile model results in a vault made of a brick pattern. Thus, the first step is to define a vault surface of zero thickness. With this base surface, the bricks can be positioned on this surface. Masonry consists of courses with the bricks in a certain bond. The model needs a definition of these courses and a definition of the bricks relative to each other based on the bond. The path of the courses needs to take the curvature into acocunt as well. With the courses and the bond defined, the vault with brick pattern is made.

The last step for the design model is to translate the global design of the vault to the required information from the bricks for the engineering and robotics models. This step can be integrated from the start with the other steps, to avoid redoing previous work. Two sets of information have to be retrieved from the design model and kept consistent for the other models. The plane of the bricks is needed to apply individual transformations on the bricks relative to the vault. For instance, the bricks have to come from a source before they're placed in the structure. This source needs the planes of the bricks to provide

proper placement. The second set of information is the position of the brick relative to others. Here position is not the spatial position, but in which course the brick is placed, or how bricks in other courses are related to each other.

2.3.2. Engineering model

The engineering model has four steps to look into. First, the characteristics of the thin-tile vault are applied from an engineering approach. Second, the applied characteristics result in the strength of the structure and where critical stresses may arise. The third step is to analyse which materials are suitable for the thin-tile vault. The last step is to implement the stresses and strengths with time as a variable.

The literature from ETH and MIT may also provide the structural characteristics of the thin-tile vault. Furthermore, the Eurocode provides a first estimation of the structural properties. Additional research and literature may be needed where the first two lack data.

With the structural characteristics of the thin-tile vault and additional information from literature, the approach for the structural analysis is made. First any simplication of vault should be applied where possible, which results in a structural model. Second, an analysis of this structural model is done to better understand the behaviour of geometric parameters on the structural performance. This results in a step-by-step description to get from the geometric shape of the structure to the stresses.

The literature of masonry in general is complemented with literature on suitable materials. The literature provides estimations on the behaviour of the material. From literature the strength development of the material is also gathered. Additionally, the material is tested in a small-scale experiment to verify the used values.

The stresses and strengths are calculated for the entire structure after the placement of each brick, or a similar time step. This is done by evaluating the cross-sections at a set of positions. This set has to be large enough to cover all important locations to test for strength and stresses. Additionally, this will limit the possible combinations in which bricks can be constructed. This results in the sequence in which the bricks are placed.

2.3.3. Robotics model

The robotics model translates the design model with the help of the engineering model to the information required for a robot. The required information is dependent on the robot used.

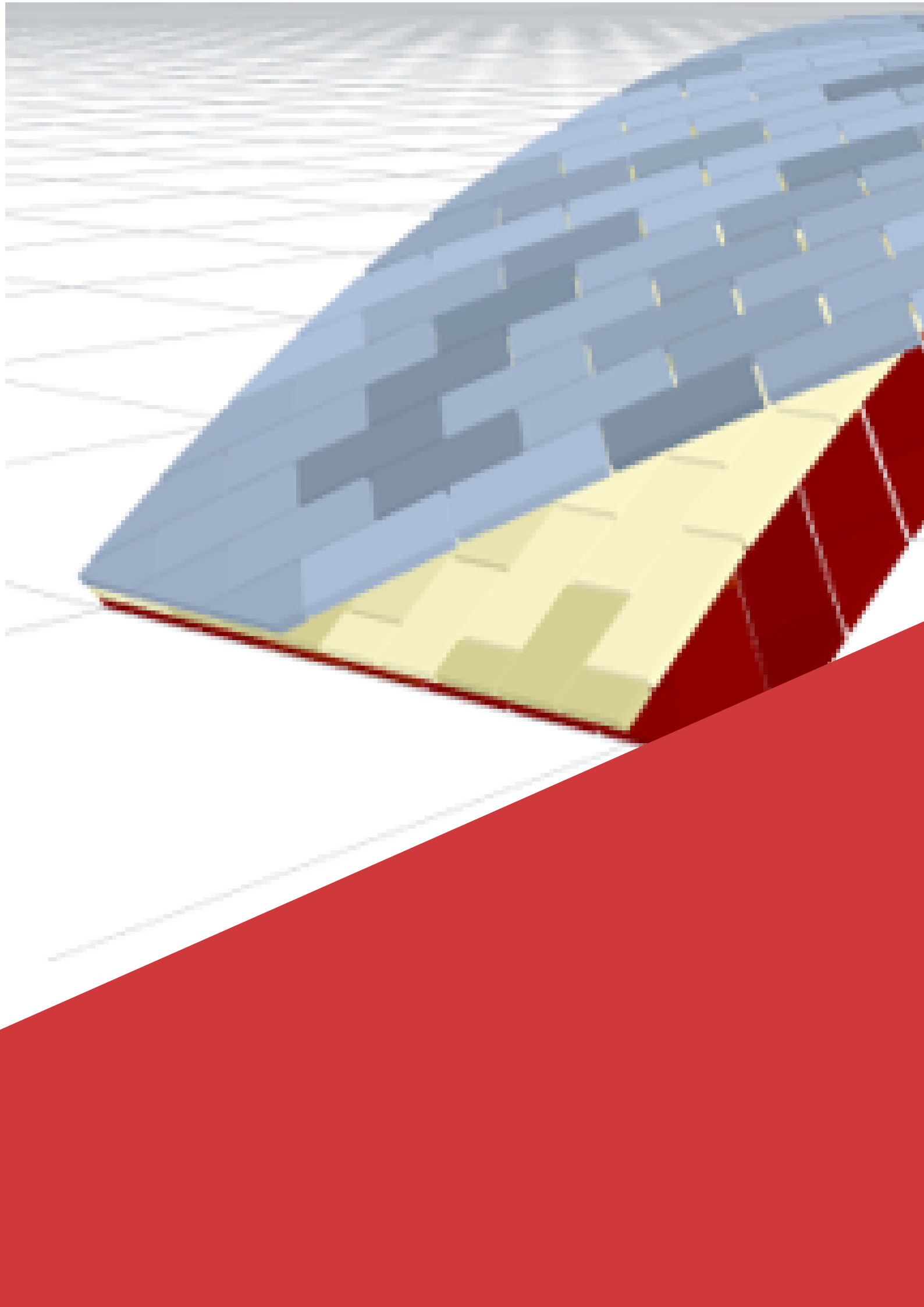
The possible availability of a physical robot should be pushed aside. That means all models stay within a simulated environment. Thus, the first step is to determined what software is capable of robotic simulation. Together with the software and literature, a stock of robots can be determined. Based on the specifications of the robots and the requirements for construction, the robots to be used can be determined.

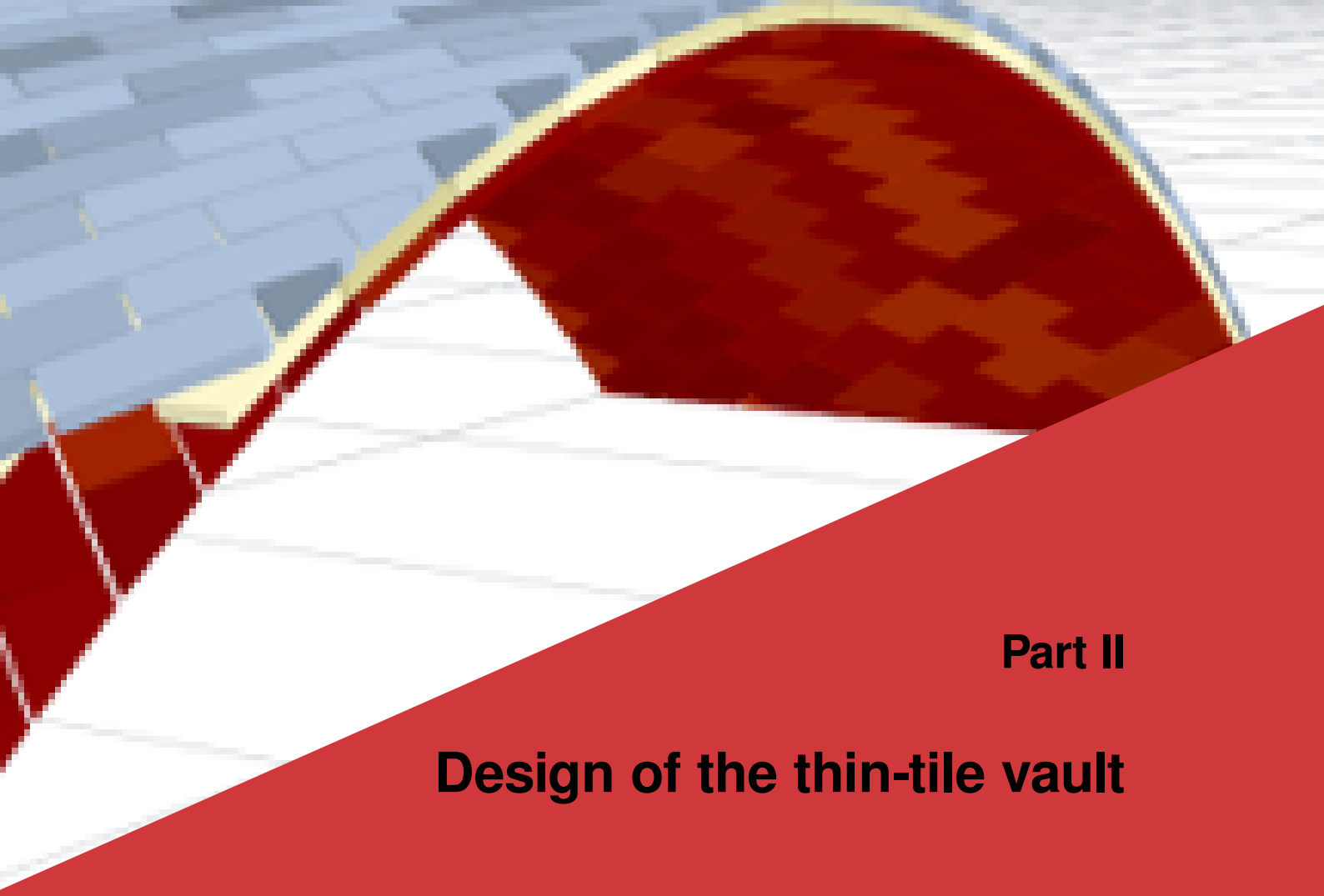
The software and the robot together define how the information between the design and engineering models and the robotic model are handled. The general path of the robot will be from a source where the bricks are placed, to the target in the vault. In between the adhesive has to be applied. This path is a set of actions that the robot has to perform, similar to actions taken by a mason. This path is then translated or converted to the format for either the software or the robot.

2.4. Software

The entire process relies heavily on software. In this research the main program used is Grasshopper 1 in Rhino 6. Additionally, the structural analysis is programmed in Jupyter Notebook (Python 3) connected with Grasshopper with a csv-file. RoboDK is the robotic simulation program. Again a csv-file connects the two programs. RoboDK comes with two Python files which will need some alteration. The first imports the csv-file into RoboDK, while the second one exports the simulation construction times into a csv-file for further analysis.

Beside these main programs, Maple and Excel are also used to provide mathematical analyses or any analysis on the csv-data, respectively. DIANA provides the possibility to model masonry to the latest understanding, which can be used in a generalization for further structural analyses.





Part II

Design of the thin-tile vault

The first part of the research is related to the structure known as the thin-tile vault. This type of structure has a lot of names, with which the exact definition diffuses. From literature the definition and the properties of the thin-tile vault should be found. The focus is primarily on the geometrical properties, but showcasing structural properties alongside these is no issue. With these definition and properties a design model can be made for further use in the structural analysis and robotic construction.

To get to that model this part is set up in two chapters. The first chapter provides the literature which is used to make and define the design model. The literature is ordered chronologically, starting with the Catalan vault, its origins and early understandings. The Guastavino vault describes both the work and understanding of the vaults made by Guastavino and his son. In the last section modern research into the thin-tile vault and free-form vaults is shown. Special attention is given to the works done at ETH Zürich and the development of *Rhinovault*.

The second chapter describes the design of the structure. It starts with showing the definitions used for the masonry structure, both from the perspective of the vault and from the unit elements. Next the goals and outputs of the parametric model are given in an overview, followed by a description of parametric modelling and the core principles used for the model definition. In the next section the principle of map projection shows how the core principles of the model definition can be used for the curved vault. The implementation of the model definition so far in an algorithmic and mathematical setting are shown in the section thereafter. In the last two sections attention is given to the edges of the vault and the visualisation of the model for user interpretation.

3

Literature on thin-tile vaults

The structure used in this research has many names: Catalan vault, timbrel vault, Guastavino vault, bóveda tabicada and thin-tile vault. This is no surprise considering the history of the thin-tile vault, as briefly described in section 1.1. In this chapter research previously done on this vault will be shown. For this the research and examples have been split based on three chronological periods. The first era is the work done in the Old World, in the time prior to the inventions of Rafael Guastavino. In the second time period, the work of Guastavino will be considered. The research of John Ochsendorf on Guastivino is also included in this section. The third section describes recent research related to the thin-tile vault. The main focus here will be research that made use of Rhinovault.

> Timbrel vaults are masonry vaults made with brick and mortar. Their uniqueness derives from their construction: the bricks are placed flatly, forming one or more layers and they are constructed without centering or other support. The bricks are placed in arches or successive rings to complete the vault [...], During construction, the bricks are supported by the adhesion of the fast-setting mortar to the completed courses, or to the bordering walls. There is no formwork, but guides are used to control the geometry of the vault,[...] (Huerta, 2003)

3.1. Catalan vault

Huerta, 2003 gives an extensive overview of the history of timbrel vaults in general and the method used for the structural analysis. In his paper, published in *Essay on the History of Mechanics* with the help of John Ochsendorf, he describes how the early application of the timbrel vault changed during the nineteenth century. The structure was used as the ceiling of churches, as flooring and as staircases. However, with the industrial revolution in Spain and France, the structure started being used as roof of industrial buildings as well. Based on his article, it seems the introduction of Portland cement widened the application of this structure: it could be used as a roof structure, without the need of a cover. This is different to its use in churches, where it only acted as ceiling, with the timber roof structure spanning over the vault. The timbrel vault became a symbol of Catalan culture, becoming vernacular architecture. Later on, the Catalan builder Rafael Guastavino would use this 'Catalan vault' as his legacy.

Huerta, 2003 goes on, stating that the Catalan vault has been used since at least the fourteenth century. One of the key sources from this first era is Fray Lorenzo and his publication in 1639. Lorenzo describes the necessity of lateral supports. In other words, the existence of thrust from the vault was known. This may not surprise modern engineers and scientists, but later on it becomes clear this aspect of the vault got lost.

The Catalan vault has also been applied in France. The Count of Espie liked the structure due to its fire-resistance and light weight, and would go on to do tests (Huerta, 2003). Espie believed the 'flat vault', as he called it, did not exert any thrust on the walls. His thinking is based on the adhesive in the bricks acting at such quality the structure is considered monolithic. He proofed this with his tests, but Huerta disputes this stating *"many of the tests can be made with normal masonry vaults with the same results"*. His work from 1754 was well received at the time and the Catalan vault became a structure known for its fire resistance and the absence of thrust. Later

publications by other writers cemented the idea and at the start of the nineteenth century Catalan vaults were thought to be absent of thrust, even back in Spain. Huerta does mention that in the Spanish translation of the Espie's book, the book starts with a comment from the architect of Madrid, Ventura Rodríguez, disagreeing entirely with the theory and mainly the observations of the Count of Espie.

In the nineteenth century sources become scarcer (Huerta, 2003). One experiment in France tests the loading capacity of the Catalan vault, using spans common for textile factories (3.75 and 4 metres). Besides the loading capacity, which turned out to be 2.700 and 1.250 kg/m^2 respectively, this experiment did include the thrust from the structure. A publication in 1841 from Spain describes in detail the construction of the Catalan vault for various typologies like staircases and domes. Interestingly, in the first part and in the projects described in the publication thrust is considered in the design. However, in the second part, the ideas of Espie are mentioned, contradicting the rest of the publication.

3.2. Guastavino vault

(Huerta, 2003) continues with Guastavino, maybe the most important practitioner of this structure. Rafael Guastavino was born in 1842 in Valencia. After his studies and constructing a couple of buildings with the Catalan vault, he moved to the United States in 1881 with his son Rafael Guastavino Jr. Convinced the Catalan vault was to be applied extensively, he would use this vault in his first big project: the Boston Public Library. Guastavino knew the main advantage of the timbrel vault: an easy to erect roof structure that is fire resistant as well. In 1889, he called his company the *Guastavino Fireproof Construction Company*. As Huerta mentions, American buildings at the time preferred timber and iron structures to hang fals vaults from, instead of heavy and cumbersome stone and masonry vaults. Having a light weight structure like the timbrel or Catalan vault was useful in the eyes of the American architects. However, since the structure was less known in the US, Guastavino had to proof the promised properties. As Huerta mentions, with Guastavino the timbrel vault was analysed in a scientific way for the first time.

Guastavino went on to promote his structure with seminars, magazine articles, two books and other articles and papers. In 2001, Huerta gave a bibliography of all of Guastavino's work, including coordinating the genesis of Parks on Guastavino's first book. This book, *Essay on the Theory and History of Cohesive Construction, applied especially to the timbrel arch* from 1892, is the main product on Guastavino's understanding of the *Guastavino vault*. The first distinction Guastavino made, was between construction by gravity and construction by cohesion. The construction by gravity creates a total mass in equilibrium, without considering the adhesion between the solids or elements. The construction by cohesion gets its properties based on the cohesion between materials, making it one material.

Guastavino tried to give his vault a historic background, which can be seen in the numerous buildings Guastavino considered being built as a construction by cohesion. However, Huerta points out that Guastavino may only have looked at structures with a good adhesion, including Roman concrete, instead of looking at the structural system. Huerta also explains why Guastavino compared a construction by cohesion to how nature makes "conglomerates", or monolithic structures. Guastavino was awed by a cave in Spain, especially how all elements of the structure, its walls, roof and floor, were all one element or solid. Something nature was able to make, but humans were not. With this perspective, he considered the construction by cohesion the natural way of construction. Although his motivation stemmed from a wrong understanding, Guastavino would go on to improve the cohesive property of the Guastavino vault, improving the vault construction as a whole.

Guastavino continues with his distinction by comparing two timbrel arches (Huerta, 2003). The first timbrel arch, representing the construction by gravity, is made of one layer of bricks. Clear joints are present between the bricks of the arch. These joints break up the whole cross-section of the arch. The second timbrel arch, representing construction by cohesion, is made of two layers. Besides the mortar between the two layers, the bricks overlap (similar to a stretcher bond). This means each joint between the bricks only takes up half the cross-section of the arch. With this cohesive structure, the arch should be able to resist bending. Huerta confirms that Guastavino has shown this property of the multi-layered timbrel arch. This second arch was built and within a few hours construction workers were able to walk on the vault. The dimensions Huerta provides, indeed show that

bending moments need to be taken in the vault. The vault had a span of 6 metres and a thickness of 7,5 centimetres. Assuming a heavy brick with a density of $2000kg/m^3$, this gives a distributed load of $1,5kN/m^2$. A couple of workers on top of the vault would give some significant point loads. It is unlikely an arch or a vault is shaped precisely on these varying point loads. Guastavino demonstrates this aspect of the multi-layered arch with a photo on which he is standing on top of one of these arches. See figure 1.1. In Guastavino's understanding, the construction by cohesion can be improved by lowering the amount of joints, with the ultimate goal to have a structure without any joints.

Huerta's list of structures Guastavino uses to give the Guastavino vault includes some made of Roman concrete, as mentioned earlier. Guastavino seemed to understand that an unreinforced concrete arch would be a construction by cohesion without any (vertical) joints. However, Guastavino considered concrete too costly and too unpredictable in its setting to be investigated further. It is, ironically, the advancements in steel and concrete that will be the downfall of the timbrel vault (Dugum, 2013).

Huerta continues with the tests Guastavino performed on specimens representing the timbrel arch. See figure 3.1. Huerta provides a summary of the results, which can be seen in table 3.1. He also comments the lack of any material property like the Young's modulus.

Guastavino was able to calculate the thrust of flat vaults by formula 3.1 (The formula, already changed in notation by Huerta, has been adapted to notations common at the TU Delft). Sadly, Huerta failed to note that Guastavino's formula does not result in the same unit on both sides (Guastavino, 1893). The left hand side results in the thrust, or horizontal force (per unit length) in the arch, while the right hand side gives a distributed load (per unit length). Huerta even provides an example calculation from Guastavino. An obvious correction would be to square the span, which would give the thrust as the moment at midspan over the rise at midspan. However, applying equation 3.1 with this correction does not result in the same values as (Huerta, 2003) provides.

$$A \cdot \sigma_{br} = \frac{q \cdot l}{8 \cdot f} \tag{3.1}$$

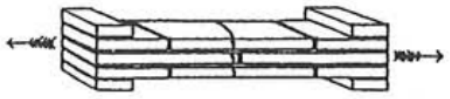
where:

A = cross-sectional area of the vault at the crown per unit length;
σ_{br} = breaking stress in compression;
q = total load (self-weight + live load) per unit length;
l = span of the vault;
f = rise of the vault.

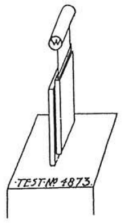
Nonetheless, Guastavino had a second formula to calculate the required thickness at the support. Huerta therefore concludes Guastavino used an equilibrium analysis to provide a varying thickness of the vault. The thrust would be compensated by either buttresses or iron ties. At the end of his book, *Essay [...] timbrel arch*, Guastavino provides a table of elastic stresses (Huerta, 2003). These were calculated by a professor of applied mechanics from MIT.

Table 3.1: Mean strength of timbrel specimens as presented by Guastavino (1893)

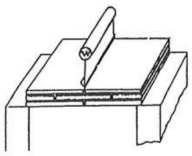
Property	Strength [N/mm²]
Compression	14,60
Tension	2,00
Shear	0,90



(a) Tension;



(b) Shear;



(c) Bending;

Figure 3.1: Specimens for the strength tests made by Guastavino, (Huerta, 2003, Guastavino, 1893 from)

Huerta states the difference between the elastic theory and the calculations by Guastavino to be

insignificant. Moving on to domes, Guastavino is inspired by works of others to determine the thrust. However, as Huerta states, the assumed thrust is a conservative value. Guastavino is aware his thrust values were higher than required, but safer as well. All things considered, Huerta states, based on the *Essay*, that Guastavino had a deep understanding of the Guastavino vault, but lacked theoretical knowledge.

Huerta concludes his reflection on Guastavino by saying that although his scientific contribution isn't large, it is clear he is a master vault builder. Guastavino designed and built by the geometry, and not by the material. He understood the theory on equilibrium and was able to apply graphic statics in an extensive way. Huerta shows in his writing a deep appreciation for the skill with which Guastavino was able to construct vaults and apply reinforcements on domes, based on the designs from the builder kept in the Avery Library.

His son, Rafael Guastavino Jr. (Rafael), continued his fathers company. Raised and educated by the master himself, he was a skilled master builder as well. Similar to how Guastavino used the modern theories at the time to verify his structure, Rafael continued including the latest calculation methods in his designs. After the development of the elastic theory in the late 1800's, the twentieth century would start of with (scientific) developments in the membrane theory (Huerta, 2003). Rafael would apply this theory to improve the design and construction of (un)reinforced masonry domes (and shells). As the twentieth century progresses, Rafael battled the upcoming concrete construction fiercely with more research in different coloured bricks and acoustics. Of Rafael only his patents, seminars and buildings remain. He never published papers.

3.2.1. Correcting the monolithic or cohesive construction theory

In Huerta, 2003 the error in thinking from Espie and the Guastavino's is related to the elastic theory. Basically, Guastavino was right to assume a timbrel vault behaved differently than more traditional vaults and arches, like arches made of voussoirs. However, this difference in behaviour did not result in a different or lack of thrust. Huerta uses the publications of Domenech, Martorell and Bayó (all early 20th century) to show the implementation of the timbrel arch in elastic theory. The first two include the bending moment in the calculation. To them, the design of a masonry vault should be similar as to the line of thrust, found using elastic theory. The timbrel vault can, miraculously, deviate from this line of thrust. Martorell even notes "as if they were metallic shells" (Huerta, 2003).

In the end, Bayó shows a true understanding of the structural behaviour using elastic theory. He compares the timbrel vault to steel arches. With this comparison he is able to explain why the timbrel arch behaves different than a traditional voussoir arch. The traditional arch can only resist compression, following the line of thrust closely in its geometry. The timbrel arch has some capacity to take tensile forces, resulting in a possible deviation from the line of thrust. This is similar to the distinction Guastavino made with his construction by gravity and by cohesion. However, unlike Guastavino, this explanation does not neglect that in fact the structure is an arch, of which its geometry is essential to resist the applied loads, including a thrust. Although Huerta leaves it at that, the comparison with a traditional arch can be approached differently as well. Masonry (and other brittle, or stony materials) is generally calculated as having no tensile stress resistance at all (Hartsuijker & Welleman, 2007). In other words, the stresses in the entire cross-section are of the same sign, usually the one assigned to compression. To get to this result with any line of thrust, this line has to go through a certain area of the cross-section. In more conventional terms: the centre of force has to be within the core of the cross-section to result in stresses all of the same sign (Hartsuijker & Welleman, 2007). See figure 3.2a for this area of a traditional barrel vault. One property of this core is when the centre of force is right on the boundary of the core. When this occurs, the neutral axis is exactly on the corresponding edge of the cross-section (on the opposing side to its normal-force centre). The neutral axis here represents the axis in a cross-section where on each side a different sign exists. Thus, starting from this boundary, moving the centre of force in the core pushes the neutral axis away from the cross-section. In the opposite direction, moving the centre of force outside of the core results in a neutral axis inside the core, with compression on one side and tension on the other.

As established, the timbrel vault has a higher resistance to tensile stresses than traditional vaults. Thus, the general assumption to need a cross-section with stresses of only one sign, cannot go for the timbrel vault. This is its essential difference to traditional arches. Consequently, the centre of force can go outside of the core. Because of this (small) tensile capacity, a new

type of core can be constructed. As long as the centre of force is within this new type of core, the timbrel vault can take the (minor) tensile stresses. This is shown in figure 3.2b. Using this approach gives a clear and intuitive way to look at the design possibilities of timbrel vaults.

With the contribution of Bayó a full explanation of the structural behaviour of the timbrel vault is related to the elastic theory. The no-tension model of Heyman is not applicable to the Guastavino vault (Como, 2017). However, in examples to calculate the vault with elasticity, Bayó leaves it at a barrel vault (Huerta, 2003). Later contributions also struggle to implement the elastic theory on the more complex structures, which builders, like the Guastavino's, seem to be able to construct with little effort. Terradas (after 1919) tried the famous staircases, Goday (1934) disliked using membrane analyses, and Torroja (1956) was awed at what builders could make and engineers were incapable of calculating. The lack of scientifically proven engineering calculations even existed in 1999, with J.L. González, Professor Architectural Technology of the Polytechnic University of Catalonia, considering empirically deriving the strength of a timbrel vault staircase the best approach (Huerta, 2003).

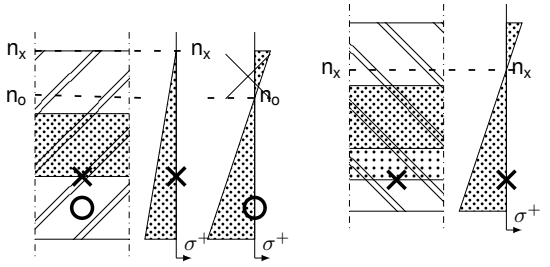
Huerta provides some additional builders and engineers that tried to tackle the timbrel vault. Moya (1957) states the insufficient calculations from the elastic theory are due to a lack of material properties of the composite. Pereda (1951) uses a similar approach as described earlier, keeping the line of thrust within the middle third of the section. Huerta also shows the timbrel vault has been analysed with the Finite Element Method (FEM) by Gulli (1990's). However, he argues whether even a non-linear analysis improves the calculations significantly compared to the elastic analysis. It is, however, unclear from Huerta, 2003 how advanced the FEM-calculations have been. In the past decades FEM has been improving, with DIANA FEA being a prime example (DIANA, 2008). Of special interest is the development of interface elements to model the likely crack formation of solids.

The use of the equilibrium method, or analyses like the one Pereda used, gives a Limit Analysis that is guaranteed safe (Huerta, 2003). The condition that as long as the masonry is in compression, it is safe. However, this analysis applies to the end-product, to the finished structure. It is during construction the advantages of the additional tensile stress resistance of the timbrel vault are apparent.

3.3. Thin-tile vaults and computational modelling

The Guastavino Fireproof Construction Company ceased to exist in 1962, marking the end of masonry vaulting with thin bricks. With it, most of its knowledge was gone as well. Only few knew or remembered what the company did. In recent decades that became crucial to rediscover the technique of thin-tile vaulting. With the help of these people the basic principles and understandings of thin-tile vaulting were not lost. John Ochsendorf at MIT started to research the structures made by the Guastavino's. Particularly interesting to him was the performance of the structures after decades, especially considering their maintenance. At universities around the Mediterranean interest rose as well (Benfratello et al., 2012; Michael H. Ramage, 2004). Here again the focus on the maintenance of existing structures was the driving force.

Ochsendorf continued his studies, where eventually he would supervise the PhD of Philippe Block (Block Research Group, n.d.). Philippe is now professor at the ETH Zürich and the Block Research Group (BRG) has been focussing on the interface of both fields of architecture and structural engineering. The ETH is known, among other things, for its research into digital fabrication, with the NCCR as the main example of this. The Block Research Group has analysed existing masonry structures and applied modern



(a) Traditional masonry; **(b)** Thin-tile masonry;

Figure 3.2: The line of thrust has to go through the core (kern) of the cross-section, otherwise both tensile and compressive forces are present in the structure. In thin-tile vaults the defiance from the line of thrust can be larger. In the figures are shown a cut out of the cross-section with the core cross-dotted; valid (x) and invalid (o) lines of thrust with their respective stress diagrams and neutral axis (n-n); the crossed out stresses that cannot be taken by masonry; the 'extended core' (matrix-dotted) that allows a greater deviance of the centroid from the line of thrust. The valid lines of thrust in both figures are similar lines of thrust of similar geometric shape.

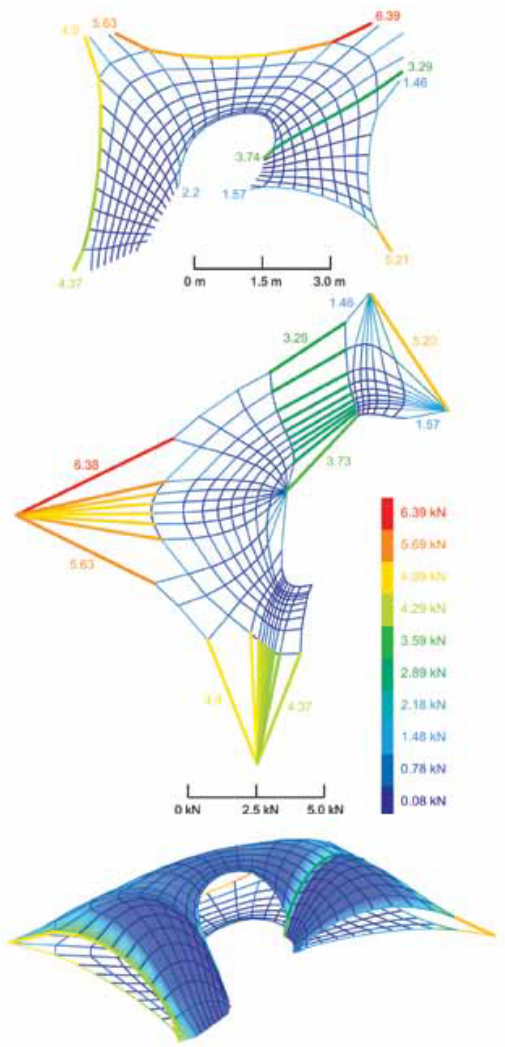


Figure 3.3: Thrust Network Analysis with Rhinovault. | (Davis et al., 2012)

structural analyses like Finite Element Modelling to better understand these structures. One of the key outcomes of the BRG is the further research into thin-tile vaulting. The PhD of Philippe Block introduces the Thrust Network Analysis (Block, 2009). This analysis is capable of analysing the shape of 3D, compression-only structures. The Thrust Network Analysis uses the gravitational loading of a vaulted surface and presents possible funicular solutions. This analysis has allowed the BRG to research existing masonry structures like Gothic vaults and develop and construct new thin-tile vaults. The Thrust Network Analysis has been developed into a user-friendly tool within the Grasshopper-Rhino environment: Rhinovault[(2)].

In general, the Block Research Group has focused on the following fields (Block Research Group, n.d.):

- Analysis of masonry structures
- Graphical analysis and design methods
- Computational form finding and optimisation
- Design of discrete assemblies
- Fabrication and construction systems

Their analysis on masonry structures mostly concerns existing (Gothic) structures, while their recent contributions in fabrication and construction systems has been primarily on the development of HiLo, which uses rib-stiffened funicular floor systems (Nervi is the historic example of this practice) and a concrete shell roof with the formwork or mould existing of a lightweight cable-net with fabric.

In recent years the BRG has assisted in various projects to show the potential of the thin-tile vault and the advancements made in their 3D free-form design of them. Some projects used the same modelling and design/structural analysis, but lacked the thin-tile vaults, using voussoirs instead. The temporary pavilion at the ETH Zürich campus showed the loading the structure can take (Davis et al., 2012; Philippe Block & Matthias Rippmann, 2013). The estimated labour for bricklaying was set at 21 days for 2 masons, 8 hours per day, with a total of 2.300 bricks. That results in 110 bricks per day per 2 masons (it is unclear how much cooperation is required of the two masons to achieve such a speed). Additionally, destroying the structure was a tougher challenge than predicted. The construction of the free-form vault was aided by the use of cardboard, both as guide line and to minimize the cantilevering length of such a structure. Further advancements with the thin-tile vault are proposed in streamlining the design, engineering and construction process. The Thrust Network Analysis form and force diagrams should be coupled with tiling patterns and thereby also the construction sequence of the bricks. Another project was 'Bricktopia' in Barcelona (López López et al., 2014). This pavilion was partly constructed with the help of cardboard and partly with the use of a steel rod net. The steel rod net replaced the cardboard to ensure a work space for the masons.

The use of (thin-tile) masonry vaulting has also been a topic for poverished or underdeveloped regions, where steel and concrete may be hard to come by, while labour and clay are in abundance. One example is the Droneport by Foster & partners, see figure 1. The United Nations has drafted a report on the construction of thin-tile vaulting, in part with the help of the BRG (Blanco, 2014).

4

The workflow of the design model of a thin-tile vault

4.1. Masonry definitions

Important for a model is to improve communication by defining the different parts of the structure. The definitions used for the timbrel vault are based on an existing and more common structure: the masonry wall. A brick wall has been used time and time again in the past and definitions for this structure have been developed throughout time. An important identical aspect of both the vault and a wall is the size of the dimensions. The height and width of a wall are of a larger magnitude than its depth. This means that both structures can be represented by a surface, thickened by its bricks. Thus, when defining the masonry definitions, only a part of this surface is enough to illustrate the necessary definitions.

Continuing with the wall as similar structure, common definitions from this structure can be found as well for the definitions. The first definition is to define the layers of the vault. In brick walls a wythe is like these layers: it spans the entire surface and has a depth of one brick. Brick walls usually have one wythe, or two when talking about a cavity wall. In figure 4.1 a brick wall of three wythes can be seen. With the thin-tile vault the number of wythes can be greater than two, but the principle is the same.

The next definition is to define the second type of layers. A brick wall consists of layers with bricks laid back to back. These layers are also called courses. Between courses mortar may be used to increase the cohesion between bricks. This mortar between courses is called a bed. The bricks within one course are also seperated by a layer of mortar. This mortar is called a perpend. See figure 4.2 as well.

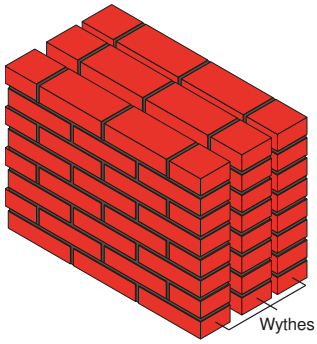


Figure 4.1: Wythes creating a brick wall of multiple layers.

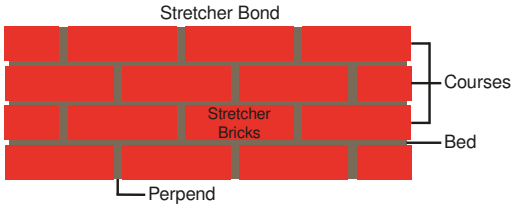


Figure 4.2: Masonry definitions in a brick wall.

Figure 4.3: Definition of brick faces, with Waalformaat (left) and a thin brick (right).

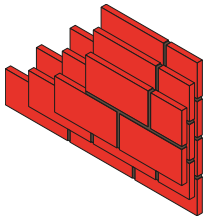


Figure 4.4: The different orientation of thin bricks in wythes.

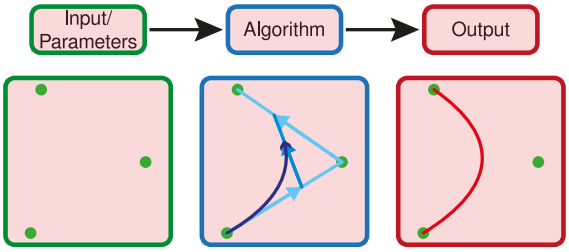


Figure 4.5: The main principle of parametric modelling: to make a design with the help of an algorithm.

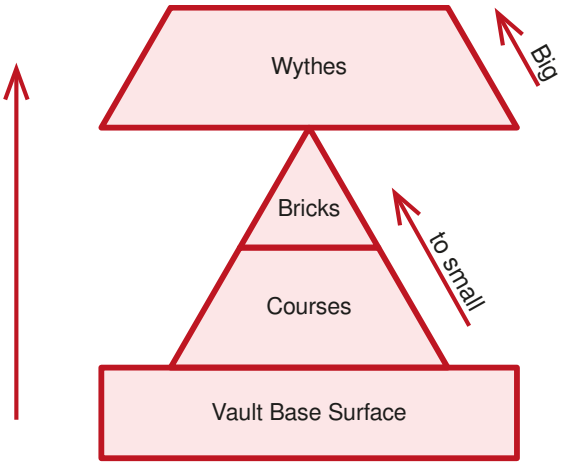


Figure 4.6: The basic sequence of making a brick pattern: first the courses, last the wythes.

Bricks are laid in certain patterns, also known as bonds. The most basic bond is called the stretcher bond (*halfsteenverband*). This bond is easily described as a pattern in which the alternating courses have an offset of half the length of the bricks. The faces of the bricks along this length are called the stretchers. The headers are the faces with which the bricks within a course meet each other. The faces the bricks have with other courses are called the beds. So, when laying bricks, a new brick is placed on the beds of the previous course, and pushed against the header of its predecessor. When one wythe or the structure is done, the stretchers are still visible.

There is one more type of brick-laying that has a common definition: the way bricks can be laid within one course. Here, bricks can be classified as either header bricks or stretcher bricks. Stretcher bricks have been explained before, these are laid with the headers touching and the stretcher visible. With header bricks this is reversed. The stretchers are touching and the headers are visible. However, these header bricks are more common as connection between wythes, as can be found in Flemish bonds. Connections between wythes are not a requirement for the thin-tile vault. Thus, it will not be further used in this research.

With these definitions the design of the vault model can be clear in further descriptions. It should be noted though, that if the faces of thin-tile bricks are the same in a structure, the faces based on a bricks dimensions are different (or vice-versa). With the Stretcher bond in conventional masonry, the stretcher bricks are the visible face (figure 4.2). The visible faces for thin-tile vaults are their biggest surface area, see figures 4.4 and 1. Thus, if the faces of the brick are based on the bond used, the largest surface area, normally called 'bed', is now the stretcher face, while the middle sized one is now the bed face. Defining the faces based on the bond results in a difference with both the presentation in figure 4.3 and the assumed fabrication of thin bricks, see section 5.2.

4.2. Basic outline of the design model

The design model has to accomplish two outputs:

1. A parametric model with which a visual representation of the structure can be made;
2. A list of positions, orientations, volumes and quantities of the bricks to be placed.

The goal of the parametric model is to automate a change in design, which enables a multitude of (slightly) different designs to be analysed based on the same principles and models, including possible errors and margins. The visual representation also reduces the need for (technical) drawings of each design. This gives a greater focus on the engineering and robotic simulation.

The list of all the brick properties are necessary for the translation to a virtual construction. It stands to reason that with robotic construction the need for a full scale model is high. Contrary to the mason given guidance by a rectangle and some text on a technical drawing, a robot needs a more in depth instruction, for it cannot analyse nor reflect. Besides the construction part of information, this process can include digital fabrication as well, as part of a BIM-approach. This integral design requires the information upfront, of which the quantity of each brick volume is a necessity.

These two outputs are dependent on a few parameters. First of all the dimensions of both the structure and the material are required. This means the size of uncut bricks and the base surface that represents the final design of the vault. Besides these lengths, the number of wythes and their angle relative to each other are important.

These inputs and outputs are used as the basis for the parametric model.

4.3. Parametric modelling

The main principle of parametric engineering is to create algorithms for a CAD environment. An example of how these algorithms create a design can be found in figure 4.5. In this figure a collection of data, in this case the point coordinates, is used to create a curve, seen in the frame for output. The algorithm is a set of operations to describe the curve with these points, in this case as a Bézier curve. The algorithm, thus, is a means to get to a design, applying the same operations on a data set. With this, design variations can be created using the same algorithm, but with a different data set, like parameters. As can be seen in the figure, the data set used as input is also called parameters.

Another important factor is how an algorithm like the one for a Bézier curve defines that curve. In this case, the algorithm travels from point to point to use these as the basis for the curve. In other words: the curve has a start and an end, which can be unitized (also known as reparametrized). This aspect, together with the main principle, is useful to further develop a design model for the thin-tile vault as a parametric model.

Components make it more convenient to produce an algorithm. These components are some common sets of operations themselves. Software for parametric modelling has a bunch of predefined components, but it is possible to make your own or add a set of components from a extensions or plug-ins. Appendix A provides a more in depth overview of the components used and if their from an extension.

From section 4.2 the inputs and the required outputs for the algorithm have been set. The design approach of the model is to create the courses of the bricks first, and afterwards place the bricks within the course. The last step is to create the wythes, possible in two ways: either creating new base surfaces for these wythes, or moving the bricks in each wythe with a unique offset per wythe. This design procedure for the algorithm can be found in figure 4.6. The base surface is the design definition of the vault. Except for the thickness, it describes the vault and needs to be 'filled' with bricks.

The first step is to define the courses along the vault. As figure 4.2 shows, a course can be described as a line on which bricks are placed at a certain interval. This means that the base surface has curves running over itself, separated by the width of the courses. This concept can be seen in figure 4.7 for a flat, free-form surface.

The bricks in each course can be defined in two ways in the parametric model: an outline approach and a centre point approach. The outline approach uses the lines of the courses as the outline for the bricks. By adding perpendicular lines between these course lines,

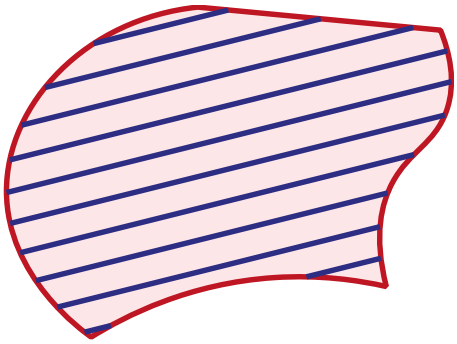


Figure 4.7: Courses represented as lines, equally distanced over a (random) flat surface.

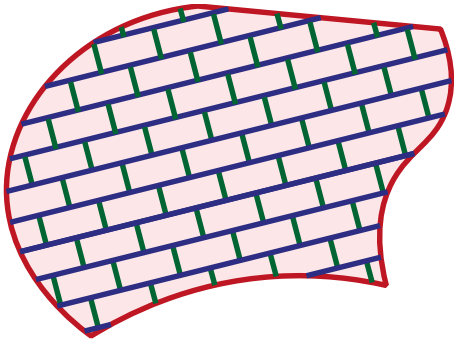


Figure 4.8: The outline approach with the course edges in blue and the remaining outlines of the bricks perpendicular to these in green.

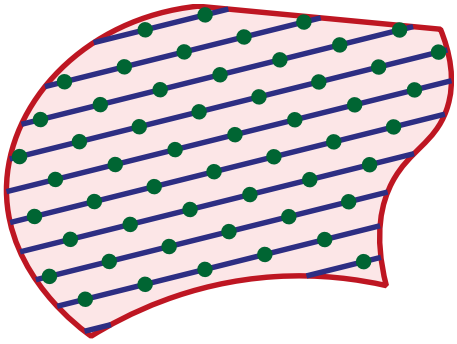


Figure 4.9: The centre point approach with the center of the courses in blue and the center of each brick in green. The resulting outline of bricks is not shown for clarity.

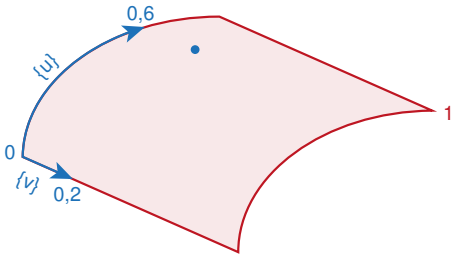


Figure 4.10: A point on the surface is relative to the curves.

the outline of the bricks are defined. This approach has the advantage to visually show the process in the algorithm displayed on the base surface. This may help with adjusting the brick patter design, without the need to fully develop it. A disadvantage is to align the course lines correctly to ensure the perpendicular lines are indeed perpendicular. Otherwise, these outlines will not fit with bricks shaped like a thickened rectangle.

Where the outline approach places the bricks between the course lines, the centre point approach places the bricks centred on these lines. With this approach the course lines are changed into points along this line. These points are the centre points of the bricks surfaces. The advantage here is the independence of each course line. A downside is that this approach requires more information for the process: the distance between the course lines, the position of the points, the orientation and direction of the surfaces and, again, the distances for the size of the bricks.

4.3.1. Using extensions to simplify the parametric definition

By using extensions with the software the parametric model, and consequently the modelling required for this research, can be simplified. After all, 'we're standing on the shoulders of giants'. For instance, a parametric definition for a masonry wall is available in an extension on the software. However, these definitions and extensions need to be investigated properly and to be understood by the users. In current extensions available, their application is limited to walls where courses run along the primary (non-zero) curvature. The parametric model in this study has the ability to place the courses not aligned with primary curvatures. Maybe future studies may be able to make a more simplified definition in their design model, based on a wider availability and application of extensions or software.

The definition of the vaults basic surface is possible with the components from the software. A catenary curve describes the vaults cross section, while the path of the vault is limited to a straight line. Although improving on the approaches mentioned in this section, would allow to use free-form curves instead of this straight line. These two curves allow the software to create a vault with a catenary shape as cross-section.

4.4. Draping bricks: a map projection approach

With the definition and the procedure of the brick positions, it is possible to place the bricks on a curved surface. An important drawback in this procedure is the implementation of various orientations of the bricks. When the courses are parallel to the cross-section of the vault, the definition of the courses is simple. However, to create courses in any other direction requires a more sophisticated definition. This can be seen in figure 4.10.

As this figure shows, a position on a (curved) surface is described with the relative position along its definition. In other words: the curves with which the vault are made, also describe the position on a vault. An interesting given with this fact is that if the procedure of the brick positions is made, this process can easily be scaled to other proportions.

If a flat or planar surface is considered, like in figure 4.7, it makes sense to see the courses as straight lines: each brick is in the same direction relative to the surface. This analogy can be expanded to curved surfaces as well. As long as the course has the same direction relative to the surface, it should uphold a tight and rightly distributed brick pattern. However, this does not mean that two courses next to each other will guarantee a correct orientation of the brick pattern between them. Thus, the outline approach from before is less suited for further use.

The next step is to define these courses given the relative positions to the vaults curves. For this the use of map projection can be used. Map projection is the problem where a 3D surface like a globe cannot be copied one on one to a 2D surface like a map. One property of the globe has to be deformed to allow for a correct projection. Figure 4.11 shows this as well.

A difference, however, is that for this design the reverse needs to be accomplished. Instead of projecting a globe on a flat surface, the flat brick pattern has to be projected on a curved surface. It

would be similar to how a cloth or wrapping paper is draped over an oddly shaped object. Thus, the procedure within the parametric model should be to describe the brick pattern as if it were on a flat surface and scale this pattern correctly to drape the surface.

4.5. Implementing projection in the parametric model

The parametric model makes use of the centre point approach in the following manner.

First the steps of bricks within a course in both u and v direction along the surface are calculated, using the orientation of the courses and the transformation matrix **R**. The next step is to use a starting position on the surface and calculate the position of each brick within a course, based on this step times the bricks number. This creates one course of bricks. Important to note is that in this process the course itself is never actually defined. However, only the positions of the bricks are grouped and it is these positions that can be used to create a line-like representation of the course.

$$\boldsymbol{\Delta_{brick}} = \begin{bmatrix} \frac{1}{2}\left(l_{unit} + 2 \cdot \frac{1}{2} \cdot h_{mortar}\right) \\ b_{unit} + 2 \cdot \frac{1}{2} \cdot h_{mortar} \end{bmatrix} \tag{4.1}$$

$$\boldsymbol{\Delta_{course}} = \begin{bmatrix} l_{unit} + 2 \cdot \frac{1}{2} \cdot h_{mortar} \\ 0 \end{bmatrix} \tag{4.2}$$

$$\boldsymbol{R} = \begin{bmatrix} \cos\alpha & \sin\alpha \\ -\sin\alpha & \cos\alpha \end{bmatrix} \tag{4.3}$$

$$\begin{bmatrix} u_{i,j} \\ v_{i,j} \end{bmatrix} = coursei \cdot \boldsymbol{R} \cdot \boldsymbol{\Delta_{course}} + brickj \cdot \boldsymbol{R} \cdot \boldsymbol{\Delta_{brick}} + \begin{bmatrix} u_{0,0} \\ v_{0,0} \end{bmatrix} \tag{4.4}$$

where:

l_{unit} = length of the brick;
b_{unit} = width of the bricks;
h_{mortar} = thickness of the adhesive or mortar beds;
α = angle orientation of the brick pattern.

To create the next course is to do a similar transformation as has been used to make the bricks in one course. The steps to the next course needs to be defined in both u and v directions. With these steps the starting position from the first course can be used to define the starting positions of all other courses. With all these starting positions, the bricks in all courses can be defined. See figure 4.13.

One problem is to make sure that these courses will cover the full surface. Thus, the first starting position needs to be outside the surface. u and v are negative! Afterwards, the number of bricks (steps within a course) and number of courses (steps along the course) need to be sufficient as well. For this the final step in both operations should result in positions where u and v are bigger than 1!

With different orientations of the brick patterns, a different number of courses and number of bricks per course need to be calculated. Again, the analogy with wrapping paper can be used. The same piece of wrapping paper can be big enough to wrap an object completely. However, with certain orientations, the same piece won't be enough. More paper is required. Similarly, with different orientations, more computational power may be needed, to calculate the higher amount of bricks.

However, this approach gives the position of bricks in a rectangular order, that can be differently orientated than the vaults surface. See figure 4.14. So, wrapping the vault completely with more brick positions, may not result in more

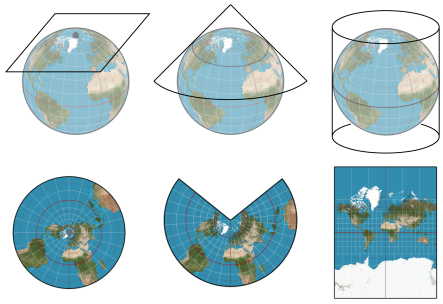


Figure 4.11: Map projections of a globe using planar surfaces as projection reference.

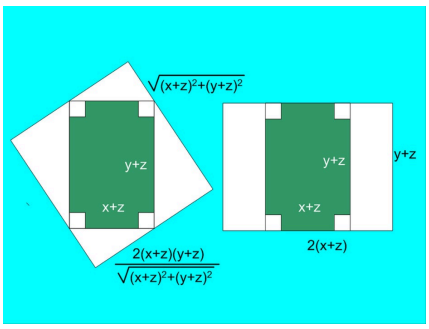
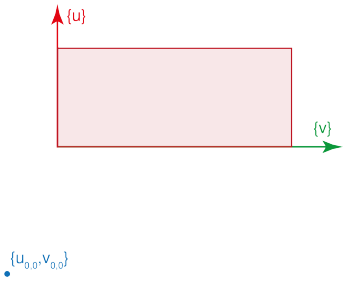
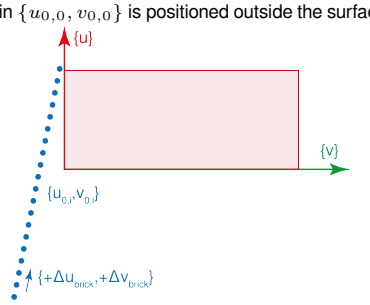


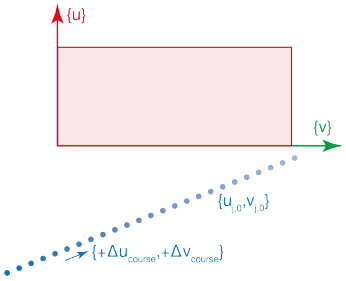
Figure 4.12: The orientation of the wrapping paper determines the minimal required size.



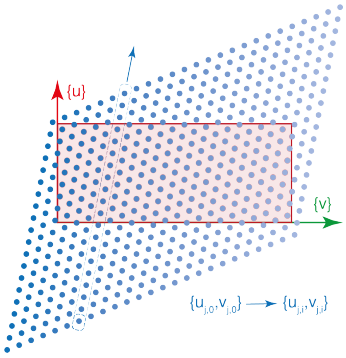
(a) The origin $\{u_{0,0}, v_{0,0}\}$ is positioned outside the surface domain;



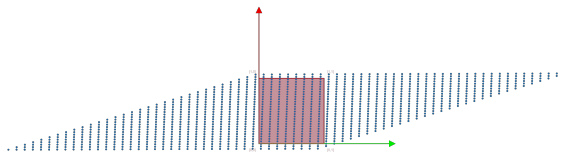
(b) The creation of bricks within a course is done with the unitized steps based on equation 4.4;



(c) After the origin point, the starting points of courses are created, where the last course still start outside the surface domain;



(d) The bricks per course are created (see the encircled course and its starting point);



(e) The implementation of this in the model on a unit surface, where the much longer vault arch length (compared to the vault course length) results in a significant downscale of the bricks unitized steps in u-direction, packing the centre points in u-direction;

Figure 4.13: Tiling the surface: the red (flat) unit surface is to be covered with the centre points, where u,v are the surface coordinates. Starting with the origin point, the starting point of each course is created. Afterwards the points in each course are created.

bricks needed to make the vault. It depends on the number of brick positions completely outside the vault surfaces boundaries (where u and v are negative or bigger than 1).

The model takes this into account by checking this relative position. With this all brick positions are set and these can be used to create the bricks surfaces. As mentioned earlier with the centre point approach, the orientation and the position are needed. With the relative coordinates u and v known, the local planar surfaces can be derived with the curvature of the vault at these coordinates. These planar surfaces are the planes on which the bricks are positioned, see figure 4.15. However, this does result in misalignment between the bricks surfaces. This is to be expected. After all, one property has to be skewed to give a correct map projection. This misalignment results in a varying thickness of the adhesive. As long as the deviation is within limits, the variation should not result in an unfortunate outcome.

4.5.1. Only barrel vaults

As the model stands it can only take vaults with a curvature in one direction. Although the approach already used gives a handle with which to expand for vaults with double curvatures, it has not been implemented. Dimensions, however, are not a limiting factor. This is dependent on the computational power.

The model also lacks a way to extract the volume of the mortar. The same goes for the misalignment of sequential bricks. Excessive rotation differences, and thus too big gaps (both perpends and beds) between the bricks may occur as of now.

4.6. Cutting corners

When all brick surfaces are created, some are cut to fit within the vault surface definition. This results in the second type of bricks: cut bricks. Again, this can be seen as similar to the wrapping paper: wrapping an object and cutting any material that is unnecessary.

This is done by two sets of information. The first set is the (outline of the) tiles. The second set is the base surface. The base surface is thickened perpendicular to its curvature to create a volume. This volume is intersected with the tile surfaces, which gives the cuts for the whole bricks. A volume is needed, since the tile surfaces are not in the plane of the base surface, but tangent.

It should be noted the tree containing the tile

surfaces has been grafted. This means each object has its own list. The tree has the same amount of lists as objects. Since not all objects interest the boundary of the volume described above, not all lists will have a cut. With the tiles each in a list in one tree and a corresponding list with or without a cut in another tree, these two trees can be combined again to cut the tile surfaces at the right place.

Now the data tree has lists with one (no cut happened) or more (the brick has been cut) items in them. From each list the correct surface has to be chosen: the surface within the volume made from the base surface. By using the centroid of each surface, this can checked and the unimportant surfaces removed from the tree. This cull includes surfaces entirely outside the base surface. This leaves only the whole bricks and the correct part of the cut bricks that are within the base surface boundaries.

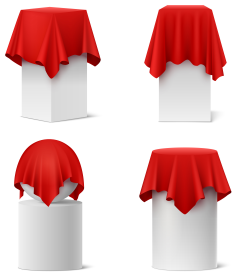
Meanwhile, the orientation of the planes of these surfaces have to be maintained. From the original tile surfaces, the plane is taken, see figure 4.15. Again, with the same information on which tile surface is inside or outside the base surface boundaries, the correct planes are chosen.

Now that some whole bricks have been cut, one last step is necessary. The brick pattern draped over the base vault surface can result in a brick tile surface being insignificant. In the parametric model, the largest brick surface area is chosen as the baseline. Any brick area smaller than 1% of this brick surface is removed. A smaller percentage could be chosen as well, this is dependent on the precision of the cutting machine or the qualities of the assisting mason.

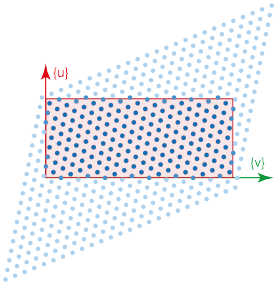
4.7. From surfaces to volumes representing the bricks

After the bricks are the correct shape and in the right place, the surfaces can be thickened or extruded. First, each wythe of bricks is moved outwards, in the normal direction of the base surface. Of course, the first wythe stays in place. This means that the base vault surface represents the bottom of the vault (model). The outward movement of the other wythes is the thickness of the brick tiles plus the thickness of the mortar joint.

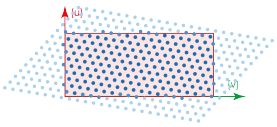
With the wythes in their correct place, the brick tile surfaces are extruded by their thickness. The next group of components allows for each wythe



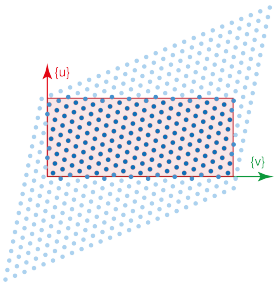
(a) The approach leads to a rectangular surface draped over the vault. Here shown as cloth draped over various objects. Note how the cloth exceeds the objects. | *macrovector* on freepik.com



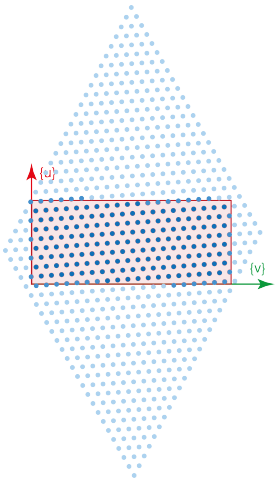
(b) 157 full bricks and 28 cut bricks: 185 in total;



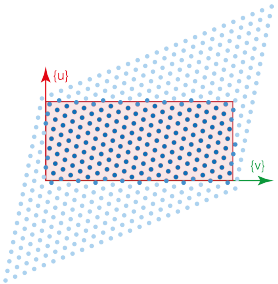
(c) 159 full bricks and 32 cut bricks: 191 in total;



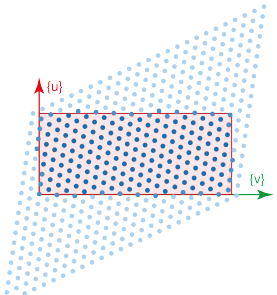
(d) 157 full bricks and 32 cut bricks: 189 in total;



(e) 156 full bricks and 34 cut bricks: 190 in total;



(f) 156 full bricks and 35 cut bricks: 191 in total;



(g) 157 full bricks and 33 cut bricks: 190 in total;

Figure 4.14: Different starting positions and orientations of the brick pattern. Each tiling has a similar amount of full or whole bricks, while the cut bricks can differ more, but here two half cuts could be gotten from one whole.

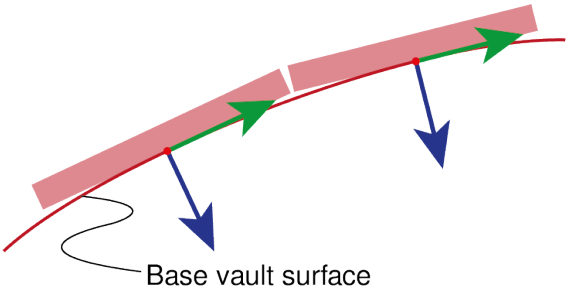
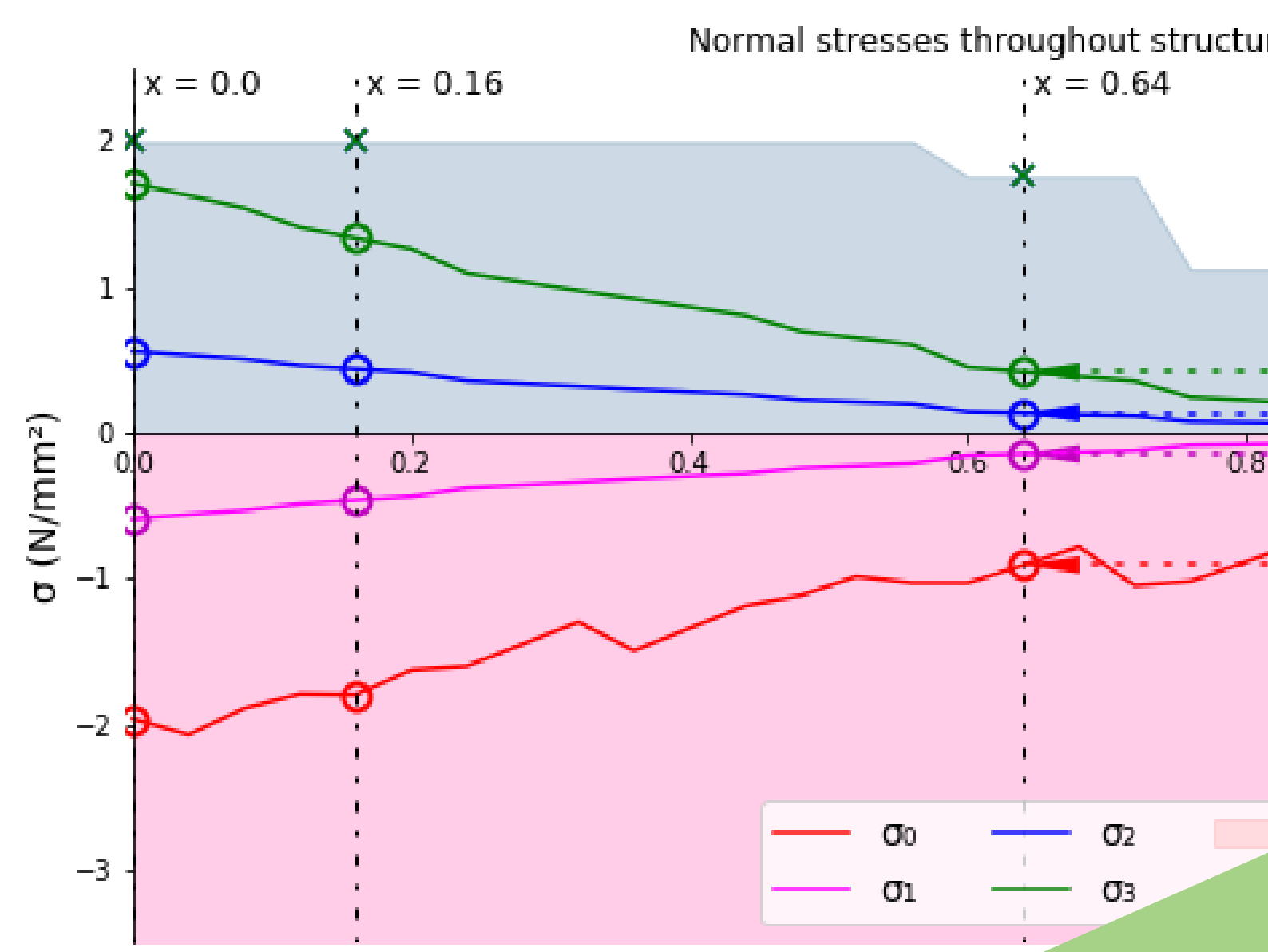
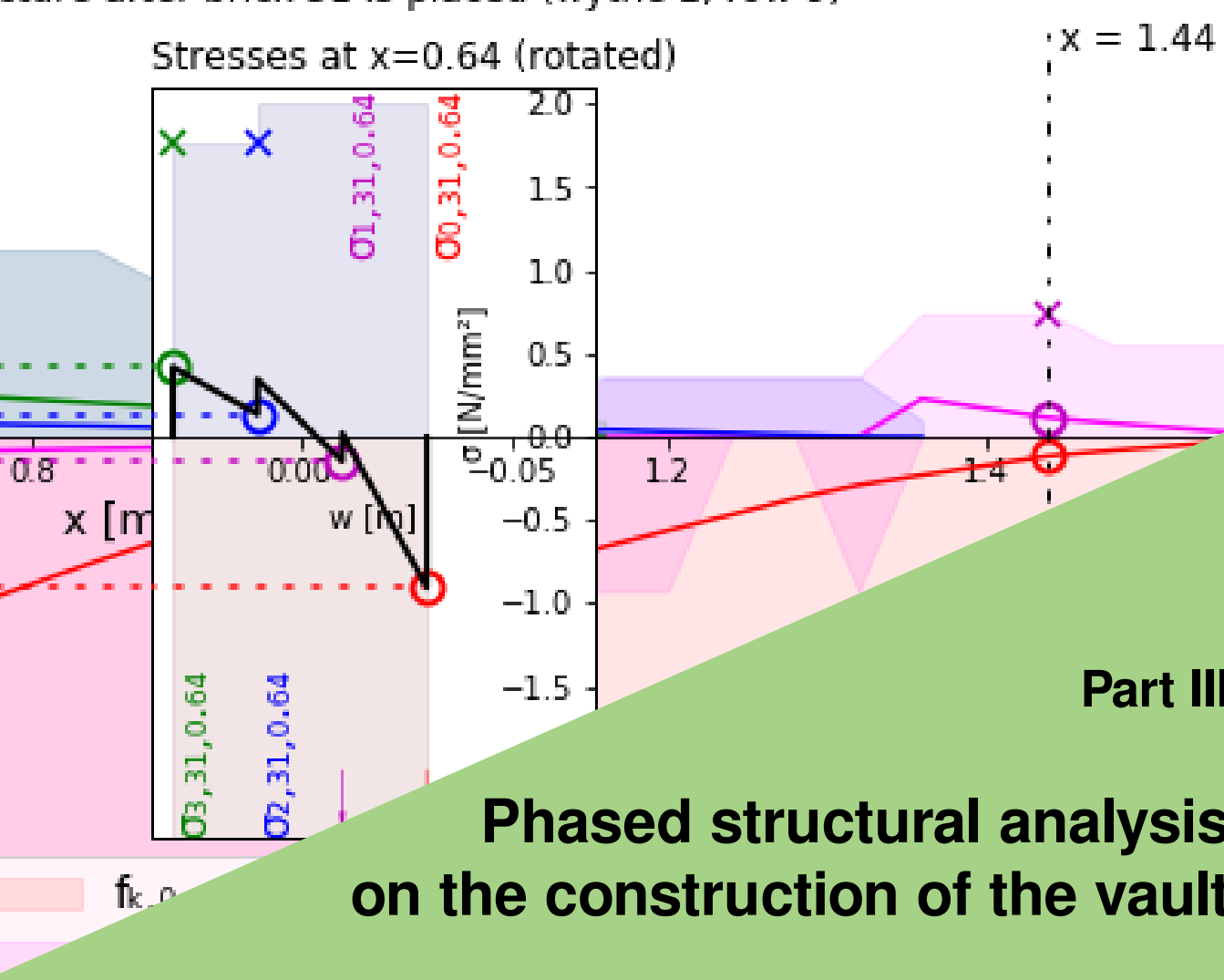


Figure 4.15: Plane of the centre points and in extension to that the brick. The gap width between the brick is smallest near the base surface and largest away from it.

to be of a different colour, creating an intuitive visualisation.


Normal stresses throughout structu
x = 0.0
x = 0.16
x = 0.64
2
1
0
−1
−2
−3
0.0
0.2
0.4
0.6
0.8
σ (N/mm²)
σ0
σ1
σ2
σ3



Part III

Phased structural analysis on the construction of the vault

The second part of the research is related to the structural analysis of the thin-tile vault. This structural analysis takes the time-component of construction into account. This is done with a phased structural analysis. Here the structural properties of masonry and the thin-tile vault are described in further detail with Part II as the starting point. In the end an engineering model provides input for the robotic construction and the extend to which thin-tile vaults can be constructed without using any additional support.

To get to that model this part is made of three chapters. The first chapter shows literature on masonry and its materials. The first section provides the general approach to analyse masonry, but also shows how the unique properties which allow the absence of centring, make this approach unfit for the thin-tile vault. The linear-elastic analysis is chosen as the better option. Afterwards the main material properties of brick are shown and how the properties for traditional translate to bricks used in thin-tile vaults. The next section provides a selection of suitable adhesives, as far as their specifications, properties and other information have been researched or provided. Lastly, relevant information from the most important source on engineering structures is provided: the Eurocode. The values presented in the Eurocode are related to the thin-tile vault.

The second chapter describes the principle method of the structural analysis. Starting with the geometric input from the design model and the structural & material properties, the end goal is to provide the construction sequence for the robotic model. This sequence is based on the stresses found in the structure with each time step. The time step is dependent of the sequence, as will become clear in this chapter. To get from the geometry to the construction sequence, a number of steps have to be taken. The first section provides an overview and the scope of this analysis. With the definition of the design model also came a shape of the vault in a lateral cross-section. The funicular curve has some interesting properties. These properties are the result of its structural scheme. However, only the shape derived from this curve is important to provide the structural analysis with sectional forces. This section shows how the partly constructed vault relates to a standard cantilever. The sectional forces found are used in the next section to calculate the stresses in the structure. Special attention is given into the stress distribution: either cohesive or incoherent monolithic. Last, the implementation is shown of how this analysis, this procedure, is used in the parametric model. It shows the use of a discrete set of analysed points, opposed to a continuous stress distribution analysis.

In the third and last chapter, the method of interpretation and verification is shown. Graphs are presented for further use in the results. Additionally, the two algorithms and their sets of rules are described with which the construction sequence is determined. Besides the use of the unity check, certain practicalities and structural optimizations can be taken. These result in two approaches which do not differ much, but differ none the less.

5

Literature on masonry and materials

5.1. Masonry material properties

Masonry is a composite material made of bricks or stones and a binding paste (generally mortar). The compressive strength of masonry is dependent on the strengths of its composite materials and therefore, the compressive strength is somewhere between the compressive strength of bricks (upper bound) and that of mortar (lower bound). Both materials have a different elastic modulus, which will result in an interaction between the two materials due to the different deformations and their cohesion. Five factors determine the reduction of the masonry compressive strength compared to the brick or stone compressive strength (Como, 2017):

- The ratio between the elastic moduli of the mortar and bricks;
- The ratio between the compressive and tensile strengths of mortar;
- The same ratio, but for bricks;
- The ratio between the thickness of the mortar bed and the height of the bricks;
- The Poisson's ratios.

These factors coincide with formulas found for FEM-calculations with masonry (Lourenco, 1996).

The tensile strength of masonry comes from the adhesion between the mortar and the brick (Como, 2017). Longitudinally tensile loaded masonry will fail in the interface between the mortar bed (fig. 4.2) and the brick, see figure 5.1. As discussed in chapter 3, the thin-tile vault has been attributed more tensile capacity than conventional masonry. In one of Guastavino's experiments, as described in the same chapter, the thin-tile vault can take up a bending moment. He attributes this due to the lack of the mortar beds taking up the full cross-section, stating that a structure without any joints is the ultimate goal to improve 'construction by cohesion'. However, it is more likely that due to the lack of an interface between the mortar bed and the bricks spanning the whole cross-section, another tensile failure mode has to take place before the masonry fails, which results in a higher tensile capacity of the structure. See figure 5.2 for the difference between traditional masonry and thin-tile vaulting when the mode of failure is failure in the bonding between the adhesive and the brick. The plane of failure & bonding in traditional masonry is not present in thin-tile vaulting. Whatever crack can be thought of, never will all three wythes have a plane of failure & bonding at the same place.

The shear strength of masonry is determined by two factors, each with their own failure mode due to shear. The first failure mode is when a masonry wall is subjected to a compressive and shear force. In general tests these two forces have the same value, and this results in a diagonal crack running through the bricks

Figure 5.1: Tensile failure in masonry arch

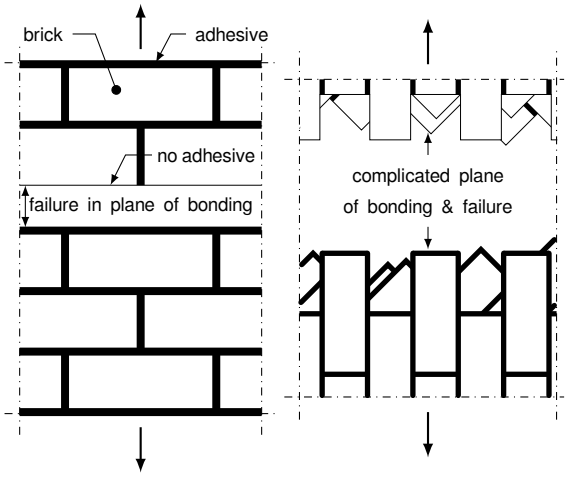
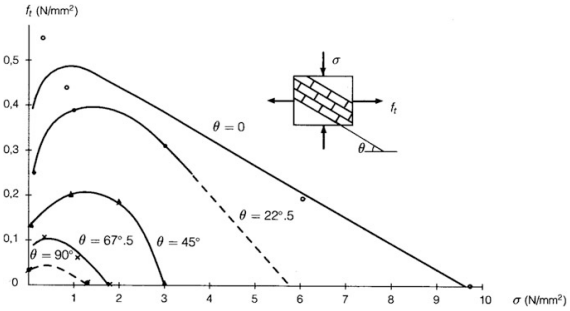
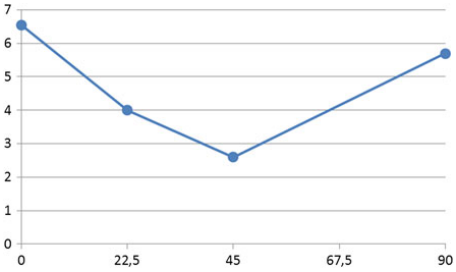


Figure 5.2: Comparison between conventional masonry and thin-tile vaults, seen at front view (normal to a brick wall) and view in the normal direction of the vault. On the left is the first mode of failure in tension in traditional masonry. The plane of failure is between a bed and a course of bricks (indicated with a thin line where adhesion has become absent). On the right is the same mode of failure in thin-tile vaults (three wythes are shown with angles (from intrados to extrados) -45°, 45° and 0°), showing that this type of failure is more difficult to occur.



(a) test results Samaringhe and Hendry, 1980;



(b) test results Facconi et al., 2014: the vertical axis is the compressive strength, the horizontal axis the angle;

Figure 5.3: Tests results from biaxial and uniaxial tests with a vertical compressive force and an altering angle of the courses to the sample (Como, 2017).

and mortar, splitting the wall in two. The other failure mode is when the mortar is of poor quality and the crack does not run through brick and mortar, but solely through the mortar. There's a third, hybrid failure mode, where both factors are present: the axial (compressive) force is quite high and the mortar is of poor quality. This failure mode exhibits both types of crack (Como, 2017).

Tests with biaxial loading (vertically a compressive and horizontally a tensile force) and uniaxial compression have been done in the past. The samples are (small) masonry walls where the courses have a varying angle to the walls/sample horizontal. In Como, 2017 two studies are presented that have a differing outcome. Of most importance is the maximum compression with the horizontal tensile force at 0. In the biaxial study, the compressive strength reduces with 0° (courses run horizontal, as is usual) as maximum and 90° as minimum. However, in the uniaxial study, the compressive strength is lowest around 45°, while both 90° and 0° are the maxima (with 0° being slightly greater). As is shown in appendix B and as Como, 2017 continues, it is more likely that the uniaxial test shows a truer relation between the compressive strength and the angle. The main factor is the ratio between brick surface area and mortar area in the cross-section. In figure 5.3 the results of both studies are shown.

For most, almost all, calculations done on masonry structures, the no-tension model of Heyman can be used. However, one of the main assumptions is that masonry can take very little tension. It has been presented that thin-tile vaults can take some tension. Thus, one of Heyman's assumptions is invalid. The construction of thin-tile vaults is better suited as if it were an elastic material, but completion of the structure comes with its own modelling approach. This shows thin-tile vaults form a unique typology within masonry structures. The different approaches for these two phases are well described in Como, 2017:

> Masonry may, in some exceptional cases, exhibit non-negligible tensile strength and its behavior could, at first sight, be modeled as a traditional elastic material. However, random dynamic actions, which can produce cracks in the masonry mass, will eventually cause the material to revert to no-tension behavior. The effects of subsequent slow penetration of humidity into the cracks can then make things even worse. In such cases, it is possible that a masonry structure which in its pristine state is able to sustain the action of given loads by virtue of its initial non-negligible tensile strength, will not be able to

> sustain the same loads later, when this strength is fading. In such cases the long-term behavior of masonry can be conservatively assumed to follow the no-tension model.

The no-tension model is not the only alternative for the structural analysis and modelling of masonry. The no-tension model of Heyman has proven to be fruitful for the stability of masonry structures. Continuum models of masonry have the masonry hypothesized as a perfectly no-tension material. Geometry-based models when used for the analysis of the stability of masonry vaults and domes, also make use of the no-tension model (D'Altri et al., 2020). This includes the Thrust Network Analysis of Philippe Block (Block, 2009)

5.2. Bricks

Fired bricks generally have a compression strength of 20 to 25 N/mm^2. Though, this may turn out lower when the bricks have been poorly fabricated, down to 5 N/mm^2. In general, the tensile strength is assumed as a tenth of the compression strength (Como, 2017).

Another material to look into is masonry made of rocks instead of (fired) bricks. Rocks come in many shapes and sizes, but most have a compressive strength from under 20 N/mm^2 up to over 30 N/mm^2. One of the interesting rock types are tuff blocks. These are volcanic rocks which have a high strength, but low density. In table Table 5.1 some mechanical parameters of tuff can be found (Como, 2017).

Thin-tile vaults, as the name suggests, make use of relatively thin bricks. The production and fabrication of these thin bricks is assumed to be similar as those found for brick façades with the use of brick slips (*steenstrips*). Vandersanden describes perfectly why these can be used for the thin-tile vault:

> "Een steenstrip is normaal gesproken een 2 cm dikke schijf, gezaagd van een baksteen. Doordat we de strip rechtstreeks van de baksteen zagen, blijven de kwalitatieve eigenschappen van de baksteen grotendeels behouden. Zo is de steenstrip even vorstbestendig als de baksteen en heeft deze dezelfde druksterkte."

The bricks used in brick slips by Vandersanden are from the common types like Waal-format. These bricks are cut in their depth, changing the dimensions from 210×100×50 to 210×20×50. With the thin-tile vaults, the cut is better performed in the other direction, to keep the bigger surface: from 210×100×50 to 210×100×20 (or a similar thickness).

As the quote also shows, the mechanical and physical properties of the cut bricks remain mostly constant with that of the original. Como, 2017 mentions the theory of proportions where the scale of the masonry structure is unimportant for its strength. Thus, it is likely the mechanical properties attributed to a masonry structure with Waal-format bricks is similar as of those with thin tile bricks.

5.3. Adhesives

5.3.1. Mortar

Mortar is a paste to bind and fill the bricks together. In general, mortars are made of an aggregate like sand, a binder like cement or lime and water.

There are generally four types of binder used in mortar: gypsum, lime, hydraulic lime and cement (Como, 2017). Gypsum is the oldest of these four and is made from the calcium-sulphate in gypsum stone. Gypsum has a relatively low strength. Lime (from limestone) is baked and slaked to create slaked lime (calcium hydroxide). Combined with sand and water simple lime mortar is made. Combining the slaked lime with a volcanic ash will create hydraulic lime. Their main difference is that the hydraulic lime is able to set under water. The strength of both also varies quite a bit. Simple lime has a compressive strength of just 0,5 MPa, while hydraulic lime is about 2 MPa. A more complicated process including both lime and gypsum will create cement. This more complicated process also results in higher strengths. Mixtures of both cement and hydraulic lime (sometimes referred

Table 5.1: Parameters of tuff blocks

Parameter	Value	Unit
Poisson's ratio v	0,15	
Elastic modulus	$30–150 \cdot 10^3$	kg/cm^2
Unit weight (volcanic tuff)	$1,1–1,7 \cdot 10^3$	kg/m^3
Compression strength	40–50	kg/cm^2
Tensile strength	1/15	f_c

Table 5.2: Physical performance of building gypsum (only strength in MPa) | (Zhang et al., 2020)

	Flex. strength	Compr. strength
2h	2,25	3,05
1d	2,62	4,46
3d (dry)	2,74	5,42

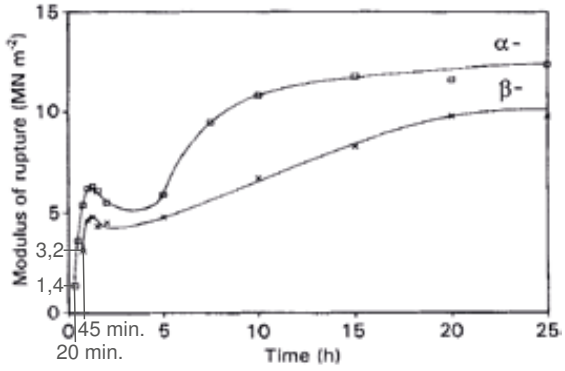


Figure 2 The development of strength as the hemihydrate sets.

Figure 5.4: The development of strength as the hemihydrate sets in gypsum plaster (altered) | (Lewry & Williamson, 1994b)

to as 'composite mortar') have a strength of 5 MPa, while pure cement mortars can go up to 12 MPa.

5.3.2. Plaster

Gypsum plaster has been the standard binder material for Guastavino. He used its quick setting properties to build the first layer. After this initial layer, he covered it with Portland cement and attached the second layer on top of the first. Thus, only for the first steps in construction would he use plaster. See illustration 5 in Philippe Block and Matthias Rippmann, 2013. The mechanical properties of gypsum plaster are quite low. That's why the first layer was built with gypsum plaster, which has a setting time of 5 to 20 seconds (depending on environmental conditions like temperature), and the rest of the masonry bonded with cement (Davis et al., 2012).

It should also be noted that throughout literature, it isn't often quite unclear whether plaster or mortar has been applied. In Davis et al., 2012 it states *"[...] gypsum mortar (plaster of Paris), [...]"*. In the abstract of Hashempour et al., 2021 it states *"Gypsum mortar is a common building material that can be used especially for plastering the walls."* It is also clear from definitions of both that a difference *is* made, but that it is unclear when a substance can be considered a mortar or when a plaster. Only its final use provides some guidance.

- Mortar: a mixture of sand, water, and cement or lime that is used to fix bricks or stones to each other when building walls;
- Plaster: a substance that becomes hard as it dries and is used especially for spreading on walls and ceilings in order to give a smooth surface (Cambridge, 2020).

Thus, two distinctions can be made. Mortar is used as a structural material with relatively high strength, while plaster is used as a finisher. In general, the ratio between the aggregate, the binder and water have a major influence on the properties of the mix, resulting in two different applications as well. As may have been noticed already from the previous section, information from literature, for instance regarding the strength development, is split on whether that piece of literature refers to their material as 'mortar' or 'plaster'.

Zhang et al., 2020 looked into the effects of several retarders on the setting time and strength of 'building gypsum' (gypsum plaster). The material used is so called β-type building gypsum from a Chinese factory. The characteristics are an initial and final setting time of 4,0 and 6,0 minutes and a strength grade of 2,0. The article provides a table of the physical performance of the gypsum, copied here in table 5.2. As can be seen, the flexural strength has already almost reach its maximum within an hour, while the compressive strength keeps growing in the days after. Sadly, the study does not provide any information for within the first few minutes, when the gypsum is actually setting. This is most crucial for the construction phase.

The Dutch Trade Association Gypsum (NBVG) provides a table of mechanical properties the varying types of gypsum have to fulfill according to DIN 1168 part 2 (NBVG, 2006). Although these are the minimum requirements for finished products, they can be indicative for other studies and give an indication of the performance of gypsum. In general, gypsum with a minimum requirement for flexural strength and compressive strength, have these at, respectively, at least 1,0 N/mm² and at least 2,5 N/mm².

Lewry and Williamson, 1994b is part of a series of studies on gypsum. This second part looks at the developing microstructure and its strength. Figure 2 in that study, here figure 5.4, shows a detailed progression of the flexural strength in

the 24 hours. Added are the values of the first measurements in both series. From this figure it is clear gypsum plaster rapidly develops a sufficient strength within the first hour. After respectively 20 and 45 minutes, the first measurement was done, resulting in a flexural strength of 1,4 and 3,2 N/mm^2. This study also provides an overview of the development of the microstructure and relates it to the strength development. The rapid rise of the strength development in the first hour is attributed to the formation of a matrix of dihydrate needles.

However, it is unclear how this first stage (the hydration reaction of hemihydrate) relates to the setting time. From the first part (Lewry & Williamson, 1994a) it is clear that this first stage coincides with the complete formation of dihydrate, while the temperature rises rapidly (more than 20 degrees Celsius). Again, it is unclear how this relates to the setting time. Yu et al., 2009 looks into the hydration process of gypsum, both looking at the dissolution of the hemihydrate and the precipitation of the dihydrate. This is related to the setting time of gypsum. From this study and other studies like the ones mentioned above, the setting time of gypsum plaster is much lower than an hour. It should be noted though, that both studies are performerd on different plaster (different manufacturer) and may not be completely comparably.

Thus, Lewry and Williamson, 1994b provides valuable information on the formation process of gypsum, and even in an earlier stage than other studies, but it still misses the vital information for the construction process.

The above mentioned studies provide some indication on the strength development of gypsum plaster. Other studies, like Karni and Karni, 1995, also looked into the hardening process of gypsum plaster, but none gives an indication into how the strength develops during the setting time, during the first minutes. It is important this development is known in the first few minutes, since it is the adhesive that binds the bricks together and holds them in place, while the structure is still unstable or cantilevering. The only graph that provides the information looked for, comes from SIKA, 2021. However, the axes provide too little information and after contact with SIKA, the information is either not present or impossible to share.

5.3.3. Epoxy

Epoxy resins are reactive intermediates used to produce a versatile class of thermosetting polymers (Pham & Marks, 2005). A categorisation of epoxies can be into one-component and two-component epoxies. One-component has the elements already mixed and requires a kick-start to start the hardening process. Two-component epoxies mixes the epoxy as it is ejected. The method of application of the epoxy differs per use. For floor coating, for instance, the epoxy-components are mixed in a bucket and spread evenly over the floor with the help of special brooms and other tools. Sometimes products that require an epoxy coating are dipped in a bath of epoxy. For most applications, the method for epoxies are quite similar to those for paint.

In case of small surfaces, especially with one dimension quite larger than the other, it is also possible to use special epoxy glue guns, see figure 5.5. These guns have a reservoir of two compartments and when ejected the components from each compartment are mixed and will start to harden. Important in the use of these guns is the time the epoxy spends in the nozzle. If the glue hardens in there, the nozzle has to be replaced. Not a major struggle to do manually, but not useful when a robot has to do it. Two important characteristics of epoxy are its pot life and the cure time. These can, roughly, be seen as similar to the initial and final setting time for mortars and other cement-products. The pot life determines how long a batch can be worked with, before it has hardened too much for further moulding. The cure time is used as a quality assurance, at which time the epoxy can be loaded.

Epoxy is influenced significantly by the

Figure 5.5: A glue gun

environmental temperature. General purpose epoxies will be comfortable with temperatures ranging from 10 to 30 degrees Celsius. However, for freezing temperatures special epoxies need to

be made, while the pot life and curing time can shrink significantly between 10 and 30 degrees.

There's a wide abundance of epoxies. This is necessary since epoxies have a lot of parameters that determine the most optimal epoxies to use. Besides the mentioned temperature and pot life/cure time, one can think of the adhesion with the base material (general distinction can be made between metals, thermosets composites, thermoplasts and various substrates, (Huntsman, 2020)), the viscosity, the mechanical properties, the ductility, the colour, and many more. Again, temperature and the setting times are important for construction. Additionally, the mechanical properties, the viscosity and the adhesion to the base material are important. A full analysis of which epoxy to use will not be necessary, but in table 5.3 the characteristics of some of the epoxies are shown.

Experiment
To investigate how the strength of epoxy develops during the setting time, a small scale experiment has been done. The layout of the experiment was to glue some ordinary bricks together with the help of a glue gun as shown in figure 5.5, and support one end of the brick while cantilevering the other. This way two aspects have been looked into: the shear strength or adhesion of the bond during setting and the tensile capacity during setting.

In appendix C the full experiment is shown. The main conclusion from this experiment is that it is best to assume the finale setting time, or cure time, as the minimum time before loading can be taken. Since this is a minimum loading (only its own weight), the required strength is similar to the categories 'walkable' (SIKA, 2012) and 'LSS > 1 MPa' (Huntsman, 2017).

5.4. Eurocode

Internationally, the Eurocodes are the most complete set of standards for designing and testing building structures for structural safety ("Eurocodes", n.d.). The Eurocodes provide information on calculation methods and values of parameters that can be used in a safe way. For the structural calculation of masonry, it is useful to provide this information from the Eurocode where applicable. The Eurocode concerned with masonry design is "Eurocode 6 - Design of masonry structures - Part 1-1: General rules for reinforced and unreinforced masonry structures", 2013. Three sections specifically are useful in this research: the strength capacities of masonry according to the Eurocode. The following three are described here: compressive strength, flexural (tensile) strength and shear strength. (sections 3.6.1, in "Eurocode 6 - Design of masonry structures - Part 1-1: General rules for reinforced and unreinforced masonry structures", 2013)

5.4.1. Compressive strength

Based on tests, bricks are given a normalised mean compressive strength (f_b). Similarly, the mortar has a compressive strength as well (f_m). The characteristic compressive strength of masonry is then determined by tests or by equation (5.1) using the two values of the components and the constants K, α and β. These constants always reduce the characteristic strength (always <1,0), compared to the strengths of the bricks and the mortar independently. Based on the brick classification the equation can be simplified. Bricks used in this research, and for thin-tile vaults in general, are class 1 bricks. The simplified formula is shown in equation (5.2).

$$f_k = K f_b^{\alpha} f_m^{\beta} \tag{5.1}$$

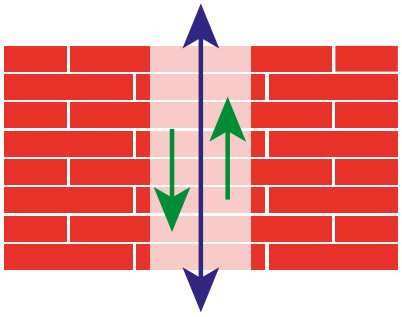
$$f_k = K f_b^{0,7} \tag{5.2}$$

The simplified equation and K are both dependent on the mortar used. Three types of mortar can be found in the Eurocode: general purpose mortar, thin-layer mortar and lightweight mortar. Since it is unusual for masonry to be constructed with epoxy, without any mortar, no standardisation has happened on this type of masonry. Thus, it is assumed epoxy, if it does fall under the types of mortar, is similar to thin-layer mortar. The difference between this thin-layer mortar and general purpose mortar is the maximum grain size of the aggregate (less than 2 mm). Epoxy corresponds best with this category. K for clay bricks in group 1 with thin-layer mortar is set as 0,75. To get to the design compressive strength, f_k should be divided by γ_m, which in this case ranges from 1,5 (temporary structures) up to 2,5 (monuments and bridges). Here, it is assumed γ_m is 2,2 (Buildings and other general structures).

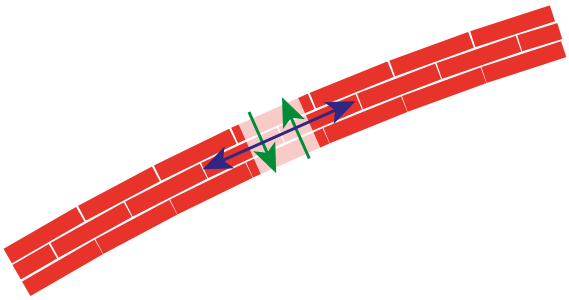
5.4.2. Shear strength

The shear strength of the masonry is calculated with equation 5.3. The calculation method for the shear strength in the Eurocode is based on a different situation that requires some attention when applying it to a thin-tile vault. In the previous section the influence of the vertical

compressive force has been mentioned (Como, 2017). However, this situation has the direction of the compressive and shear forces in the same direction. In the thin-tile vault, which in chapter 6 will be discussed further, these two forces are orthogonal, similar to that of a beam, see figure 5.6. Thus, a part of the formula can be neglected ($\sigma_d = 0$).



(a) Compressive and shear force directions in the Eurocode



(b) Compressive and shear force directions in the thin-tile vault

Figure 5.6: The difference in force directions between the Eurocode and the thin-tile vault

$$f_{vk} = f_{vk0} + 0,4 \cdot \sigma_d \tag{5.3}$$

f_{vk0} is the initial shear strength, usually taken from tests. However, in a broader framework, the values provided by the Eurocode will suffice. Again, following the same assumptions from the previous section f_{vk0} should be taken as $0,30 N/mm^2$.

Table 5.3

Name	Proc. Temp. (°C)	Bond strength (MPa)	Tensile strength (MPa)	Pot life (min)	Cure time	source
Sikadur-51	8-30	$\geq$ 1,5	$\geq$ 2,5	60 (20°C)	1-2 days	(SIKA, 2012)
Araldite 2015-1	10-40	-	31	45 (25°C)	4 hours	(Huntsman, 2017)
Araldite 2011	10-40	-	-	100	7 hours	(Huntsman, 2020)
Araldite 2012	10-40	-	-	6	20 minutes	(Huntsman, 2020)
Araldite 2019	10-40	-	-	110	4,5 hours	(Huntsman, 2020)
Araldite 2031-1	10-40	-	-	60	3 hours	(Huntsman, 2020)
Araldite 2051	0-40	-	-	5	15 minutes	(Huntsman, 2020)
Araldite 2053-05/15	10-40	-	-	5-15	20-40 minutes	(Huntsman, 2020)
Epoxy Brick Adhesive	-	2 (7 days)	15,2 (3 days)	20	12 hours (20°C)	(Epoxy Products Ltd, 2020)

5.4.3. Flexural strength

The flexural strength determines the maximum tensile stresses in bending. The Eurocode makes use of two flexural strengths (respectively f_{xk1} and f_{fxk2}. The first is bending where the curvature is perpendicular to the courses (the plane of failure is parallel to the bed joints). The second is bending where the curvature is parallel to the courses (the plane of failure is through the courses and perpends). Again, when possible these two values should be based on tests on the masonry. The Eurocode also provides values for these two flexural strengths if data is unavailable. With the previous made assumptions, f_{xk1} and f_{xk2} have the same value: $0,15N/mm^2$.

However. as stated in section 5.1, it is most likely the flexural strength of the thin-tile vault is higher than that considered safe for ordinary masonry. Since the adhesion is not the mode of failure, it can be assumed the flexural strength can be higher. The values of f_{xk1} for instance, can be considered useless. The plane of failure is parallel to the bed joints and the mode of failure in this direction would be between the bed joints and the course of bricks. Thus, the failure for f_{xk1} is likely the failure in adhesion, which is not present in a similar way for thin-tile vaults.

f_{xk2} has two modes of failure: either a plane of failure right through the course and perpends, or a plane of failure snaking around the bricks failing in the adhesion. These two modes of failure are similar to the shear failure modes. It could be possible the second failure mode (snaking around the bricks) is non-existent in thin-tile vaults. One reason is in the different wythes with each having their plane of failure at a different point. A misalignment of the perpends per wythe occurs due to the different curvatures. A second reason is in possible different orientations. The Eurocode does not provide planes of failure that run neither parallel nor perpendicular to the courses/bed joints. Studies on (two-way) bending of unreinforced masonry walls also lacked wythes with different orientations.

However, even with these two reasons, a full-scale experiment or in-depth analytical model would be required to confirm this. This is outside the scope of this research. The other mode of failure, where the plane of failure cuts right through the bricks, is determined by the lowest tensile strength. The tensile strength of bricks can vary on the type of bricks of used. One study found that a hollow clay block has a ratio of 0,039 between its compressive strength (34,7 N/mm^2) and its tensile strength (1,343 N/mm ^{2}) (Mojsilović, 2011). Although the block type is different than that of a solid block, this provides some indication into an expected tensile strength of bricks. The tensile strength of the mortar or the glue differs with time. Even after the adhesive has set, the strength capacity is still developing. The strength of mortars can vary greatly, but table 5.2 provides an indication for gypsum plaster after two hours, while figure 5.4 shows it after half an hour. Similarly, epoxy will have a final tensile strength of a couple of MegaPascal, but to get there the strength development only happens after the epoxy has set. Thus, it is likely the adhesive will surpass the tensile strength of brick, but until then the tensile strength of the adhesive is decisive.

Which failure mode will happen first is hard to estimate, as well as the flexural strength during construction. In a review of analytical formulations on the capacity of unreinforced masonry subjected to two-way bending, an overview of the mechanical properties of masonry walls taken from 8 studies have been shown in their table 1 (Chang et al., 2020). Important is that f_{x1} in that study ranges from 0,14 up to 1,37 N/mm^2 (with a half-way split roughly at 0,5 N/mm^2) and f_{x2} ranges from 0,41 up to 4,12 N/mm^2 (with the same split at roughly 1,5 N/mm^2). This would mean assuming a flexural strength for the thin-tile vault of 0,5 N/mm^2 is a safe assumption, more than three times the value found in the Eurocode. However, as stated earlier, f_{x1} will likely be higher in the configuration of the thin-tile vault. Thus, taking into account the maximum strength of the individual elements as well, it is likely the flexural strength will develop from a minimum value after setting time, up to 1,0 N/mm^2 as a conservative estimation.

'NPR 9096-1-1:2012 nl' provides some more in-depth values. Important is table 4 in the code where 'thin layer mortar with clay bricks from group 1 with additional specification in the scope statement' gives fxk1 and fxk2 of respectively 0,6 and 1,22 MPa. One of the assumptions is the flexural strength of clay bricks at 2,0 MPa, indicating the provided flexural strength of masonry is governed primarily by the mortar.

6

Structural analysis

6.1. From shape to stresses

The approach to determine the construction sequence is based on the maximum allowable stresses. The construction is assumed to take place from the base to the crown. This also allows for an implementation of such a construction sequence in more free-form vault designs.

However, the vault design considered here is a barrel vault, with assumed insignificant lateral loads. This simplifies the spatial geometry to a planar geometry: from a vault to an arch. As will become clear in section 6.2, this arch is symmetric in a barrel vault. Thus, whatever analysis is used to determine the construction sequence, can be applied to the other side as well. To get the stresses within the structure, a similar procedure can be used as is generally applied to slender structures. Starting from the base, the structure during construction is modelled as a cantilevering (curved) beam with a width of unit length.

The position of the bricks is determined by the design. The design should be seen as a shape, zx. This 1D-element (in 2D-space) is modelled as a (curved) line. This line should be assumed as the neutral line. Based on this line, this design, the diagrams for the displacement, the bending moment (Ms), the shear force ($V(s)$) and the normal force ($N(s)$) can be found. It should be noted though that the displacement of the structure during construction is omitted. Based on the diagrams of these sectional forces, the diagrams of the normal and shear stresses can be determined. However, to get from the sectional forces to the stress distribution, the cross-section is required as well. The cross-section is a rectangle (with width as 'per unit length'). Thus, the cross-section is only dependent on the thickness of the vault, which in turn is dependent of the number of wythes.

The material determines the maximum allowable stresses, together with the cross-section. Thus, the occurring stresses need to be within the domain these allowable stresses create. As the cantilever progresses, the only way to increase this domain is to increase the cross-section, thus the wythes of the vault. This is essential for determining the constructing sequence: when to apply the additional wythe on top of what has already been constructed.

6.2. Funicular curve

The first step is to define the design of the vault. As is clear from chapter 4, the design is limited to that of a barrel vault. However, research and projects of the past years have used the Rhinovault plugin from Grasshopper. This plugin designs the vaults as furnicular curves. Thus, the barrel vault also takes the shape of a furnicular curve. Since all parameters, or factors, are constant along its path, the vault can be simplified to an arch of unit width. Therefore, the standard formula for a furnicular curve can be used, see equation (6.1).

$$z(x) = \frac{H}{q}\,\text{cosh}(C_1 + \frac{q \cdot x}{H}) + C_2 \qquad (6.1)$$

where:

- H = constant horizontal compressive force throughout structure (or: horizontal component of the thrust) $[kNm^{-1}]$;
- q = downwards acting load along shape (describing the self-weight) $[\frac{kN}{m}m^{-1}]$;
- C_1 , C_2 = constants in this family of functions $[-], [m]$.

The constants in this equation are determined

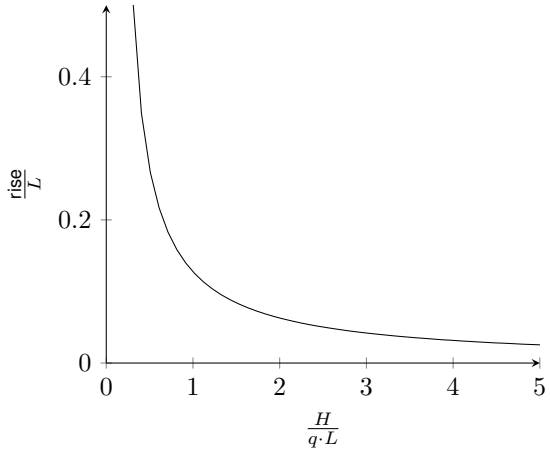


Figure 6.1: Relationship of forces versus shape in a furnicular curve, based on equation (6.2)

with the design space of the structure. In this case the design space is defined by the span and the rise. However, this design space should be anchored, or still a family of functions is possible, resulting in a family of values for the constants as well. In this research every analysis is assumed to be with the axis of symmetry at $x = 0$. This results in the following boundary conditions:

$$z(x = 0) = \text{rise}$$
$$z'(x = 0) = 0$$

Immediately those familiar with mathematical analysis will recognize that with this assumption this results in:

$$C_1 = 0$$

Furthermore, with the general assumption, the domain of the structure has a lower and upper bound:

$$x_{Lower} = -\frac{\text{span}}{2}$$
$$x_{Upper} = \frac{\text{span}}{2}$$

At these bounds the other constant can be found, following this elaboration:

$$z(x = -\frac{\text{span}}{2}) = 0$$
$$C_2 = -\frac{H}{q}\cosh(\frac{q \cdot \text{span}}{2 \cdot H})$$

The hyperbolic function $\cosh()$ has an important characteristic, resulting in the same value from either side of the structure.

$$\cosh(-\frac{span}{2}) = \cosh(\frac{span}{2})$$

Implementing these constants in the formula gives equation (6.2).

$$z(x) = \frac{H}{q}\left(\cosh(\frac{q \cdot x}{H}) - \cosh(\frac{q \cdot L}{2 \cdot H})\right) \quad (6.2)$$

where:

$L =$ span of the structure, only positive values $[m]$.

From this the rise of the structure can be found for $x = 0$:

$$\text{rise} = \frac{H}{q}\left(1 - \cosh(\frac{q \cdot L}{2 \cdot H})\right)$$

Within structural design, the ratio between the rise and the span is quite important for the designs performance. Relating these gives:

$$\frac{\text{rise}}{\text{span}} = \frac{H}{qL}\left(1 - \cosh(\frac{qL}{2H})\right)$$

Now it becomes clear both the term inside the hyperbolic function as the overall multiplication are related (off by a factor 2). Making it possible to plot this relationship, as can be seen in figure 6.1. Interesting to note as well, is that the y-axis displays the ratio of geometry, while the x-axis displays the ratio of forces. From this figure, it is clear that the function has an asymptote at $\frac{H}{q \cdot L} = 0$, which makes sense, given that the thrust is dependent on the geometry. In a similar way, the asymptote at $\frac{\text{rise}}{L}$ is impossible, since the structure would be like a beam, requiring an immense thrust. Another point of interest is when $\frac{H}{q \cdot L} = 1$, or when the thrust is equal to the total load. At this point, the rise is *almost* equal to an eighth of the span. An eighth is a well known constant from the calculation of uniform distributed loads:

$$M = \frac{1}{8} q_{\text{UDL}} \cdot l^2 = H \cdot \text{rise}$$

The reason it isn't exactly an eighth, in this case, is that the load is non-uniform: the self-weight along x is more where the slope of the structure is larger.

6.2.1. Deriving the formula for a furnicular curve

Now that it's clear how the function of a funicular curve influences its geometry, it is useful to take a step back at the derivation of this formula, to use its analogy further for the structural analysis.

In figure **??** the fundamentals of the arch typology are shown. Notice how the load is not projected

onto the arch, but is following it. For convenience, the notation already takes into account that the limit of Δx goes to zero. The first step is to define the load. q is the self-weight of the structure, dependent on the volume of the structural material. This volume can be defined as the (constant) thickness perpendicular to the funicular curve times the length of the curve. Thus, the total load can be found in equation (6.3).

$$F_{total}(x) = q \cdot s(x) = \rho \cdot t \cdot s(x) \tag{6.3}$$

where:

$s(x) =$ the length of the structure along its geometry.

The length of the structure can be approximated by the Cartesian coordinates x and $z(x)$. A section of the structure Δs is almost similar to the hypotenuse of Δx and $\Delta z(x)$. Taking the limit, this difference becomes zero. Rewriting this gives equation (6.4).

$$\begin{aligned} \Delta s(x)^2 &\approx \Delta x^2 + \Delta z(x)^2 \\ \mathrm{d}s(x)^2 &= \mathrm{d}x^2 + \mathrm{d}z(x)^2 \\ \mathrm{d}s &= \sqrt{\frac{\mathrm{d}x}{\mathrm{d}x}^2 + \frac{\mathrm{d}z}{x}^2}\,\mathrm{d}x \\ \mathrm{d}s &= \sqrt{1 + \frac{\mathrm{d}z}{\mathrm{d}x}^2}\,\mathrm{d}x \end{aligned} \tag{6.4}$$

To relate the loading to the (horizontal component of the) thrust, the equilibrium of forces is required. The first is the moment equilibrium from equation (6.5) and the second is the vertical force equilibrium from equation (6.6). A horizontal equilibrium already exists, since H has to be constant throughout the arch.

$$\begin{aligned} \Sigma T(+\mathrm{d}x) = 0 &= -H \cdot \mathrm{d}z - V(x) \cdot \mathrm{d}x + \frac{1}{2} q \mathrm{d}s \mathrm{d}x \\ &= -H \cdot \mathrm{d}z - V(x) \cdot \mathrm{d}x \end{aligned} \tag{6.5}$$

where:

$V(x) =$ the shear force in the xz-plane.

$$\begin{aligned} \Sigma F_V(+\mathrm{d}x) = 0 &= -V(x) + q\mathrm{d}s + V(x) + \mathrm{d}V \\ &= -V(x) + q\frac{\mathrm{d}s}{\mathrm{d}x}\mathrm{d}x + V(x) + \mathrm{d}V \end{aligned} \tag{6.6}$$

The last term in (6.5) is equal to zero and the term $q\mathrm{d}s$ in (6.6) can be rewritten to relate to x. Combining equations (6.3), (6.4), (6.5) and (6.6), results in equation (6.7)

$$H\frac{\mathrm{d}^2 z}{\mathrm{d}x^2} = q\sqrt{1 + \frac{\mathrm{d}z}{\mathrm{d}x}^2} \tag{6.7}$$

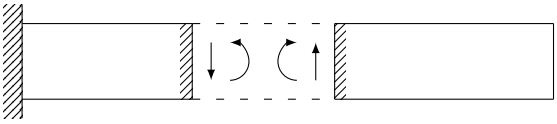
This equation is further altered by substituting $\frac{\mathrm{d}z}{\mathrm{d}x} = \sinh(\zeta)$, resulting in equation (6.1).

$$\begin{aligned} H\frac{\sinh(\zeta)}{\mathrm{d}x} &= q\sqrt{1 + \sinh(\zeta)^2} \\ \cosh(\zeta)^2 + \sinh(\zeta)^2 &= 1 \\ \frac{\mathrm{d}}{\mathrm{d}x} sinh(\zeta) &= \cosh(\zeta) \\ H\cosh(\zeta)\frac{\mathrm{d}\zeta}{\mathrm{d}x} &= q\cosh(\zeta) \\ \frac{\mathrm{d}z}{\mathrm{d}x} = \sinh(\zeta) &= \sinh(\frac{qx}{H} + C_1) \end{aligned}$$

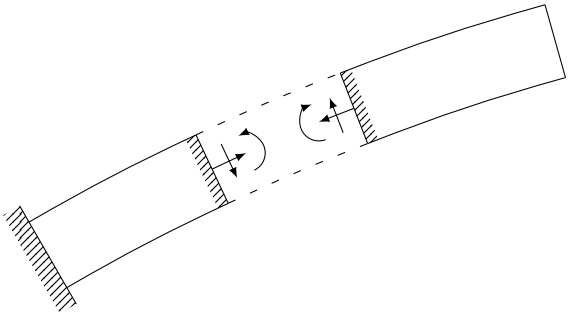
6.2.2. Aligning planes & the cantilever

The forces considered above are in the xz-plane. However, to calculate the maximum stresses, it is necessary to align the plane to the geometry. For this, an sr-plane is introduced. This plane has as orientation the s-axis aligned to the tangent of the geometry and the t-axis aligned to the normal. The plane of reference is the global xz-plane and consequently the transformation happens from the xz-plane to the sr-plane. This transformation is a rotation of size α. This rotation is equal to the angle the geometry makes with the x-axis. This angle has been defined before as the derivative of the geometry. Thus, the rotation between planes is equal to the slope of the geometry.

$$\begin{aligned} R_\alpha(x, z) &\rightarrow (s, r) \\ \tan(\alpha) &= \frac{\mathrm{d}z}{\mathrm{d}x} \end{aligned}$$



(a) The sectional forces in a cantilever aligned with global Cartesian directions;



(b) The sectional forces in a cantilever of an unfinished arch;

Figure 6.2: Sectional forces in a cantilever used for the equilibrium method. The arrows shown indicate the moment, the shear force and the normal force.

This transformation can be used later to transform the global forces to the sectional forces. Additionally, during construction the structure behaves differently than after completion. During construction, only part of $s(x)$ or $z(x)$ is built. This has some implications for the further analysis of the structure. Mainly, from here on out, the structure is not considered an arch anymore. It is a cantilevering curved beam (or shell per unit length), clamped on one side (at $x = \frac{-L}{2}$) and free on the other. The x-position of this free end varies during construction. To determine this position, and of every brick in between, the equations previously introduced need to shift from $s(x)$ to $x(s)$.

$$s = n \cdot l$$

$$x(s) = \frac{H}{q}\left(\text{arcsinh}\left(-\frac{qs}{H} + \sinh(\frac{qL}{2H})\right)\right)$$

where:

$n =$ index of brick along geometry;
$l =$ length of one brick unit.

6.3. Equilibrium and sectional forces

As already touched upon in the previous section, the analysis of the forces happens in two stages. First the forces are calculated from the reference of the xz-plane, then these are transformed to the sectional forces in the sr-plane.

The global forces are calculated using the equilibrium method. If the cantilevering beam, as described in subsection 6.2.2, is assumed as a rigid body, it becomes clear this structure is statically determinate. This allows for the calculation of the reaction and internal forces, without the need for material properties. With the construction of this structure, two situations can be analysed. Either the load is considered continuous along the structure (q) or the load is a series of point loads, positioned at the centroid of each brick unit. Both situations have their advantages and disadvantages.

For the first, this is useful as this loading is independent of the orientation of the bricks along the structure, as described in chapter 4. Additionally, as the structure becomes larger (as the construction progresses), the existing structure is uniformly loaded and becomes a larger fraction of the total load on the structure. The point loads, however, represent better the way the loading is over time. As each brick is added, it increases the loading at once, instead of a gradual increase as the distributed load would suggest. Another perspective can be seen when these loading schemes are related to a variable. Based on the described (dis)advantages, the distributed load helps to understand the loading on and forces in the entire structure at a given time, while the concentrated loads show a better development over time. The effects of the distributed load is best analysed from the perspective of x, while the effects of the point loads is best analysed from t, where x is the extend of the cantilever and t is the construction time.

From here on out the calculations are done using a distributed load. Mathematically this aligns better with the structural analysis done on (continuous) elements. Furthermore, the point loads can be rewritten as a distributed load, if the software with which the calculation is made, makes this possible.

In figure 6.2a this simplified structure is shown for the equilibrium method. Similarly to subsection 6.2.1 the necessary calculations can be done, given the following equations.

$$\Sigma T(+\mathrm{d}x) = 0 = M + V_x\mathrm{d}x - M - \mathrm{d}M - \frac{1}{2}q\mathrm{d}s\mathrm{d}x$$

$$\Sigma F_V(+\mathrm{d}x) = 0 = V_x - V_x - \mathrm{d}V_x - q\frac{\mathrm{d}s}{\mathrm{d}x}\mathrm{d}x$$

These will result in equations (6.8) & (6.9). A

visual representation of these equations can be seen in figure 6.3.

$$\frac{\mathrm{d}M}{\mathrm{d}x} = V_x \tag{6.8}$$

$$\frac{\mathrm{d}V_x}{\mathrm{d}x} = -q\frac{\mathrm{d}s}{\mathrm{d}x} = -q\sqrt{1+(\frac{\mathrm{d}z}{\mathrm{d}x})^2} \tag{6.9}$$

where:

$q =$ either a continuous, constant value $q(s)$, or a function with point loads F spread evenly with Δs.

However, the forces from these equations are within the xz-plane, the global coordinate system. These forces need to be transformed for the sr-plane, where s is in the direction $\frac{\mathrm{d}z}{\mathrm{d}x}$ and r is perpendicular to that in the same rotation as z is to x. The sectional forces $V(x)$ and $M(x)$ then result in $(V(s)$, $N(s)$ and $M(s)$. Since the moment is the rotation around the normal of the xz-plane, and the normal does not undergo any transformation, the moment in the sr-plane is the same: $M(x) = M(s)$. The forces $V(s)$ and $N(s)$ are the resolution in the sr-plane of the force $V(x)$. The orthogonal triangle of these forces has a geometrical equivalent. This can be seen in figure 6.4.

When these have similarity, the forces in the sr-plane can be found using the known geometrical values. The relationships are shown in (6.11). Important is that the correct signs are used. Since the global z-axis is downwards, the slope of the arch/vault in the first half, as shown in figure 6.3a, is negative. Also, from figure 6.4a it is clear that a positive shear force in the global system $(V(x))$ results in a negative normal force along the geometry $(N(s))$. In this figure, the slope direction of the arch is already taken into account. Noted, if this was shown in the direction with a positive $\frac{\mathrm{d}z}{\mathrm{d}x}$, the resolution of $V(x)$ would have resulted in a $N(s)$ and $V(x)$ of the same sign.

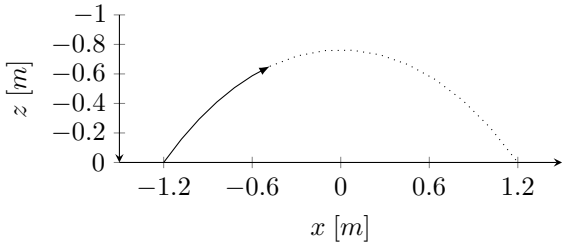
Due to the similarities between the vectors of force and geometry, equation (6.10) can be made. Rewriting this equation results in the calculation of $V(s)$ and $N(s)$, as shown in (6.11). With this the sectional forces in the cross-section of the structure are known. These are necessary to test whether the occurring stresses will exceed the maximum allowable stresses.

$$\frac{V(x)}{\frac{\mathrm{d}s}{\mathrm{d}x}} = \frac{V(s)}{1} = \frac{-N(s)}{-\frac{\mathrm{d}z}{\mathrm{d}x}} \tag{6.10}$$

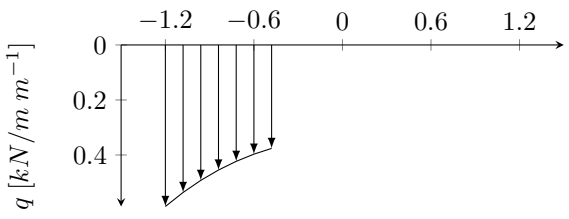
$$V(s) = \frac{V(x)}{\frac{\mathrm{d}s}{\mathrm{d}x}}$$
$$N(s) = \frac{V(x)\frac{\mathrm{d}z}{\mathrm{d}x}}{\frac{\mathrm{d}s}{\mathrm{d}x}} \tag{6.11}$$

6.4. Stress distribution

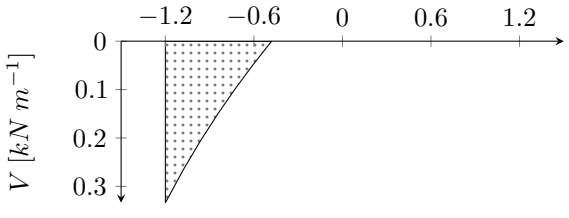
The sectional forces result in normal and shear stresses. As presented in (Hartsuijker & Welleman, 2007), (the maximum of) these stresses can be found given the equations (6.12). However, the cross-section is symmetric, resulting in a simplification and rewriting the factors I_{rr}, t and S_r. These are shown in



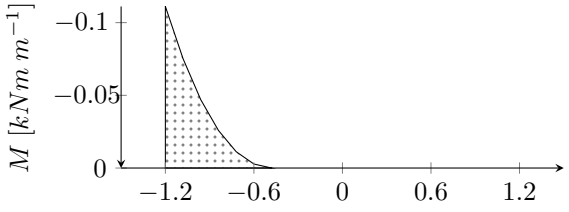
(a) Shape z(x) partly constructed (solid line) and final shape (dotted line);



(b) Load q(x) projected onto shape z(x);

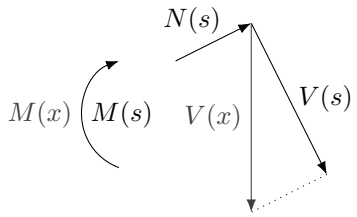


(c) Shear force V(x) in the xz-plane for loading q(x);

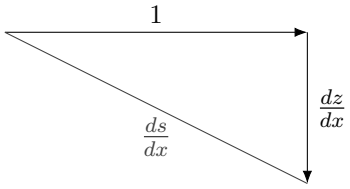


(d) Bending moment M(x) in the xz-plane for loading q(x);

Figure 6.3: A structure partly constructed and its accompanying loads, shear forces in the global coordinate system and the moment.

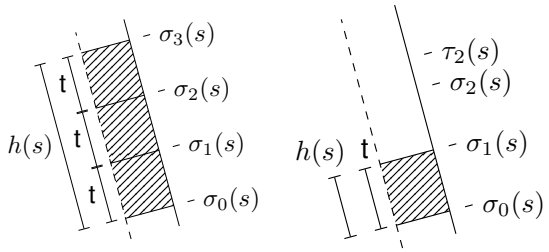


(a) Resolution of force $V(x)$ in $V(s)$ and $N(s)$, and $M(s)$ and $M(x)$ shown (positive directions shown);



(b) Resolution of the geometry in the xz-plane (positive directions shown);

Figure 6.4: The similarity between the vectors of the forces and the vectors from the geometry.



(a) Three wythes constructed; **(b)** One wythe constructed;

Figure 6.5: Position of the stresses used in calculations.

Table 6.1: Factors with which to multiply the maximum bending stress from equation (6.13)

		Number of wythes constructed		
		1	2	3
Position	0	-1	-1	-1
	1	1	0	-1/3
	2	-	1	1/3
	3	-	-	1

equation (6.13).

$$\begin{aligned} \sigma(r) &= \sigma^N + \sigma^M = \frac{N}{A} + \frac{Mr}{I_{rr}} \\ \sigma_{sm} &= \tau(r) = -\frac{V_r S_r}{bI_{rr}} \end{aligned} \tag{6.12}$$

$$\begin{aligned} \sigma(r) &= \sigma^N + \sigma^M = \frac{N}{A} \pm \frac{M}{W} \\ \sigma_{sm} &= \tau(r) = -V_r \frac{12\alpha^2 - 3}{2h(s)} \end{aligned} \tag{6.13}$$

where:

$\sigma(r)$ = normal stresses in the structure due to $N(s)$ and $M(x)$ $[\frac{N}{mm^2}]$;
$\tau(r)$ = shear stresses due to $V(s)$ $[\frac{N}{mm^2}]$;
A = area of the cross-section $[m^2m^{-1}]$;
W = section modulus of the cross-section as $\frac{I_{rr}}{r}$ $[m^3m^{-1}]$;
α = relative height from -0.5 to 0.5, with the neutral axis at $\alpha = 0$ $[-]$;
$h(s)$ = height of the cross-section at s;
I_{rr} = $\frac{h^3}{12}$, moment of inertia for this cross-section in the sr-plane $[m^4m^{-1}]$;
S_r = $h^2\frac{4\alpha^2-1}{8}$, static moment for this cross-section in the sr-plane $[m^3m^{-1}]$.

As can be seen in these equations, A and W are per stretching meter. For instance, the area is normally height times width, but here the width is neglected, since the model has been reduced from a 3D vault to a 2D arch. Thus, formally the area is still in square meters, but per meter along the vault. This results in $A = h(s)$ and $W = \frac{I_{rr}}{1/2h(s)}$.

The thickness of the vault (h) varies during the construction. So, actually, it should be $h(s,t)$ with s being along the structure and t being the construction time. However, as will be shown later, the construction time can be ignored for the calculation. The thickness of the cross-section (t) varies between four discrete values: 0, 1, 2, 3 times the thickness of the bricks. A thickness of zero is of course for that part of the structure that has not yet been constructed (see the dotted line in figure 6.3a). h increases with t for each wythe constructed at that point. Thus, with three wythes, the maximum increase is $3t$.

The maximum normal or bending stresses occur at the top and bottom of the height of the cross-section. However, this height varies over the structure's path and it varies in time. To be able to calculate these maximum stresses with ease, as can be seen in section 6.5, the stresses are measured only at specific locations of the cross-section. This is done at the start and end of each wythe. In subsection 6.4.1 it is shown that where two wythes meet, the stresses are continuous as if the structure were monolithic. Thus, the total of specific locations is only four: $0t$, $1t$, $2t$ and $3t$. See figure 6.5.

The normal stresses are constant with the height. However, to be able to add these to the bending stresses, these values are also presented at these specific points. The bending stresses at each location are the value found from equation (6.13)

multiplied with the factors to the relative height. These factors can be found in table 6.1. Since W already takes the height into account, the factors are normalized. After all, the bending stress distribution can only take place along the existing height of the structure. By multiplying from -1 to +1, the sign of $\frac{M}{W}$ in equation (6.13) is always positive.

The maximum shear stress can be found with $3/2V/A$ at half the height. However, throughout the construction this midway point will shift. Thus, it is important to calculate the shear stresses at multiple points as well. The formula from equation (6.13) makes it possible to do that. Again positions along the cross-section height are chosen. Different to the bending stresses, now it is the three mid-points per wythe, of which τ_2 is shown in figure 6.5. The factor α is shown in table 6.2.

With these factors the applied stresses $\sigma(s,r)$ and $tau(s,r)$ for any $h(s)$ can be found. These are compared to the maximum allowable stresses in a unity check. The maximum allowable stresses are found in literature and are repeated in table 6.3. The unity check shows whether the structure can hold its own weight.

6.4.1. FEA Modelling of the monolithic structure

The challenge with a layered structure is to know with certainty how much the layers or wythes work together. Probably the most well-known cooperating layers are (concrete-steel) composite floors and cross-laminated timber. This last one is most similar to the structure in this research: an adhesive between more familiar material layers. In CLT-design it is assumed the glue has a higher strength than the timber. This makes the material monolithic: the strain and stresses are distributed continuously over the cross-section. Otherwise, each layer would have its own distribution, resulting in discontinuous endings of the elements. See figure 6.6 for a visualization. A Finite Element Analysis program has been used to find out which of these two models is best suited for the structure.

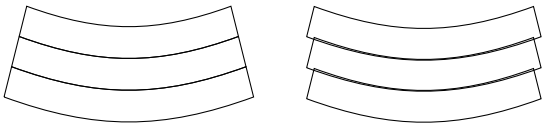
In this FEA program two models have been made. The first model is one volume which acts as a monolithic material. The second model is three volumes which have been connected with interface elements. Both models have the same material properties, and the same loading schemes. In appendix B an overview of the full model is given. In figure 6.7 the stresses in both models can be seen. These stress distributions show that the material can best be seen as a

Table 6.2: Factor α from (6.13)

		Number of wythes constructed		
		1	2	3
Position	0	0	-1/4	-1/3
	1	-	1/4	0
	2	-	-	1/3

Table 6.3: Maximum allowable stresses after full hardening. Note f_k is derived from $-0,75 \cdot {f_b}^{0,7}$

yield strengths	value $[N/mm^2]$
compressive strength	-6,11
tensile strength	2,00
shear strength	0,30



(a) Cohesive monolithic element;

(b) Incoherent composite element;

Figure 6.6: The cooperation between layers or wythes in the stress- & strain-distribution.

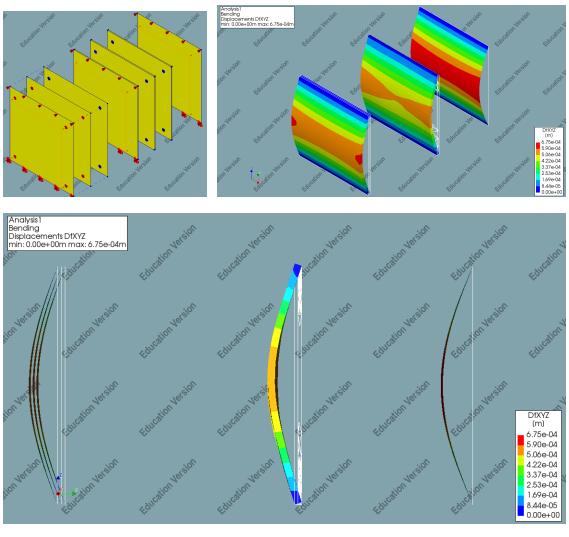


Figure 6.7: An FEA model of multi-layered masonry. In the top left the model is shown. The wythes are modelled as plates or solids. Three plates are connected with their interfaces to each other (lines from blue to red dots). The closest plate is supported on its top and bottom edges (red arrows) and the loading is a distributed load normal to this first plate (orange box and arrows). The other two images show the displacement in the plates. All plates have a similar displacement (bottom image), showing their working as one material. Additional information can be found in appendix B.

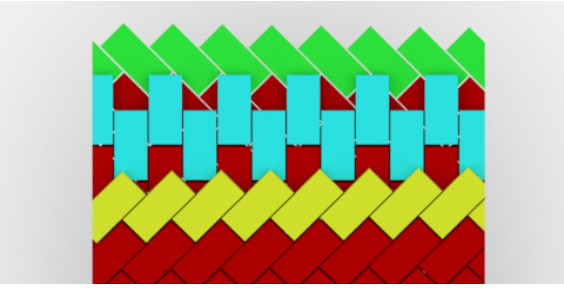
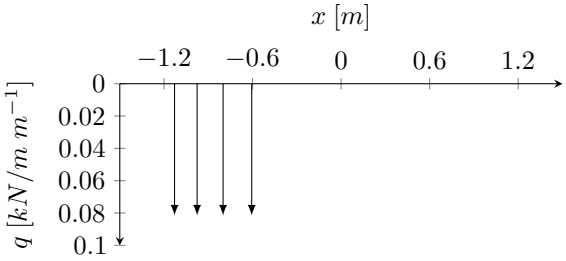
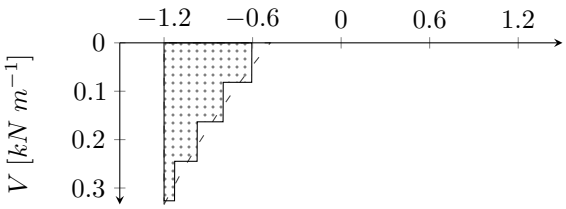


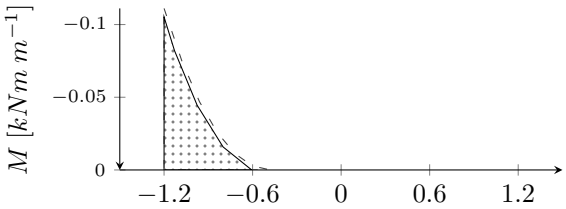
Figure 6.8: Set of bricks placed per row. Green are a row of bricks in the first wythe, blue a row in the second wythe, yellow for the third wythe.



(a) Loading q(x) with point load F);



(b) Shear force V(x) in the xz-plane (in dark gray the loading from 6.3c;



(c) Bending moment M(x) in the xz-plane (in dark gray the loading from 6.3d;

Figure 6.9: The loading scheme with a discrete load distribution, differing from figure 6.3b

monolithic material. Thus, the stresses within the structure are calculated with a continuous distribution.

6.5. Implementation in the parametric model

In the parametric model the calculation from the previous sections has been done in a script component written in a programming language. The full code of this model can be found in appendix A. The programming language can be found in appendix D.

As described in chapter 4, the design model results in bricks oriented and positioned correctly in a 3D-space. In chapter 9 the sequence to build the bricks is further described, but for now it is assumed the bricks are laid down in such a way that the added bricks have a similar cantilever length, before moving on to the next set of bricks to be placed, see figure 6.8. This is done to keep the simplification from the structural analysis: reduce the structure from 3D to 2D. In the parametric model this simplification is implemented by taking the average position of the set of bricks to be placed and use these as the cantilever length of the structure so far.

These distances result in the loading scheme of and during construction. With the same procedure from figure 6.3, figure 6.9 shows how this loading results in the shear force and the bending moment in the structure. However, the calculation in the parametric model is done at certain intervals. The programming language requires the calculation at specified locations and these have been defined in list **x** of size n. The same goes for the centroids of the bricks: for each wythe a list $\mathbf{y_i}$ is used. Both lists are in the same direction (x in the figures here and y in the global coordinate system of the parametric model. Again the goal is to get to the stresses of the structure. In the programming language, the stresses can be found in the matrix Σ with dimensions $n \times m$, where m comes from figure 6.5.

It is important that for each row of bricks added, a change happens in the matrix Σ. Thus, equation (6.14) should hold. This also means that n is bigger than the amount of rows to be placed per wythe. For computational purposes, n should be as low as possible, however. It is easy to assume that x_i should be between the connection of two rows of bricks, roughly halfway $y_{i,j}$ and $y_{i,j-1}$. However, with different orientations, the length of each $\mathbf{y_i}$ is different (the amount of bricks per wythe differs). Thus, in the programming language the span of the vault is split with a constant step to create **x**. After that the code checks whether every consecutive couple in $\mathbf{y_i}$ has any value of **x** in between. If this would not be the case, it is advised to decrease Δx, increasing n.

$$\begin{aligned} \Delta x &= x_i - x_{i-1} \\ \Delta y_i &= y_{i,j} - y_{i,j-1} \\ \Delta x &< \Delta y_i \end{aligned} \tag{6.14}$$

As mentioned before, scripting requires positions to make the calculations. This is the purpose of the positions in **x**. Simultaneously, the script in the programming language creates zero list **h** with scale n. Within the script, with each iteration, the four force lists are created as well, all with scale n. See 6.4a for these forces. These force lists are divided by the lists **A** and **W**, both dependent on the values from **h**, to get to the stresses at that position. A number of these lists can be seen in equation (6.15).

$$\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_{n-2} \\ x_{n-1} \end{bmatrix}, \mathbf{h} = \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_{n-2} \\ h_{n-1} \end{bmatrix}, \quad \mathbf{A} = \begin{bmatrix} A_0 \\ A_1 \\ \vdots \\ A_{n-2} \\ A_{n-1} \end{bmatrix}, \mathbf{W} = \begin{bmatrix} W_0 \\ W_1 \\ \vdots \\ W_{n-2} \\ W_{n-1} \end{bmatrix} \tag{6.15}$$

Subsequently, it is required to create the factor matrices for σ and τ. As mentioned before, the stresses are matrices $n \times m$. Thus, these factor matrices $\boldsymbol{F}$ have the same dimensions. Based on the value h_i, the value of these factors are determined. For the four, respectively three values in $\mathbf{f_i}$ the values of one column in table 6.1 are used. These matrices are used to check whether the maximum allowable stresses are surpassed. $\boldsymbol{\Sigma}$ can be seen in equation (6.16).

$$\boldsymbol{\Sigma} = \begin{bmatrix} \sigma_{0,0} & \sigma_{1,0} & \sigma_{2,0} & \sigma_{3,0} \\ \sigma_{0,1} & \sigma_{1,1} & \sigma_{2,1} & \sigma_{3,1} \\ & & \vdots & \\ \sigma_{0,n-2} & \sigma_{1,n-2} & \sigma_{2,n-2} & \sigma_{3,n-2} \\ \sigma_{0,n-1} & \sigma_{1,n-1} & \sigma_{2,n-1} & \sigma_{3,n-1} \end{bmatrix} \tag{6.16}$$

It is the unity check that determines whether or not a row of bricks is added to the structure. The unity check should be lower than 1. Since this research does not aim to optimize the structure, it is not necessary to approach this upper bound value. However, it is better to analyse the influence of the unity check on the maximum cantilevering length. In the early phase of construction this upper bound value is best to be even lower than 1. This is further looked into in chapter 7.

7

Stress distribution in the phased construction

On the next pages the output of the structural analysis is shown. The parameters used for this calculation are similar to those used for the first computation in section 11.2. Shown are the structural analysis after one, two, three, four and thirty-one bricks have been placed. For each position in **x** a stress distribution diagram can be made. For clarity purposes a select view are shown, see figure 7.1a. These four cross-sections have been selected based on $x = (4 * i)^2$ [m] with i being one of [0,1,2,3]. These four, combined with the chosen analyses, show how each stress diagram comes to be, and how these translate to the figures used in chapter 11.

Figure 7.1 is an introduction into the stress diagrams. The position of the four cross-sections are shown on the vault, see figure 7.1a. Figure 7.1b shows the stress diagrams and what markings have been added for translation to the later figures. It starts with a classic stress diagram with tension on the positive side. This stress diagram results in a couple of significant stresses in this cross-section, indicated with circled markings. Furthermore, the strength also changes over time. Thus, it is important to show how the stresses relate to these strengths. These are shown as background filling in the colours maroon (compression) and navy (tension). Again, significant strengths can be found that are used for later figures, indicated with crossed markings. Only stresses and strengths can be seen for $x = 0.0$ [m], since only in this cross-section do the centroids of the bricks placed, extend further. In the stress diagrams of the figures 7.2 to 7.6 the upper and lower bound of the horizontal axis have been changed to ensure all occurring stresses can be shown. In figure 7.1 these bounds have been reduced to maximize the extend of the stresses and strengths after one (row of) brick(s) has been placed.

On a side note, since each row of bricks is calculated as one, the figures may simply state 'after brick i is placed', while fully it would be 'after row of bricks i is placed'.

In figure 7.2 the stresses and strengths after one row of bricks are shown. As stated before, only for the cross-section $x = 0.0$ [m] do these occur. The stress diagrams have been placed along the structure, positioned similarly as the cross-section markings are in figure 7.1a. Since only one wythe of bricks has been placed, only two significant stresses are found. As showing each cross-section would clutter the pages, an alternate visualisation has been found. Note how the axes have changed in this new figure! In the stress diagrams the horizontal axis is the stress, and the vertical axis is the position in the cross-section after all three wythes are constructed. In this new figure, figure 7.2b, the vertical axis is now the stress, and the horizontal axis is the position of the cross-section on the vault (based on the global Cartesian coordinates). The lines represent the stresses σ and the areas represent the strengths f. Since both from the stresses and from the strengths two significant values are found, only four of the nine lines/areas are shown in figure 7.2b. The markings from the stress diagrams are also present in this figure.

In total nine lines/areas will become visible, see the later figures. The four stresses are based on figure 6.5. The colours (red, magenta, blue, green) are based on their likely stress state: red for the interface under compression, magenta for the interface switching signs, and blue and green for the interfaces always under tension. The colours of the strengths are based on their accompanying stress. The second interface σ_1

has two strengths in these graphs. This interface can be both tension and compression. Thus, for whichever sign the stress has, the accompanying strength is shown. Hence why both $f_{k,1}$ and $f_{kt,1}$ are present.

These graphs are able to show that in each cross-section each significant stress does not surpass its accompanying strength. For each new row of bricks it is determined whether this will be the first, second or third wythe. Two factors determine which wythe is chosen: the preferred placement of these rows and the unity check, see also 6.5. First the unity check determines whether or not it is possible at all if the row of bricks can be placed. Secondly, if placement at multiple wythes is possible, the preferred placement is used. The combination of the visual progress in the design model, see figures 7.2a, 7.3a, 7.4a, 7.5a and 7.6a, and the comparison overview of all strengths and stresses, see figures 7.2b, 7.3b, 7.4b, 7.5b and 7.6b, ensure that these two factors can be verified to work. For even further analysis, chapter 11 not only shows this graph of each stress and strength, but also shows their resulting unity check. Since this reduces the clutter, two additional stresses/strengths of interest are shown: the other stresses/strengths at interfaces 1 and 2.

Figure 7.7 is an additional visualisation to figure 7.6. Here, the stress diagram at $x = 0.64$ [m] is placed in figure 7.7b, of course rotated to align the axes representing the stress σ. Arrows are drawn between the two graphs to show how the position of the circled markings translates between them. It should also be noted that in this figure and in figure 7.6 one may see that the lines/areas stop earlier than the bricks in the design model. To restate: the structural analysis is done based on the centroids of the bricks and are assumed to be a point force load, instead of a distributed load. Thus, stresses and strengths develop in a certain cross-section only after the centroid of the row of bricks in that wythe has surpassed that cross-section.

7.1. Preferred placement of the bricks

The construction sequence is based on the unity check per wythe for the next placement, with thereafter additional preference of placement. How the unity check is calculated can be found in chapter 6. The preferred placement of the bricks is either based on an optimization analysis or on a set of rules. Since the optimization analysis is outside the scope of this research, a set of rules is put in place to determine the preferred placement.

The first rule for the preferred placement has to do with practicalities and with the initial optimal stress accumulation. The robot needs to be able to place the bricks appropriately. Here it makes sense to do either of the following:

- Inside out - place the inner wythe first, the middle wythe second and the outer wythe last. Using the indices of the wythes so far that would be $0 - 1 - 2$.
- Outside in - the same as the previous, but reversed. This gives $2 - 1 - 0$.
- Middle outwards - start with the middle wythe and place on both bottom and top. This results in both $1 - 0 - 2$ and $1 - 2 - 0$. More wythes increases the number of possible wythe orders.

Based on the figures 7.2 to 7.6, a couple of principles can be found regardless of the work order.

1. The stresses at a certain cross-section are a summation of the stresses due to each row of bricks placed thereafter and when the second and third wythe were placed.
2. The stresses at a certain cross-section are symmetric, but will be skewed since each wythe is placed at a different moment, resulting in a difference in summation.

It is clear the order in which each wythe is placed determines the stress distribution in that cross-section. As can be seen in the figures 7.2 to 7.6, in the cross-sections where only one wythe is present the stress distribution is symmetric. Meaning the maximum tensile and compressive stresses are equal and on the outer bounds. A stress of zero is at the middle of the wythe's height. Additionally, when a second wythe is present in that cross-section, this distribution will not alter further. It is only due to the addition of the stress distribution with two wythes (and later three) that the total stress distribution of the cross-section changes. This is shown with equation 7.1. This equation does not show the quantity of the stresses based on their position in the cross-section. This is explained in section 6.4 and table 6.1. As can be seen in this table, each interface may have a different sign based on the number of wythes at that cross-section. This means that based on the order the wythes are placed, the

signs and magnitude per interface may change.

$$\sigma_i = \sum_{i=n_a}^{n_b-1} \sigma_{i_a} + \sum_{i=n_b}^{n_c-1} \sigma_{i_b} + \sum_{i=n_c}^{n} \sigma_{i_c} \tag{7.1}$$

where:

i = number of rows of bricks placed;
n_a = the i^{th} brick placed when the cross-section becomes one wythe thickness;
n_b = " " becomes two wythes thickness;
n_c = " " becomes three wythes thickness;
n = total number of rows of bricks placed;
a, b, c = the first, second and third wythe to pass this cross-section.

The magnitude, and sometimes the sign as well, of the stresses per interface are dependent on the order of wythes. Table 6.1 shows the factors for the situation *Inside out*. If *Middle outwards* is chosen (with wythe$_0$ as second), only the first column changes with -1 at position 1 and 1 at position 2. The other two scenarios have the same factors, but reversed. If these are compiled together, the equations (7.2) to (7.5) can be made. σ_0 and σ_3 are opposites of each other in each scenario. If σ_3 has to be minimized, equations 7.2 and 7.3 are best to be used. For σ_0 the reverse is true. Similarly, σ_1 and σ_2 are each others opposites. If σ_2 had to be minimized, equations 7.2 and 7.4 are the best, depending on which is larger: n_a or n_b. A couple can also be found. If the tensile stresses in general have to be reduced, σ_2 and σ_3 are most likely the first to hit those, while the opposite is true for compressive stresses.

σ_1 and σ_2 are dependent on the ratio between n_a, n_b and n_c for their sign and therefore their value. This means that concluding anything for these two is dependent on the case specific. Therefore, speaking in general situations, it is assumed σ_1 is couple with σ_0 in the compressive stresses and σ_2 with σ_3 for the tensile stresses. Using this and the other parameters like the strengths, additional principles can be found.

3. Where tensile stresses are the determining factor, scenario *Inside out* is best, where compressive stresses are, *Outside in* has to be chosen;

4. Based on the literature in chapter 5, the tensile strength is much lower after completion than the compressive strnegth;

5. The tensile and compressive strength are comparable after the initial setting time;

With the tensile strength as the weaker of the two, it is preferred to use the *Inside out* scenario $(0 - 1 - 2)$. This results in the lowest tensile stresses for the outer interfaces, which coincides with the lower tensile strength. Of course, when applied on a project case, the specific circumstances may result in a different scenario. These specific circumstances are considered unlikely though.

$$\begin{aligned}
\sigma_0 &= -1 \cdot n_a \cdot \sigma_{avg,a} - 1 \cdot n_b \cdot \sigma_{avg,b} - 1 \cdot n_c \cdot \sigma_{avg,c} \\
\sigma_1 &= 1 \cdot n_a \cdot \sigma_{avg,a} + 0 \cdot n_b \cdot \sigma_{avg,b} - \frac{1}{3} \cdot n_c \cdot \sigma_{avg,c} \\
\sigma_2 &= 1 \cdot n_b \cdot \sigma_{avg,b} + \frac{1}{3} \cdot n_c \cdot \sigma_{avg,c} \\
\sigma_3 &= 1 \cdot n_c \cdot \sigma_{avg,c}
\end{aligned} \tag{7.2}$$

$$\begin{aligned}
\sigma_0 &= -1 \cdot n_b \cdot \sigma_{avg,b} - 1 \cdot n_c \cdot \sigma_{avg,c} \\
\sigma_1 &= -1 \cdot n_a \cdot \sigma_{avg,a} + 0 \cdot n_b \cdot \sigma_{avg,b} - \frac{1}{3} \cdot n_c \cdot \sigma_{avg,c} \\
\sigma_2 &= 1 \cdot n_a \cdot \sigma_{avg,a} + 1 \cdot n_b \cdot \sigma_{avg,b} + \frac{1}{3} \cdot n_c \cdot \sigma_{avg,c} \\
\sigma_3 &= 1 \cdot n_c \cdot \sigma_{avg,c}
\end{aligned} \tag{7.3}$$

$$\begin{aligned}
\sigma_0 &= -1 \cdot n_c \cdot \sigma_{avg,c} \\
\sigma_1 &= -1 \cdot n_a \cdot \sigma_{avg,a} - 1 \cdot n_b \cdot \sigma_{avg,b} - \frac{1}{3} \cdot n_c \cdot \sigma_{avg,c} \\
\sigma_2 &= 1 \cdot n_a \cdot \sigma_{avg,a} + 0 \cdot n_b \cdot \sigma_{avg,b} + \frac{1}{3} \cdot n_c \cdot \sigma_{avg,c} \\
\sigma_3 &= 1 \cdot n_b \cdot \sigma_{avg,b} + 1 \cdot n_c \cdot \sigma_{avg,c}
\end{aligned} \tag{7.4}$$

$$\begin{aligned}
\sigma_0 &= -1 \cdot n_c \cdot \sigma_{avg,c} \\
\sigma_1 &= -1 \cdot n_b \cdot \sigma_{avg,b} - \frac{1}{3} \cdot n_c \cdot \sigma_{avg,c} \\
\sigma_2 &= -1 \cdot n_a \cdot \sigma_{avg,a} + 0 \cdot n_b \cdot \sigma_{avg,b} + \frac{1}{3} \cdot n_c \cdot \sigma_{avg,c} \\
\sigma_3 &= 1 \cdot n_a \cdot \sigma_{avg,a} + 1 \cdot n_b \cdot \sigma_{avg,b} + 1 \cdot n_c \cdot \sigma_{avg,c}
\end{aligned} \tag{7.5}$$

Additionally, the placement of the robot is important as well. When the robot places the bricks from the outside (or top), *Inside out* is the preferred scenario anyways. With the robot on the inside (or beneath), problems may arise with reachability near the end. It could be useful in that case to switch to *Outside in*, even if it is only for the last rows of bricks. Considering the previous reason due to the stresses, it is best to start with *Inside out*. but a switch near the end will have little influence on this reason.

The second rule also has to do with practicalities and can be seen as a continuation of the first

rule. The order of wythes is established, which determines which wythe crosses a cross-section first. However, the row of bricks have a different length. This could result in an overlap of the following wythes on their preceding wythe (wythe 2 follows after wythe 1 and wythe 0 precedes wythe 1 in *Inside out*. Thus, the end of a possible row of bricks to be placed may not extend further than the start of the next row of bricks to be placed in the preceding wythe. Figure 6.8 gives an example of this. With *Inside out*, the yellow coloured bricks cannot be placed as long as the cyan coloured bricks haven't. The same goes for cyan and green. Thus, the preceding wythes will always extend further than their following wythes.

These two rules say when it is possible to switch to the next wythe, but they do not mention when. It stands to reason that the first option is to place the next wythe as soon as possible. To go back to figure 6.8, the placement of the green row in wythe 0 is followed by the placement of the cyan row in wythe 1, instead of the placement of another row in wythe 0. The reason this is useful is based on the factors of the normal stress. This stress is based on the normal force and the moment. The latter of these is much greater than the first. As equation 6.12 shows, the thickness of the vault is a major factor in the stresses due to the moment. Thus, it is best to have the greatest thickness as soon as possible. This is done by placing the next or following wythe as soon as rule 1 & 2 allow it. This option is also shown in figures 7.2 to 7.6.

The second option is based on the stress development found in those same figures. As can be seen in figure 7.6b, σ_3 will soon hit the tensile strength capacity at the support, even with the preferred placement in favour of σ_3! Thus, to lower this stress, it is better to place the last wythe as late as possible. This results in an increase of stresses at the other interfaces. Both σ_0 and σ_2 seem to be the next limiting place in this scenario. If the third wythe is not built, both will increase with the same amount, see in equation 7.2 the terms with n_b. As figure 7.6b indicates, σ_2 has roughly 1,5 MPa capacity left before reaching the tensile strength, while σ_0 has roughly -4 MPa left. Thus, if σ_3 is relieved, σ_2 becomes the next limiting factor.

The rows of bricks placed have the same coordinates with both options. This means that the stresses σ_{avg} due to the placement of a row do not change with the second option. However, n_b and n_c do change. Therefore, an estimation of the decrease in stresses for σ_3 and the increase for σ_2 can be stated, see equation 7.6. This estimation does not take into account that both σ_{avg} from the previous equation will change as well. Thus, the estimation is an approximation.

$$\begin{aligned} \Delta\sigma_2 &= \frac{2}{3} \cdot \Delta n_c \cdot \sigma_{avg} \\ \Delta\sigma_3 &= \Delta n_c \cdot \sigma_{avg} \end{aligned} \tag{7.6}$$

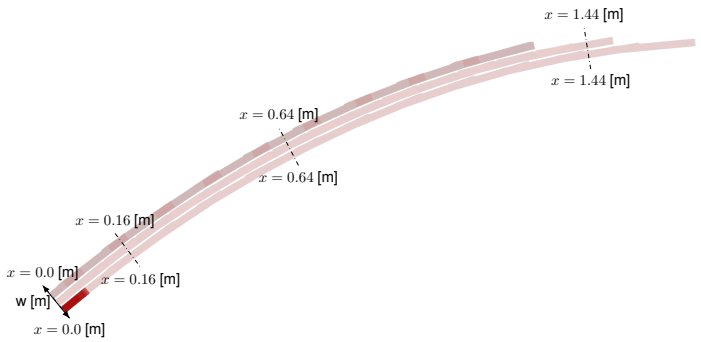
Additionally, the change with this option is only when the first row of bricks will be placed. The later rows in the last wythe can only be placed after the first row in that wythe has been placed. Furthermore, the maximum stresses in the last wythe in the first option are at the support, which remains so for the second option as well. Thus, if the second option aims to delay σ_3 for reaching the tensile strength, the only 'when' that matters is when the first row of the last wythe is placed. The equation 7.6 contains a lot of numbers which may vary per configuration. It is not possible to give a general formula or indication how much Δn_c should be, for the second option to be of maximum use. However, because the second option results in the last wythe entirely to be placed last, it isn't hard to investigate this maximum with trial & error.

Without the last wythe the maximum reach is when σ_2 has exceeded the tensile strength. By placing the last wythe earlier than that, the exceedance of σ_2 is postponed. This lowers the stresses in σ_2 in the last possible row of bricks to be placed and increases it in σ_3. This should also result in more possible bricks to be placed. This continues until σ_3 has exceeded its strength. Either this last configuration or the one before (where σ_2 was the exceeder) will have the most possible bricks to be placed.

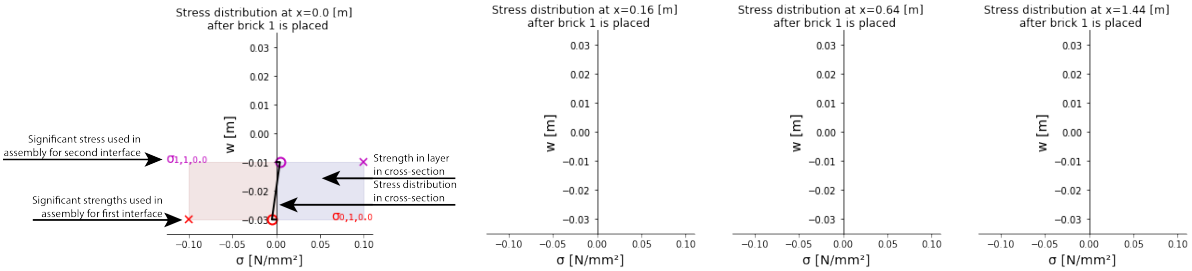
A third option could be to extend the second option to the first and second wythe as well. Here the implementation is much simpler. Since the addition of the second wythe does not change the value of σ_1 (see equation 7.2, second term), it is not a question of when the maximum stress occurs at σ_1. However, the development of the strength is more significant here. Thus, for this option it is more important to place in the second wythe when the first would result in exceedance of strength anywhere in the structure. It seems this allows for even more bricks to be placed. Thus, this is investigated when option 2 is also explored. Both result in a placement of the following wythe as late as possible.

This gives two preferred placements beyond rule 1 & 2. The third rule is the placement of the following wythe to be either as soon as possible,

or as late as possible. In the computations this is abbreviated to w_{rule} = a.s.a.p. or w_{rule} = a.l.a.p. See figures 11.2 and 11.17 for a visualized construction sequence.

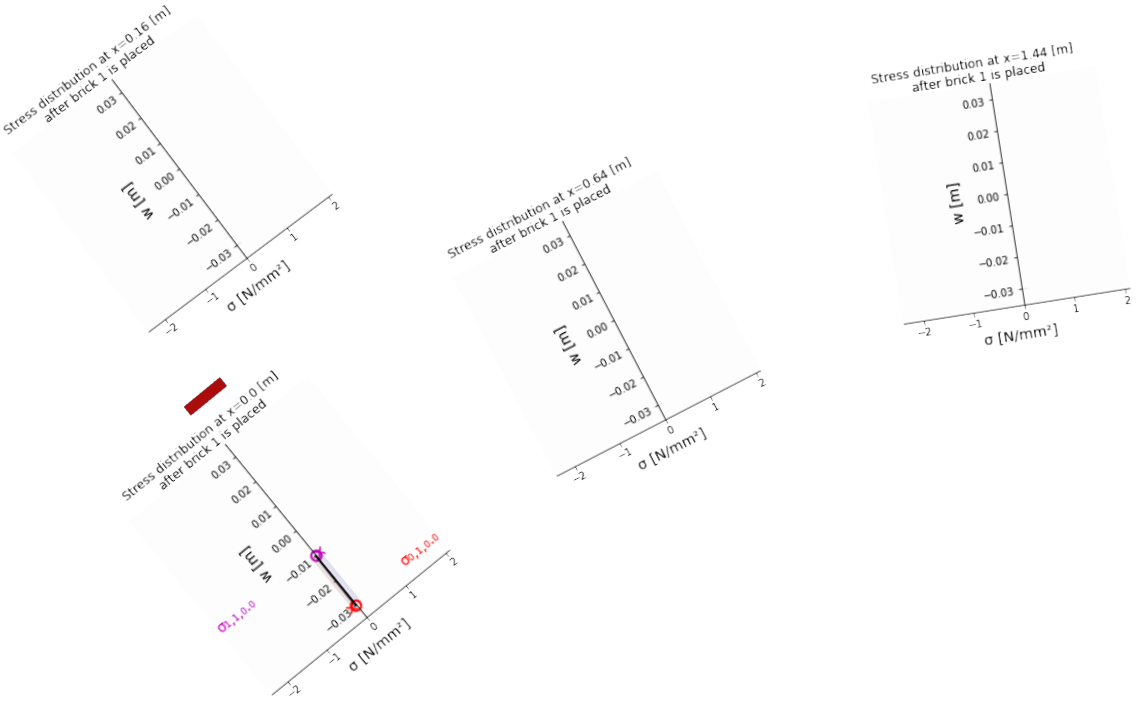


(a) The position of the cross-sections of $x = 0,0$meters, $x = 0,16$, $x = 0,64$, $x = 1,44$; the stress diagrams shown after are based on 1 brick placed, hence the dark-colored part; the shaded part is the rest of the structure, to be constructed, and shown for clarity;

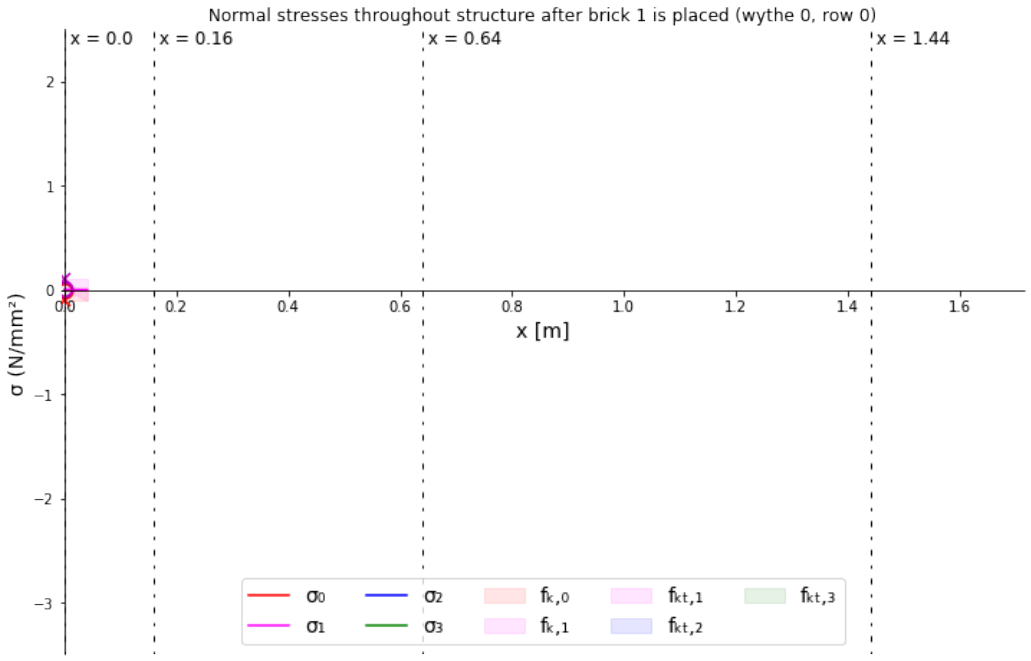


(b) The stress distributions at the cross-sections of $x = 0,0$meters, $x = 0,16$, $x = 0,64$, $x = 1,44$ when 1 brick has been placed, also showing the significant strength and stress values used in later figures;

Figure 7.1: The stresses in the structure after 1 brick has been placed, where both the stress distribution in four cross-sections is shown and the stresses along the entire cantilever.

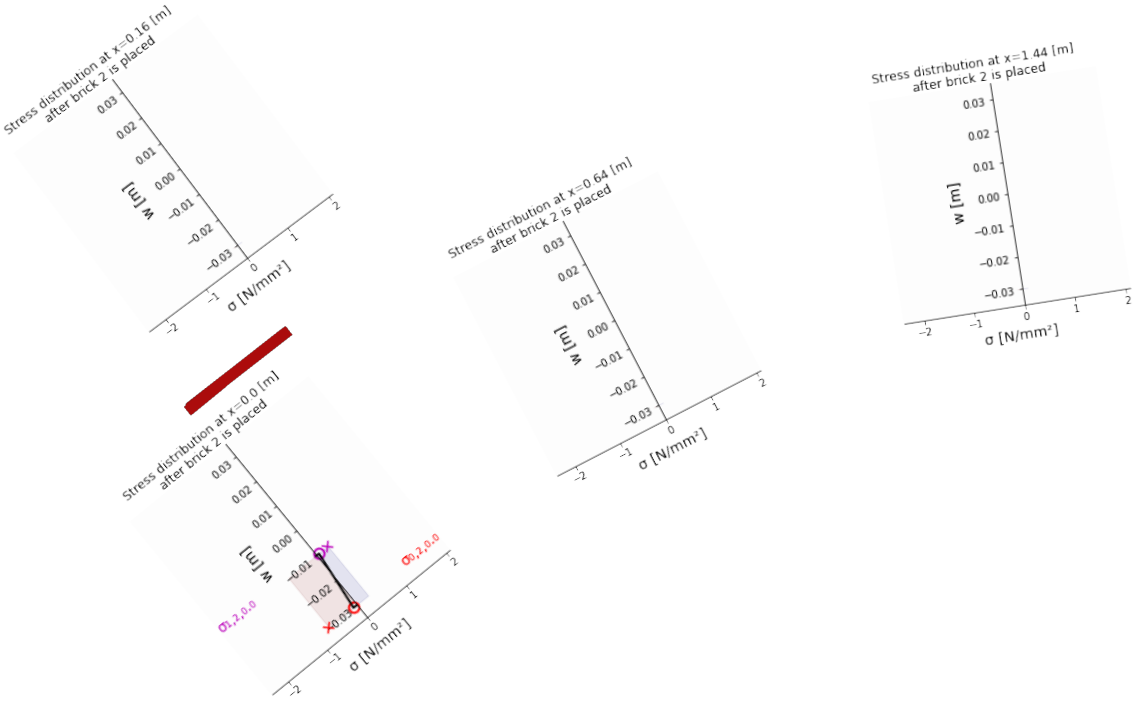


(a) The stress distributions at the cross-sections of $x = 0,0$meters, $x = 0,16$, $x = 0,64$, $x = 1,44$ when 1 brick has been placed;

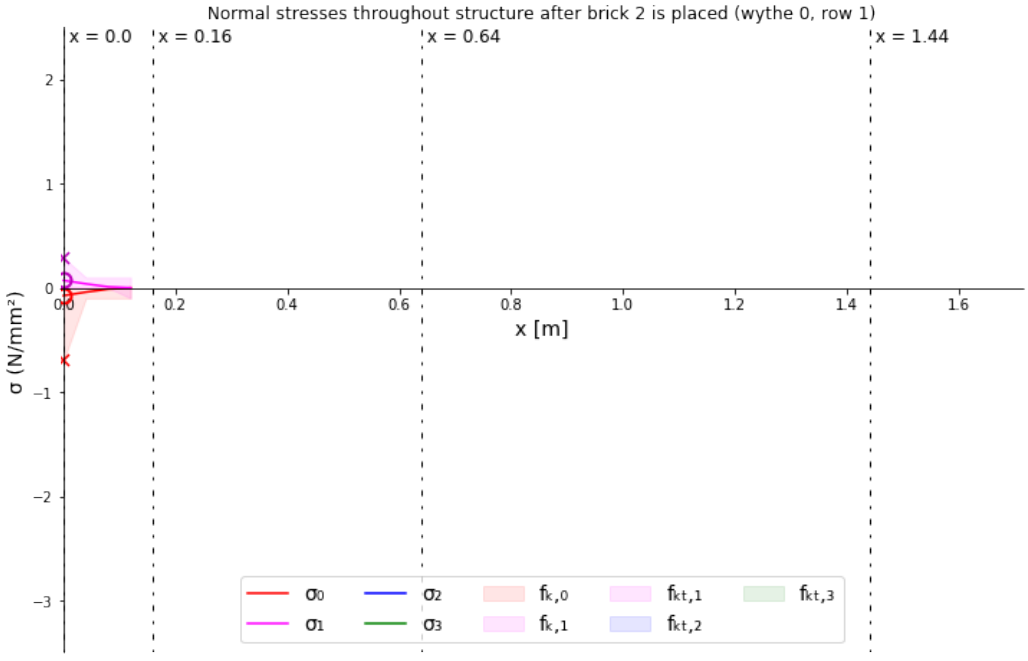


(b) The stresses at the four interfaces as indicated in figure 7.2a when 1 brick has been placed;

Figure 7.2: The stresses in the structure after 1 brick has been placed, where both the stress distribution in four cross-sections is shown and the stresses along the entire cantilever.

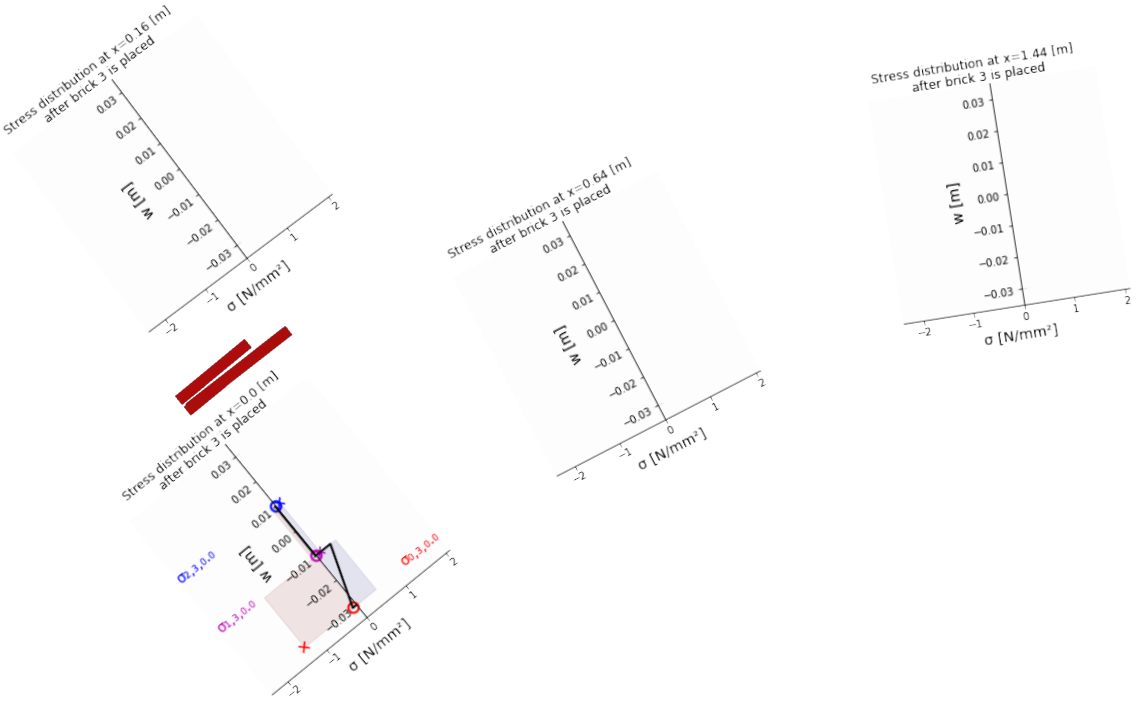


(a) The stress distributions at the cross-sections of $x = 0,0$meters, $x = 0,16$, $x = 0,64$, $x = 1,44$ when 2 bricks have been placed;

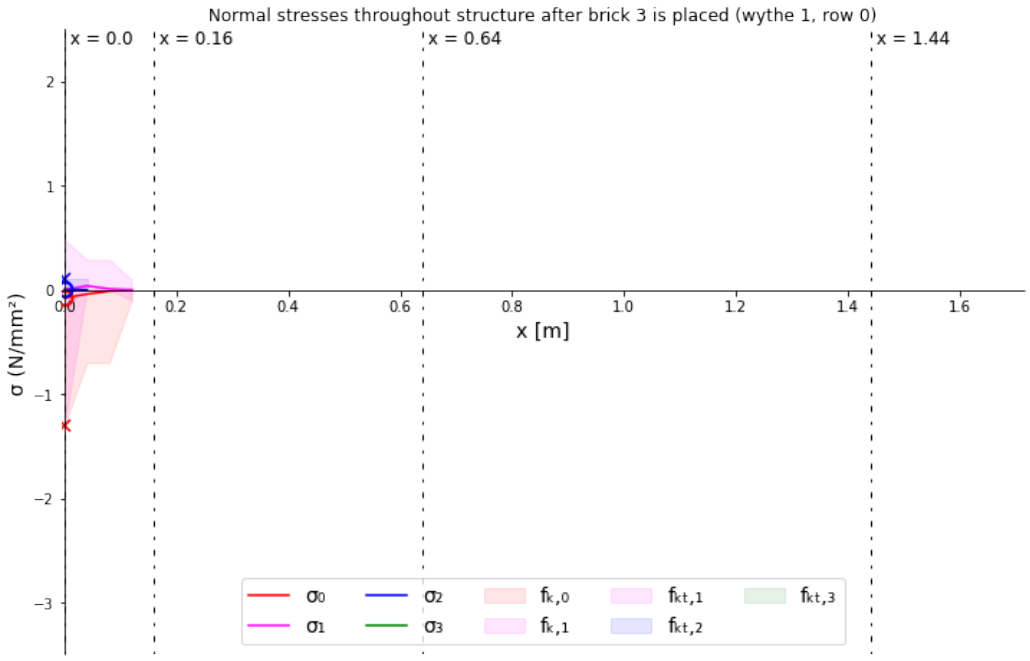


(b) The stresses at the four interfaces as indicated in figure 7.3a when 2 bricks have been placed;

Figure 7.3: The stresses in the structure after 2 bricks have been placed, where both the stress distribution in four cross-sections is shown and the stresses along the entire cantilever.

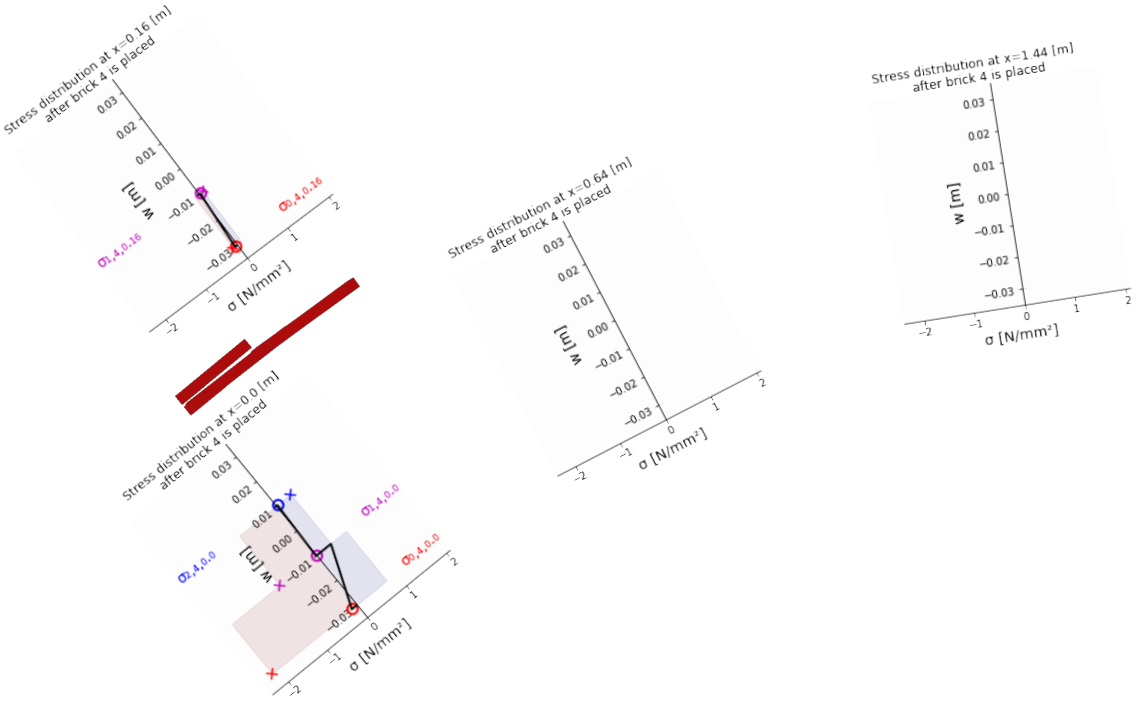


(a) The stress distributions at the cross-sections of $x = 0,0$meters, $x = 0,16$, $x = 0,64$, $x = 1,44$ when 3 bricks have been placed;

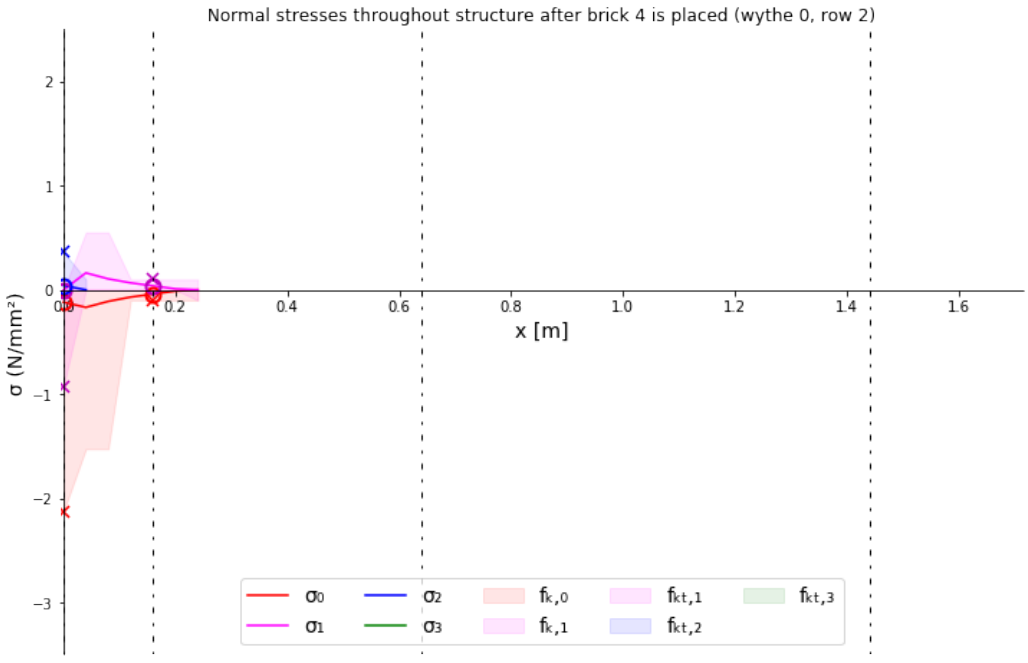


(b) The stresses at the four interfaces as indicated in figure 7.4a when 3 bricks have been placed;

Figure 7.4: The stresses in the structure after 3 bricks have been placed, where both the stress distribution in four cross-sections is shown and the stresses along the entire cantilever.

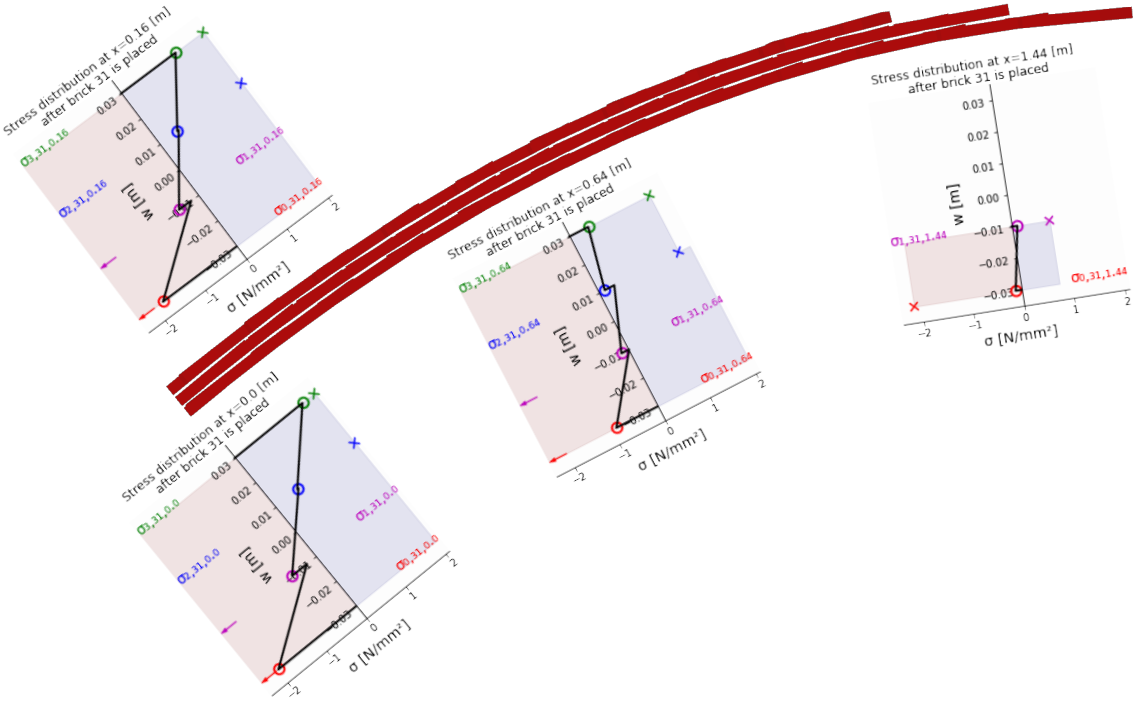


(a) The stress distributions at the cross-sections of $x = 0,0$meters, $x = 0,16$, $x = 0,64$, $x = 1,44$ when 4 bricks have been placed;

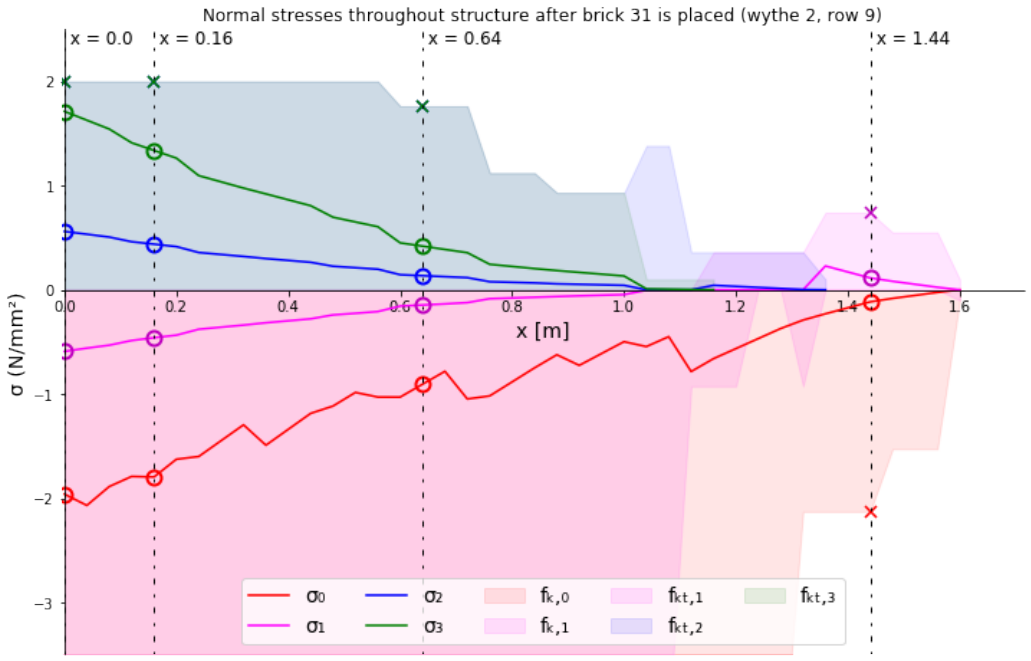


(b) The stresses at the four interfaces as indicated in figure 7.5a when 4 bricks have been placed;

Figure 7.5: The stresses in the structure after 4 bricks have been placed, where both the stress distribution in four cross-sections is shown and the stresses along the entire cantilever.

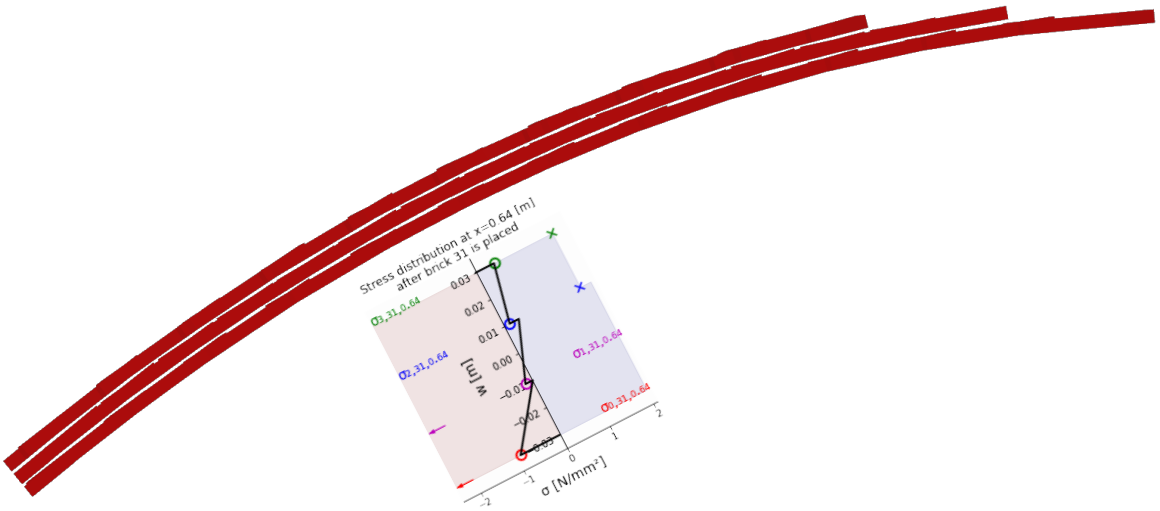


(a) The stress distributions at the cross-sections of $x = 0,0$meters, $x = 0,16$, $x = 0,64$, $x = 1,44$ when 31 bricks have been placed;

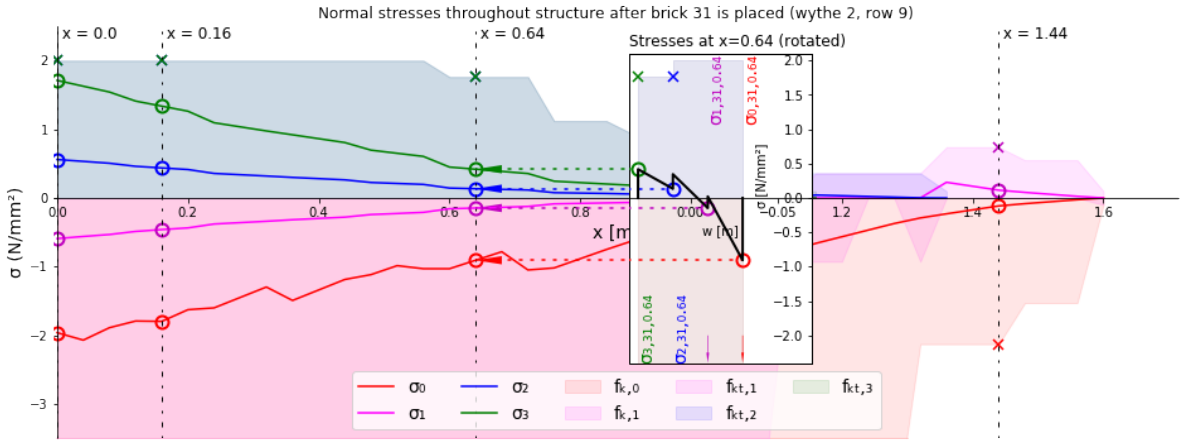


(b) The stresses at the four interfaces as indicated in figure 7.6a when 32 bricks have been placed;

Figure 7.6: The stresses in the structure after 32 bricks have been placed, where both the stress distribution in four cross-sections is shown and the stresses along the entire cantilever.

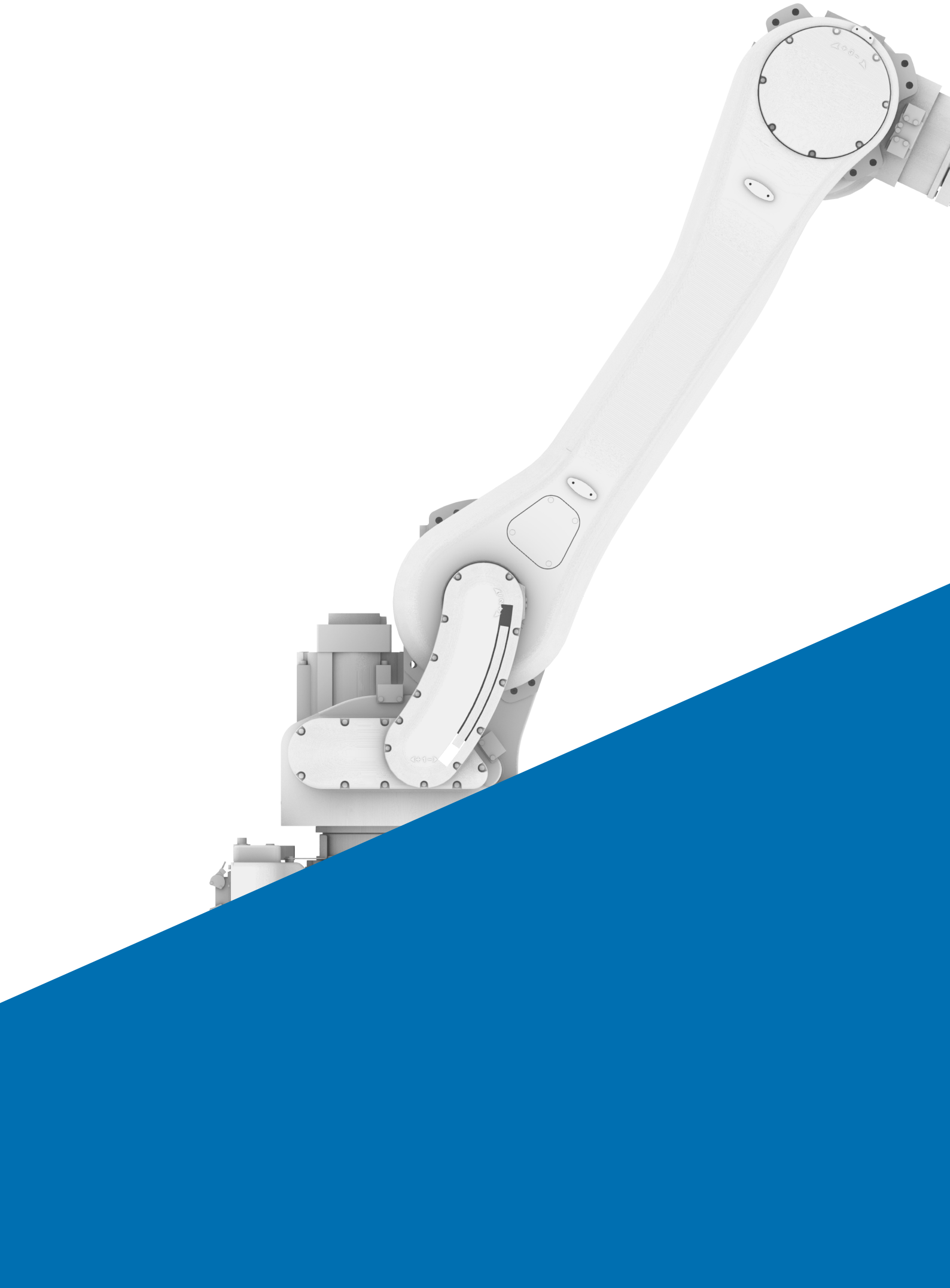


(a) The stress distribution at the cross-sections of $x = 0,64$ when 31 bricks have been placed, which serves as an example of how the stresses of the stress diagrams translate to the stresses in figure 7.7b;



(b) The stresses at the four interfaces as indicated in figure 7.6a when 32 bricks have been placed;

Figure 7.7



Part IV

Robotics and the work process for thin-tile vaults

8

Literature on robotics

A robot is a machine controlled by a computer that is used to perform jobs automatically (Cambridge, 2020). This definition shows three components that are required to use a robot: a code, a machine and a tool. Tools come in many shapes and sizes, each designed for the job at hand. Machines can also differ quite a lot, with machines for robotic systems having its own design and functions. The code to operate these machines is usually very similar. However, program coding suited for robots differs from coding for other applications, like the computer. In this chapter this system is further looked into, how it has been applied in the construction industry and how this influences the construction of a masonry vault.

8.1. Robotic typologies

The implementation of robotics in architecture is relatively late compared to other industries (Estrella, 2017). However, in the past decades the digitalisation of the building industry has taken flight. Since the 1970's CAD (Computer-aided Design) has been greatly implemented in the building process, creating the field of digital architecture. Since the 1990's and especially the early 2000's the gap between design and manufacturing within digital architecture has decreased, with the introduction of robots and other machines (Estrella, 2017). This is also known as CAM (Computer-aided Manufacturing) or digital fabrication. Nowadays robots can be applied in a wide range of applications: design processes, construction processes, single-tasks robots and dynamic surfaces/living architecture. Within these applications, four most commonly used manufacturing methods can be categorised: cutting, addition, formation and subtraction (Estrella, 2017). This categorisation is not definitive though. Joining and Generative may be used as well (Bidgoli, 2015). However, the ISO has (so far) only defined additive, subtractive and formative shaping, stating the manufacturing process can consist of one or a combination of these principles (ISO/ASTM, 2017).

Table 8.1: Most common robots applied in architecture (Estrella, 2017)

CNC machines	Subtractive processes Formative processes Cutting Turning Milling
Robotic Arms	Formative processes Gluing Melting Drilling Cutting Pouring Handling
3D-Printers	Additive processes

The robotic machines commonly used in the construction industry can be grouped in three categories: CNC machines, robotic arms and 3D-printers. CNC stands for Computer Numerical Control. An overview of the applications per group can be found in table 8.1. It should be noted that this list is limited to robots applied in manufacturing processes. This excludes robots or processes for other purposes, i.e. to acquire data with 3D-scanners. A robot that combines such processes, though, is not unimaginable. Since robotics, and especially digital fabrication, is still a developing field, it is uncertain though how long the distinction from table 8.1 is applicable. Already, combining a robotic arm with CNC or 3D-printing is possible within software (RoboDK, 2021).

With digital architecture, CAD and CAM work together to create structures and projects with digital fabrication. CAD-software is used

to generate the design, while CAM-software converts that design to hardware code (Estrella, 2017). The conversion is needed to change from file formats commonly used in CAD (i.e., .dwg) to formats used in CAM (i.e., STL). Examples of CAD-software for additive processes are Rhino3D, Blender, and SketchUp (Estrella, 2017). As explained before, Rhino3D (with Grasshopper) can be expanded upon by its users with the help of plugins. Some companies have started to bridge the gap between CAD-software and robots by implementing a CAM-plugin. Examples are BrickDesign (ROB Technologies, additive), KingKong (Robofold, formative), Unicorn (Robofold, subtractive), IO (Robofold, robotic arms) and RoboDK (RoboDK, robotic arm) (Estrella, 2017), (RoboDK, 2021), (RoboFold, 2013).

The reach of a robotic arm rarely exceeds 3.5 meters (Bidgoli, 2015) (RoboDK, n.d.). Considering the reach of the arm as the radius of a perfect sphere, the work space cannot be larger than 7 meters across. Other factors may downsize this further: the robots reach is not a perfect sphere (similarly to how an arm cannot reach all positions within its length) and the pose of the tool (orientation and position) may not allow a fully extended arm.

8.2. Construction robotics

8.2.1. SAM100

One of the first robots used in masonry projects is the SAM100, where SAM stands for Semi-Automated Mason. The robot consists of a generator, lasers and sensors, a conveyor belt, a concrete pump and a robotic arm with gripper (Madsen, 2019). Companies that have purchased and used the robot claim the speed of brick-laying from human masons to the SAM100 goes up from 400 to over 2.000 bricks a day or similar increases (Madsen, 2019). Similarly, it seems people are more content with the end result, the high quality, the robot provides. Besides, the number of required workers on site can be reduced where the masons are now assisting the robot in its construction process (Madsen, 2019). SAM100 still needs assistance to operate efficiently. The mortar needs to be mixed and bricks need to be cut and fed to the robot. Someone has to operate the machine as well and remove the excess mortar after placement.

The application of the robot are limited, though. The costs are high ($500.000 to buy and $20.000 per month to rent) and the high speed increase in brick-laying is only achieved at certain types of walls and *after* the robot has been set-up. Preferably, the walls are long, straight and high, like in warehouses. The construction time is reduced significantly, but when the construction site is complex (for instance, not on the ground floor), the set-up time can nullify this effect (Madsen, 2019).

8.2.2. Hadrian X

Still in development, the Hadrian X is a robot developed with a different approach to brick-laying. Different to the SAM100, this one is fully independent and includes the feature to determine the full brick laying sequence of the structure (Bogue, 2018). It operates from a fixed location and is even capable of cutting the bricks and applying either mortar or adhesive to the bricks under pressure.

8.2.3. Other construction robots

Students at the Bartlett School of Graduate Studies in London have created a robot program for bricklaying. The small-scale test included monitoring masons for the proper technique and a full process of bricklaying: the program is able to pick and place the bricks; switch to the appropriate tool or end effector; lay mortar and remove excess mortar (Bloss, 2014b).

At the Massachusetts Institute of Technology (MIT) a team combined a lorry with a robotic arm. With a reach of 24 meters and a payload of 680 kilograms, the robot has eleven axes of freedom. The lorry is a mobile with five axes and attached to its end is a KUKA robotic arm with six axes (Bloss, 2014b).

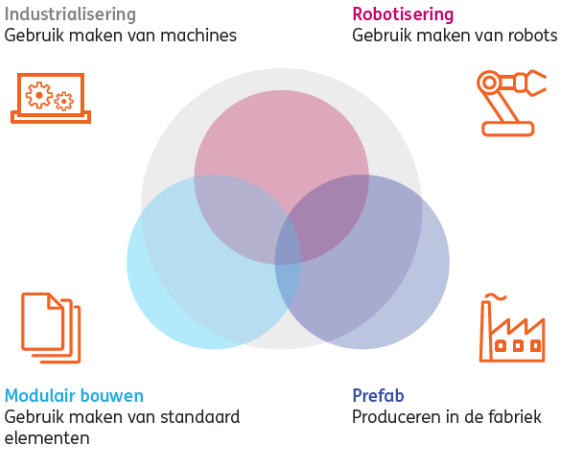
8.3. Dutch robotic/construction industry

8.3.1. ING

ING Economisch Bureau is the economic research department of the ING Group, one of the biggest banks in the world. Their publication in May 2020 investigated the industrialisation of the construction industry (Van Sante, 2020). The publication reports industrialisation as 'the mechanisation of the production processes in which manual labour is done by machines to increase the labour productivity'. The publication continues with relating industrialisation of the building industry to other terms from the construction industry: modular construction, prefab and robotisation. As can be seen in

8.1 the department considers robotisation in its entirety a subset of industrialisation, whereas modularity and prefab can also be applied outside the industrialisation.

(Van Sante, 2020) compares the robotisation of the Dutch construction industry with other industries and with other countries. Compared to other industries, robots are used little in construction (roughly 150 to 1). Compared to other countries the Benelux is leading in the robotisation in the building industry (in both comparisons the Netherlands is grouped with Belgium and Luxembourg), even more than Japan, a country known for its robotics. Robots can increase the industrialisation even further, especially combined with prefabrication at indoor facilities. The publication does emphasize that a part for the high robotisation in the Netherlands can be attributed to the high share of repetitive buildings (like row houses) and the high labour costs.



* Robotisering valt volledig binnen industrialisering. Modulair bouwen en prefab zijn deels een verzameling van industrialisering. Zo hoeft modulair bouwen niet per definitie te gebeuren met machines en kan prefab ook in een fabriek met de hand in elkaar gezet worden. Digitalisering speelt bij de verschillende begrippen vaak ook een rol en is vaak ook een eerste voorwaarde.

Bron: ING Economisch Bureau

Figure 8.1: Overlap of industrialisation with modular construction, prefab and robotisation. | ING Economisch Bureau (Van Sante, 2020)

9 Stations

The robot will move from station to station and perform a routine of instructions. Each station has been developed in the parametric model first, to set the proper information for the robotic simulation software. See section 10.2 for more about the proper information. Important in robotics is the use of reference frames or coordinate systems. In general, six reference frames are essential for this construction. The first reference frame is the world coordinate system, all other reference frames are relative to this one. This reference frame is also vital outside of robotic construction, since this frame is also the starting point in the real world. The second reference frame is that of the robot itself. Positioned at the base of the robot, it is commonly known as the base coordinate system. Thirdly, the reference frame of the tool or end effector is used: the tool coordinate system. These two reference frames essentially determine the shape the robot takes (in the case of a robotic arm in which angle each axis is positioned). The fourth reference frame is from the export program. It contains all the objects the robot has to be aware of. In industrial applications this can be the surface on which the operations are done, also known as the user coordinate system. Within this reference frame are two other, more variable, reference frames. One is the reference frame of the object it holds, the object coordinate system. The other is of the target position, to which the robot has to move. In figure 9.1 four of these reference frames are presented. The terms used come from ABB robotics.

The export from the parametric model provides a general user coordinate system for all objects. However, within this system, three stations are present. In this situation a station is a defined space in which the same routine of instructions takes place, although with different objects. The three stations here are the pallet station, the adhesive station and the vault station.

9.1. Pallet station: the source of the bricks

The pallet station is the area in which the objects are assumed to be upon delivery and ready for construction. As per the scope of this research, the objects are assumed to be of the correct shape before construction. Thus, this creates three initial conditions for the construction.

- The objects are assumed to be of the correct shape and finishing. No additive, subtractive or formative manufacturing processes have to happen to the object.
- The objects are assumed to be delivered on a standard pallet. In the program a Europallet is used with dimensions of 1200×800 millimetres.
- The packing arrangement of the objects is similar to that of a pallet filled only with whole bricks. This means, there's room around the cut bricks. It is possible to apply other packing arrangements, but due to simplification it is done with this arrangement. See figure 9.2.

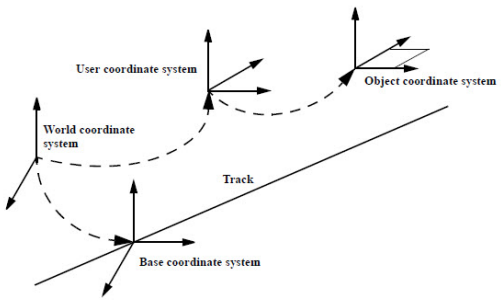


Figure 9.1: Four of the six described coordinate systems and to which reference frame they are related to | (*Technical reference manual - RAPID Overview*, 2019)

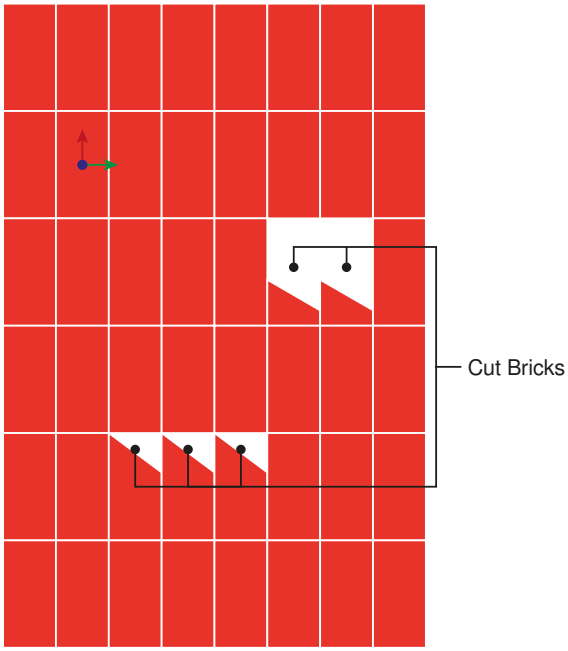


Figure 9.2: Packing arrangement of the bricks in the pallet station

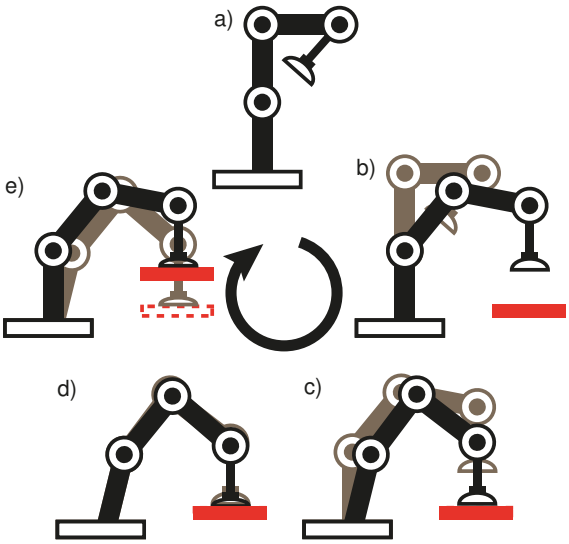


Figure 9.3: Routines at pallet station (elements not to scale): a) initial situation; b) approach to object; c) contact with object; d) attachment of object; e) retreat with object.

The objects in this station contain two items of information: the object itself and an object coordinate system. This is why the packing arrangement does not really matter, as long as this information is set. The object is required for visualisation in the virtual environment and the parametric model has some functionalities that are more intuitively when working with the object as well. The reference frame of the object is set as the projected point of the centroid on the top face of the brick, with the z'-axis directed upwards, see figure 9.2. Note that in this figure the local axes for a whole brick are shown, and that for cut bricks the centroid not aligns with the center of grid cell. Using the centroid has three benefits. It is easy to find the centroid (of a surface) in CAD software (with parametric modelling capabilities). The centroid will always be within the objects geometric boundaries, meaning the robot actually does grip onto something. Lastly, the centroid is the point of an object on which it balances in equilibrium. This ensures little additional forces and movements of the objects while being moved by the robot.

Important is the order in which the bricks are placed in the pallet station. An object is only reachable if its top surface is completely clear of any obstructions. The order is based on when the brick is constructed. In the parametric model this order is done after the construction sequence has been determined. The lists in the data tree containing the bricks in the vault has been reordered based on the construction sequence. This data tree of multiple lists is flattened to one list ('list with bricks'). The pallet station is a spatial grid of points positioned at the centroid of each grid cell. Again these points are collected in a flattened list ('list with points'). Each item (each brick) in the list with bricks is moved to each point in the list with points, including with the required rotation to ensure the stacking is done correctly. Which brick has to go to which point is based on a 'First in, Last out' ordering system. The bricks that are to be placed *last* are linked to the first points, the points at the bottom layer of the spatial grid. This not only ensures that the bricks are reachable when needed, but also provides the certainty that the packing arrangement is physically possible: the stack of bricks rests on the pallet and does not have a void beneath it.

With the bricks placed correctly in the pallet station, together with the transformation of the object coordinate system, the tool path algorithm has all the information required to reach each object correctly. At the pallet station five routines are repeated for each object. First the robot moves to a point slightly above the object, with the correct pose of the end effector or tool. Secondly, the tool is lowered to the origin of the object coordinate system. Thirdly, the tool is activated to connect the object to the robotic system. Fourth, a movement similar, but reversed, as the second routine is done to the point slightly above where the object used to be. Lastly, the object is moved away to the next station. The first and last routines

are the transitions from one station to the other. See figure 9.3 for this cycle.

9.2. Adhesive station: moving passed a glue gun

The adhesive station is the second station in the construction process. The adhesive station is the area where the adhesive, be it mortar, plaster or glue, is added to the brick. The current set up in the parametric and simulation models is suited for glue guns, for instance for epoxy. The adhesive station consists of two parts. The first is the glue gun, or whatever tool is used for the application of the adhesive. The second is space for the robot to rotate and move objects in proximity of the gun, to be able to orientate the correct faces to the glue gun. Again this station has a couple of boundary conditions.

- To limit robot movements, the adhesive is applied horizontally (the surface on which it is applied is vertical). This way the orientation of the objects taken from the pallet station stay constant (flat, with the end effector above the object, pointed downwards). Afterwards as well, the orientation of the objects does not change drastically. The movement mostly includes a (slight) tilt of the object, to orientate it correctly to the sloped surface of the vault.
- The adhesive is stationary and the objects are moved alongside it for application. This reduces the need of an assistant to the robot (either human or machine). It is likely this has little influence on the time it takes to apply the adhesive.
- The adhesive has an adequate high viscosity that it may smear slightly, but not for more than a few millimetres within the first minute, to ensure the adhesive does not droop from the object. Epoxy glues applicable for ceramics (bricks) that harden in a relatively small time are likely to fit this assumption.
- A scanner, laser or other measurement device is present next to the nozzle of the glue gun, to determine the moments to activate the gun.
- Only one type of adhesive is used in this model. If multiple adhesives (like the combination of epoxy/gypsum mortar and Portland cement) were to be used, it is best to have enough separation to ensure the scanners for each adhesive aren't activated, not even when just passing by in a robot movement.

The information for this station is set in three pieces. The first information is the attachment point of the object (the projected centroid on the top surface) to the robot. The second is the position of the glue gun. The last piece of information is the set of vector lines that represent the sides of the object to be glued. These vector lines consist of the length of the side and the direction relative to the centroid. This determines the orientation of the brick and the distance of the brick with the end-point of the glue gun. A couple of bricks will only have one vector line to be glued, while most will have multiple. The order of applying the glue to each vector line is continuous with how the robot has to rotate for the next vector line. In other words,

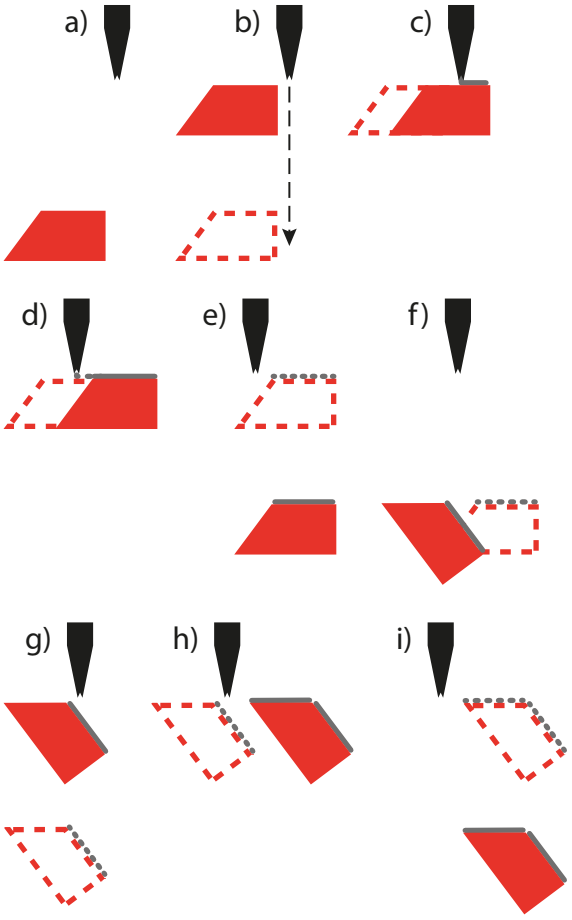


Figure 9.4: Routines at adhesive station (elements not to scale): a) approach to station; b) alignment to muzzle, with muzzle summetry axis shown; c) placement of adhesive; d) end of placement; e) retreat from station; f) approach with different orientation; g) alignment to muzzle; h) end of placement of adhesive; i) retreat from station.

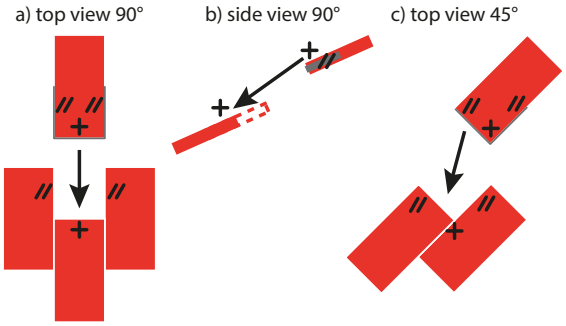


Figure 9.5: Approach path through 'sliding' (elements not to scale); '+' represents the area resisting more bending/tensile stresses, '//' represents the area resisting more shear force.

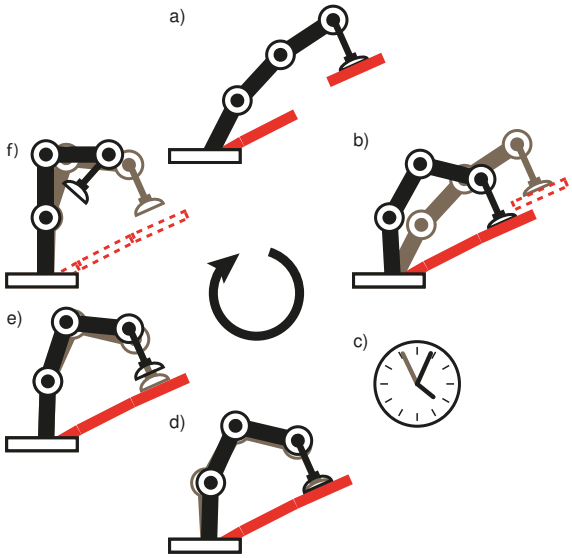


Figure 9.6: Routines at vault station (elements not to scale): a) approach to vault; b) place object; c) wait for hardening; d) detachment of object; e) retreat from object; f) back to initial position.

the end point of the previous vector line is the starting point of the next one. However, the robot retracts from the glue gun to rotate the object for the next vector line. This creates four routines per vector line.

With the first routine the object is moved closely to the glue gun, with the correct rotation. Its position is off-center, aligned with the starting point of the vector line. The second routine is to move the object parallel to the symmetry axis of the muzzle of the gun (see in figure 9.4b). The object is moved along this parallel line until it reaches the correct distance to the glue gun. This distance is dependent on the used adhesive. With most a millimetre or two should be fine, but a practise run is advised. The third routine is to move the object perpendicular to this muzzle axis. In this step the adhesive is applied. In the last routine the object is moved away from the glue gun. The next routine would either be to move the object to the next station, if it was the last vector line, or to move and rotate the object as described for the first routine. In figure 9.4 the full cycle for this station is shown, with two vector lines to be glued for this object. The total number of routines is $4 \cdot \#\ of\ \ vector\ \ lines$.

9.3. Vault station: the target of the bricks

The vault station is the third and last station in the process. The vault station is the area where the actual structure is constructed. The vault station is the first station to place in the environment. After all, this station is the goal of the whole process. The other two stations are placed around the structure. As with the previous two, this station has initial and boundary conditions.

- It is assumed the supports of the structure are in good condition for the vault. This includes a clean and dry foundation that is stiff, stable and strong. The foundation or supporting element(s) are designed in such a way that the objects placed onto the support will have, at minimum, the same structural mechanical properties as the vault structure itself.
- Besides the vault to be constructed, no other objects or structures are within the work space environment.

The vault station requires three sets of information. The first is, again, the attachment point of the object. The second is the position in the vault. The third piece of information is the existing structure. Similarly to the situation at the pallet station, the objects in the vault station could obstruct each other in the construction process. Although it should be able to construct each object, the tool path to the end position could be hindered by the existing structure. For this a special approach path to the end position is used. The objects are slided in their position. The benefit of sliding is that the adhesive will be less skewed after placing than would have been the case were it lowered from directly above (where 'above' is normal to the vaults surface/curvature). This skewness happens due to the friction between the objects surfaces and its neighbours. The adhesive will be distributed more to the top of the connecting surfaces than to the bottom due to this skewness.

Sadly it is not possible to remove this skewness entirely, but it is possible to control which surfaces

get what kind of skewness. By sliding, this skewness will be mostly concentrated to the sides of the objects. The most important surfaces to be the least affected by the skewness, are those resisting the tensile stresses occurring in the structure. To ensure the adhesive has the proper tensile stress capacity, the adhesive has to be predictable, with the least skewness present. In other parts of the surface the tensile stresses will be less, allowing for (slightly) more unpredictability. Thus, it is best to approach the end position in such a way that the (parts of the) surfaces experiencing the most tensile stresses will have the last contact with the other objects and their surfaces. A visualisation can be seen in figure 9.5 for the orientations of 90° and 45°.

With the defined approach path the routines at the vault station are as follows. The six routines are similar to those found in the pallet station. The first step is to approach the end position, based on the above described approach path. The second routine is to position the object in its end position. There, the robot has to wait for the adhesive to develop the required strength. The fourth routine is to release the object. At the fifth routine the robot retreats to a position above (where, again, above is related to the normal) the object. The sixth and last routine is to return to the initial position upon completion of the project, or to return to the pallet station for the next object. This is illustrated in figure 9.6.

9.4. Environment

In figure 9.7 the full work space during construction is shown. The pallet is the standard size of an EU-pallet. The robot is made to scale with a base of approximately 500 mm and a reach of 3.125 mm (radius of reach is 1.562 mm, see section 10.1 for the reason behind this reach). The vault can only have a maximum span of 1,7 meters with a length of 1,8 meters. Lowering this length will allow for a slightly longer span, but increasing the length will reduce the maximum span even more. In this scenario the robot only builds upto the apex of the vault (shown as a red dashed line) 0,85 meters away. This requires the robot to be moved to the other side to complete the build.

Another configuration is to place the robotic arm between the foundations. This way, the robot can reach vault spans of 2,6 meters. Sadly this reduces the length of the vault it can construct at once. If the robotic arm is placed on a (single-axis) track though, it would allow the

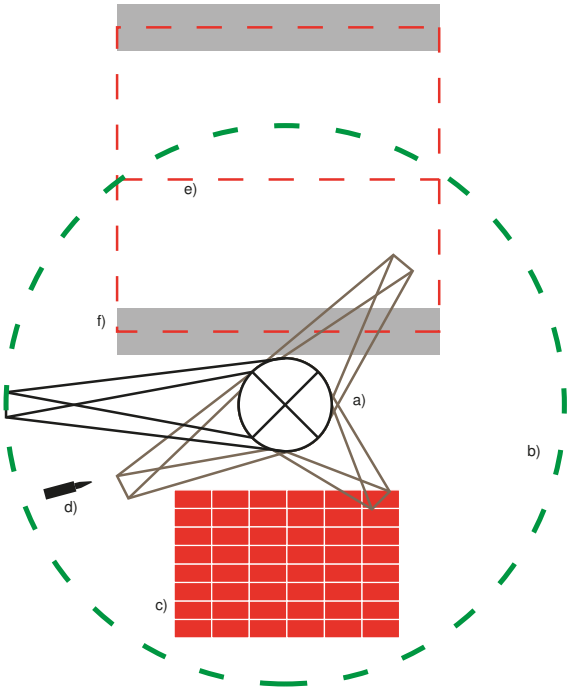


Figure 9.7: Work environment during construction (top view, configuration 1): a) robotic arm (in gray the arm at each station); b) the reach of the arm; c) Pallet Station; d) Adhesive Station; e) Vault Station (red outline is total vault, with the apex notated with a dashed red line as well); f) Foundation of vault.

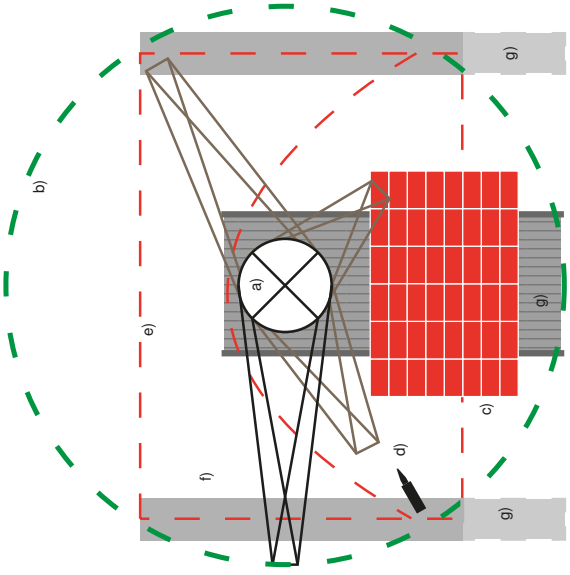


Figure 9.8: Work environment during construction (top view, configuration 2)): a) robotic arm (in gray the arm at each station); b) the reach of the arm; c) Pallet Station; d) Adhesive Station; e) Vault Station (red outline is total vault, with the constructable area shown to the right of the red dashed curve); f) Foundation of vault; g) Extendible length of vault due to the single-axis track.

Figure 9.9: Spatial robotic assembly at ETH Zürich | ETH Zürich

length of the vault to be as long as the track can be. Figure 9.8 shows this configuration, including the track. This way two construction strategies are possible. The first one is to build the vault with a stationary robot, where the robot is moved only when that part of the vault it can reach has been constructed (see the curved red line in figure 9.8). An advantage is that the vault has reached a stable structure quite quickly in the construction process. The brick pattern becomes more important though, since the method described in section 9.3 may not be possible in this configuration.

The second strategy is to build the first line of bricks (the ones placed on the foundation) first, where the coordination of the robot with the other stations and the track becomes much more complex. The advantage is that the construction sequence is very similar as to the first configuration in figure 9.7. However, the construction space becomes very narrow as construction reaches the apex. This also requires a precise calibration of the work space.

Of course, many more strategies for both configurations are possible. The first configuration can also make use of a track to lengthen the vault, while the second configuration can have the robot placed on wheels to provide dual-axis movement. Other configurations are possible as well. For instance something similar to the robotic assembly done at ETH Zürich, see figure 9.9. Here the robotic arms hang from a gantry system attached to the ceiling. If the assembly of the thin-tile vault were to take place within a factory or a construction site with part of the building's structure already overhanging this work space, this would be an alternative. However, when prefabrication is used, dynamics during transport need to be taken into account as well. The dynamics of the masonry structure are outside the scope of this research.

10

Model input

10.1. Robot

Current technology in robotics and especially catered for masonry, shows the use of a robotic arm as most suitable. The robotic simulation program has a wide variety of these available in their library. Similar to choosing the right epoxy, this can seem overwhelming for picking the right robotic arm. However, unlike epoxy, robotic arms can be categorized based on only five specifications: reach, payload, weight, speed and repeatability (combined with accuracy). One other factor could be the ability to attach end effectors. However, since a robotic arm is basically useless without its end effectors or tools, they should be able to attach a wide variety of them.

First of all the speed of robotic arms. These are not specified by the simulation program. Table 10.1 shows the speed of a couple of robotic arms. These six-axis robots can rotate around their base left right (JT1), with their shoulder up-down, with their elbow up-down again, with their forearm around their axis, with their wrist up-down and with their end around their axis again (JT6). Of course, the speeds may differ from these numbers based on the weight attached to the arm and the Cartesian movement of the tool path versus the motions of the robotic arm. How these speeds and the mentioned factors result in a speed of the Cartesian movement, is unknown by the author. However, the differences seem very little, with a maximum of 70°/sec (16% lower than 430°/sec).

The second specification is the repeatability (and accuracy). Repeatability and accuracy are two terms that can give a lot of confusion. Accuracy is the ability to hit on target, whatever the direction of the miss. Repeatability is the ability to hit the same spot again, whatever the distance to the target. It is common to find the repeatability of a robotic arm, but less so its accuracy. However, the repeatability is not a determining factor in choosing the robotic arm, since the margin of error is smaller than half a millimetre, sometimes even smaller than a tenth of a millimetre. For general building applications, this is precise enough.

The payload of the arm is the third specification. As can be seen in figure 10.1a, the payload is quickly over ten kilograms, with one exception at 3. The weight of one brick is barely one kilo (unless very large bricks are used, like 250×250×20 mm). However, this payload includes the end effector. It is unspecified how much an end effector may weigh, but a couple of kilos is not out of the question. Therefore, a payload of 3 kilos may just be too few, while a payload of 10 or higher seems sufficient. Most robotic arms fulfil this with the next specification in mind.

The reach of the robot may be the most important factor. As already stated in section 9.4, the reach of the robot influences the maximum dimensions of the vault. And with a movable robotic arm, how many calibrated positions are required. It seems the reach of a robotic arm, without any additional machines, is up to 4,5 meters.

Table 10.1: Maximum speed of joint axes: 1) Motoman MH50 II-20; 2) Fanuc M-710iC-20L; 3) Fanuc M-710iC-12L; 4) Kawasaki RS015X (Motoman, 2019; Fanuc, 2019, 2017; Kawasaki, 2020)

Robotic arm	1	2	3*	4*
JT1 (°/sec)	180	175	180	180
JT2 (°/sec)	178	175	180	180
JT3 (°/sec)	178	180	180	200
JT4 (°/sec)	400	350	400	410
JT5 (°/sec)	400	360	430	360
JT6 (°/sec)	600	600	630	610

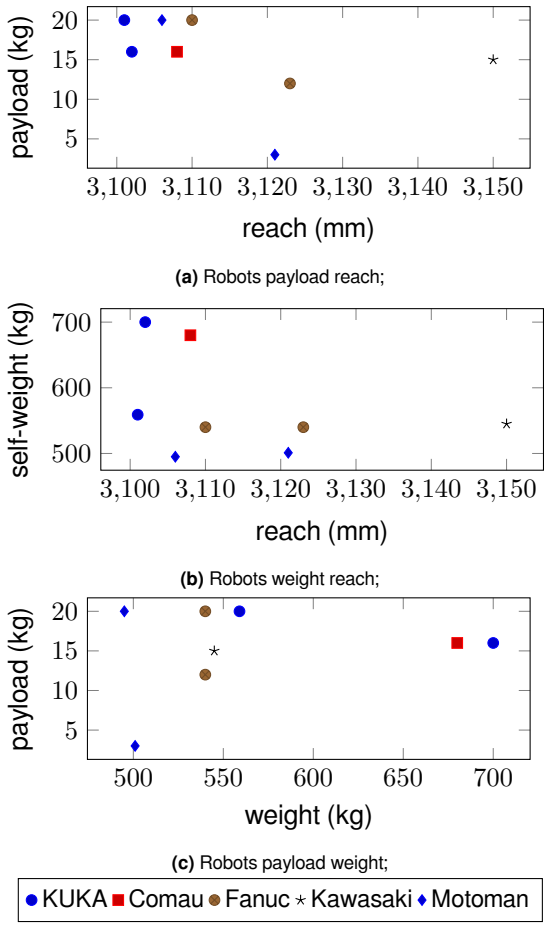


Figure 10.1: The relative positions of bricks around a $brick_{i;j}$, with the required edges in gray.

This limits the design space (immovable arm) and/or construction time greatly. For instance, in dwellings it isn't uncommon to find spans of 5,6 or 7,2 meters. Since no greater reaches can be used in the robotic simulation program, the vault will have to have limited dimensions.

The last specification is the weight of the robot. Since it is assumed here the construction happens on site, the structure supporting the robot cannot be assumed to be of industrial standards. Although a strong maximum is not present, it is more preferred if the robot were only 500 kilos than over a 1.000. With 500 kilo the robot is close to, or within the range of point loads used for the calculation of structures, as found in the Eurocode. Thus, a defined range is unspecified in this research, a low weight should be chosen for if other specifications are not lowered much.

In figure 10.1 a selection of robots found in the library of the simulation program are shown. These have been selected on two criteria: a six-axis robot and a reach of over 2 meters. The full range of robots can be found in the figures in appendix E. The Kawasaki and the Fanuc (both specified in table 10.1) have a high reach and a sufficient payload within this group. Thus, these two will be used for the analysis on the construction time.

10.2. Tool path

The tool path is in continuation of the chapters 4 and 9. Chapter 4 described how the geometry has been attained, while chapter 9 showed how this geometry is connected to the construction site.

The tool path requires two descriptions. The first is that of the order of visiting the stations. The second is to describe the implementation of the orientation of the end effector within the tool path.

The tool path is the route the robotic arm has to follow. This information is exported from the parametric model to the simulation program. The export can either be a set of points through which the robot has to move to, or a curve along which the robot moves. With the latter the total path is described in the parametric model. With the first, only the starting and destination point are described, with the movement inbetween left to the robot and the simulation program to figure out. The stations have been modelled in the parametric model as well. The description and conditions from section 9.1 for instance, have resulted in a list of points to be used within the parametric model. The same goes for the adhesive station. Although, to get to the information from section 9.2, a couple of other steps have to be taken as well. This has resulted in three groups of components each with the following tasks: to find the edges to apply the adhesive to; to describe the line of the adhesive applied to the bottom surfaces connecting two wythes together; to convert these edges and lines into positions and orientations (in the parametric model this results in planes).

10.2.1. Edges

The edges to apply the adhesive to are found from the vault model. First the surfaces next to each brick are identified. This is done with the data handling so far. To recap, the data tree containing all the bricks is made from two-level lists. The first level is the wythe of the brick, with 0 being the inner and 2 the outer wythe. The second level is the row it is away from the support. These rows are different than the courses of the brick. This allows for easy and general selection

criteria for which bricks are adjacent. Based on the orientation of the wythes, these criteria do differ. In the model three categories have been identified: under 0°, 0° and over 0°. A glimpse of why can already be seen in section 9.3 and figure 9.5. Each orientation has their own (part of) sides to apply the adhesive to. The bricks orientated at 0° have their short bottom side and bottom half of both long sides selected. Bricks under 0° have their bottom side and one of the longer sides entirely selected. Bricks over 0° have the same, but it is the other longer side.

With the help of the data structure these sides can be selected easily. In the case of 0°, two situations are defined. When an 'upper' brick is placed, only the previous brick in the row and the brick in the row beneath are selected. When a 'lower' brick is placed, the two bricks on either side in the previous row are selected as well. For under 0° and over 0° these bricks are the two closest in the previous row and the previous laid brick in its own row. In other words, let the placed brick be the j-th brick in row i, then for each category the following bricks are selected. See figure 10.2.

- For 0° at {i;j}: {i-1;j}, {i;j-1}, {i-1;j+1} (lower) or;
- For 0° at {i;j}: {i-1;j}, {i;j-1} (upper, see $bricki; j + 1$ in 10.1b);
- For all others at {i;j}: {i-1;j}, {i;j-1}, {i-1;j+1}.

This list differs slightly from the bricks mentioned above. This is due to two factors. The first is to remove duplicate selections. i-1;j andi;j-1 for others than 0° will select the same edge. Only one has to be selected. i;j-1 is preferred, because the first row of bricks (directly to the support) does not have a previous row to select from, while it does need this edge selected. The same goes for i;j-1 and i-1;j-1 for bricks at 0°, where i;j-1 is preferred. The second difference is due to the data structure. Although the long edges that need to be selected for non-0° bricks are opposites, the wythes with bricks over 0° will be laid in reverse. The data structure, as mentioned before, has twolevel lists. These lists contain all the bricks in one row from left to right (or from lowest x-value to highest x-value in the global coordinate system).

While this suits bricks for under and at 0°, the approach described in section 9.3 needs to be altered for over 0°. That's why wythes with bricks over 0° have their lists reversed. The lists has all the bricks in one row from right to left. Hence why the selection relative to i and j can stay the same. This procedure results in the correct edges positioned within the used data structure (although now the tree is at level 3: besides each wythe and each row, each brick has also been given its own list). These edges are only selected though. They're position and orientation in the model are still related to the vault design.

10.2.2. Interface and plane

The second group of components are needed to apply adhesive between the wythes. The bricks from the second and third wythe have the edges of the brick and its bottom applied with adhesive. As with the edges, the adhesive is applied on one line and not altered, like smearing, afterwards. This line has a thickness similar to the width of the muzzle of the adhesive gun. To apply the adhesive to the whole bottom of the surface, a space-filling curve is required. The space-filling curve here is lines parallel to the longer edges of the brick, with these lines connected at their ends. This shape is visualized in figure 10.3.

The third group of components relates the lines from the edges and the interfaces to the position and orientation of the bricks and end effector. As mentioned in section 9.2 the adhesive is applied around the brick. This means each of the edges found in the first group has to be transformed to the horizontal plane and in front of the adhesive

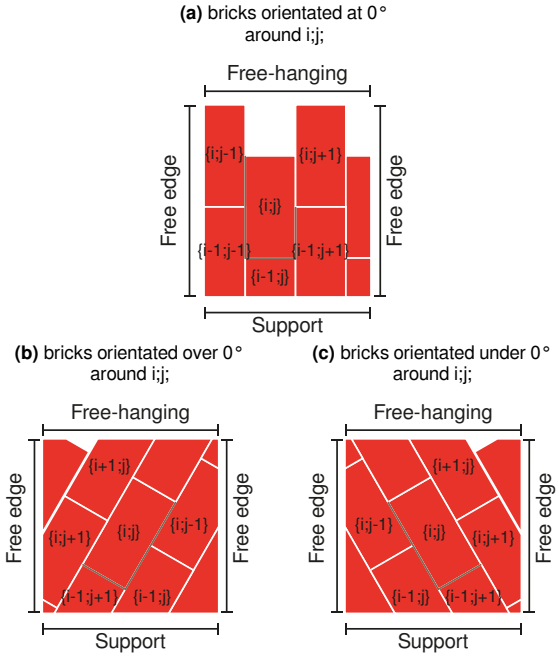


Figure 10.2: The relative positions of bricks around a $brick_{i;j}$, with the required edges in gray.

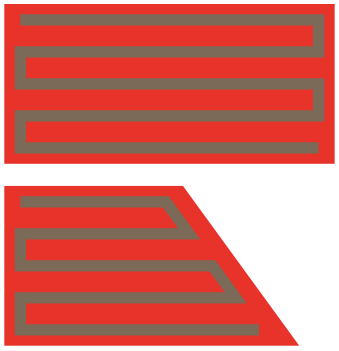


Figure 10.3: The layout of the adhesive on the bottom side of the bricks from the second and third wythe. Both a whole and cut brick are shown.

l plane to have the adhesive station in the normal direction of the bottom surface.

station. Meanwhile, the bricks in the vault design have their plane as described in chapter 9. These planes are transformed in parallel with the edges. As can be seen from figure 10.2 and as described in section 9.2, the adhesive is applied to a set of lines, somewhat in a continuous fashion. Since these lines are straight, the tool path created in the robotic simulation program will be as desired, if only the starting and destination points are given. That is, so long as the orientation at both is the same. Thus, with the edges transformed correctly passed the adhesive station; with the planes transforming along side the edges; and with the end effector having sufficient information from these planes, the third group results in a list of planes (again in the three-level data structure) through which the robotic arm has to move. The derivation of the planes for the interface are in a similar fashion. However, here the bricks aren't flat in the horizontal plane, but in the vertica

10.2.3. Passed stations

At each station the actions as described in chapter 9 lead to the tool path at that station. The previous subsection also showed how the tool path at the adhesive station is constructed. This also included the planes through which the robot has to move. At the pallet and vault stations these planes are derived much simpler. As described in their respective sections, the position and orientation of the end effector coincide with the centroid projected on the top surface of the brick. All these planes representing the actions ensure that the robotic arm moves correctly through each station.

The movement between the stations is based on the start and end planes of each station's (first and last) actions. The plane from the last action of one station is the starting point for the movement to the destination point, which is the plane from the first action of the next station. The order in which the robot visits the stations is as follows.

1. Home position of robotic arm;
2. Pallet station;
3. Adhesive station;
4. Vault station;
5. Repeat step 2-4 until the structure has been constructed;
6. Home position of robotic arm.

The export from the parametric model to the simulation program is done with a script provided by the simulation program and altered for further usability. This program takes a CSV-file of six values: XYZWPR. These are standard robot and airplane coordinates. Here XYZ are the Cartesian coordinates and WPR are the Euler angles (RoboDK, 2018). WPR stands for Yaw, Pitch and Roll. Roll is the rotation around the effector's longitudinal axis. Pitch is the rotation around the transverse axis. Yaw is the rotation around the vertical axis of the end effector.

With the planes per station per brick now in the correct order, a full list of positions and orientations is ready for the robotic arm. However, these planes need to be related to the base reference frame for the simulation program. That's where XYZWPR is used. Dr. S. Asut from the TU Delft has provided a component for the parametric model. The output of this component gives the yaw, pitch and roll. The planes' origin give the x-, y-, and z-coordinates. Combining these in the correct order, separated by commas, will give a csv from the parametric model. Additionally, the CAD program can stream the contents of a csv to a file. Thus, the exported file is automatically updated. The parametric model also exports a seventh CSV-value. Besides XYZWPR, the wait time at each plane is also exported. This is zero for almost all planes, except for the third action at the vault station. The script file in the simulation program has been altered to extract this seventh value appropriately.

.

Part V

Results & Conclusion

11

Results

11.1. Parameters of the model

The parameters used in the model can be split into two categories: design and geometric related parameters and structural related parameters. The design and geometric related parameters are shown in figure 11.1. All parameters and related information are shown in table 11.1. Although the global shape of the vault has the ability to be double curved, this has not been worked out for later stages in the parametric model. The orientation of the brick pattern is similar. Although most angles will work for the general shape, only for -45°, 0° and 45° is the certainty this will work in later stages. The parameters are based on the following assumptions.

- The vault has a single curvature over the span and a curvature of 0 along the length;
- The robots used are the Fanuc M-710iC-12L and the Kawasaki RS015X (see section 10.1;
- The thin bricks are cut from normal clay bricks, similar to brick slips;
- The vault can be reduced to an arch of thickness per stretching meter;
- The vault is only analysed from one side until the apex, where the construction time of the other side is assumed similar.

11.2. Configurations

The construction time of various parametric inputs have been looked into. To reduce the number of computations, a base parametric input has been set-up. With each additional computation, only one parameter will be altered to investigate its influence. The base parametric input can be found in table 11.2. The alteration per computation is shown in table 11.3. The used robot is changed in the first altered computation to show the influence of the robotic movements and the robotic dimensions. With a changed design, the second altered computation puts the influence of the shape and therefore the position of the bricks in a spotlight. The third computation switches the angles of the wythes to see if this has any influence. The base computation uses a relatively low pot life to give greater emphasis to other aspects of the construction, since it is expected that the hardening time will take a large portion of the construction. Therefore, the pot life is increased in the fourth altered computation to a pot life more likely to be with most suitable epoxies. Narrowing the length of the course of the vault in the fifth computation influences both

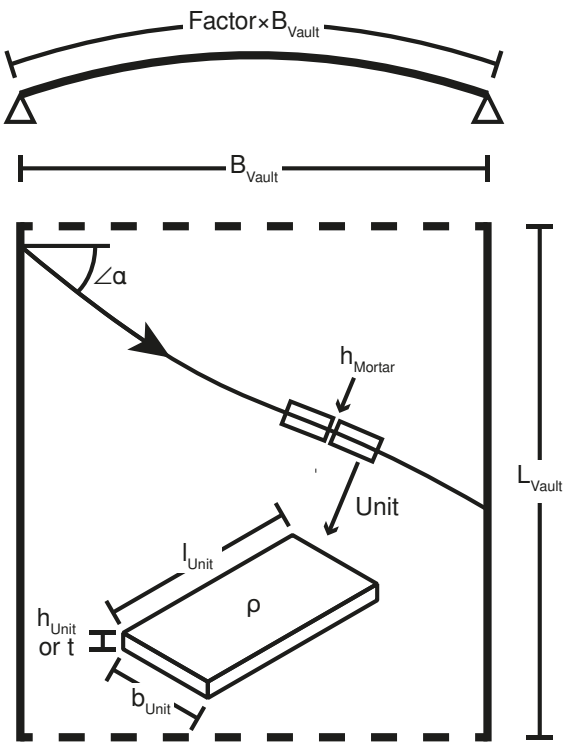


Figure 11.1: The design or geometric related parameters in the model.

Table 11.1: Parameters of the model and related information.

Parameter	Description	Range	Unit
	Design or geometric parameters		
B_{Vault}	The span of the vault (transverse axis)	≤ 2,6	m
L_{Vault}	The course length of the vault (longitudinal axis)	1,8	m
$Factor$	The overlength of the catenary relative to the span	>1,0	-
$\angle\alpha$	The angle of the course with the transverse axis	-45 - 45	°
h_{Mortar}	The thickness of the adhesive beds (all sides)	0,001 - 0,020	m
l_{Unit}	The length of the brick units (edge of bed and stretcher faces)	0,100 - 0,600	m
b_{Unit}	The width of the brick units (edge of bed and header faces)	0,040 - 0,240	m
h_{Unit}	The thickness of the brick units (edge of stretcher and header faces)	< $l_{Unit} \vee b_{Unit}$	m
t	See h_{Unit}*	*	m
	Structural parameters		
ρ	Density of the brick unit (g = 9, 81m/s^2, γ is mass density)	1400 - 1800	kg/m^3
f_b	The compressive strength of the brick units	5-30	MPa
$f_{m,init}$	The initial compressive strength of the adhesive upon release of the unit	> compressive stresses due to the weight of one cantilevering brick	MPa
$f_{m,final}$	The final compressive strength of the adhesive	1,5-7,5	MPa
f_{vk0}	The initial shear strength of the masonry	0,30	MPa
$f_{m,t,init}$	The initial tensile strength of the adhesive upon release of the unit	> tensile stresses due to the weight of one cantilevering brick	MPa
$f_{m,t,final}$	The final tensile strength of the adhesive	0,5-2,5	MPa
$tpotlife$	The time for the adhesive to reach $f_{m,init}$ and $f_{m,t,init}$	0,5 - 100	minutes
Δx	The step between analysed cross-sections in the domain/shape of x within the parametric model	$\Delta x < \Delta rows$	m

the design, similar to Comp$_2$, but also reduces the time a row of bricks can develop strength before moving on to the next one. The sixth altered computation changes the work space or work environment with which also the robotic movements change. The last computation changes the preferred construction sequence.

11.3. Computation 0: the base parameter input

The first computation is the base parameter input. In table 11.2 the used values for this computation can be found.

The structure has been analysed on its phased stresses, strengths and consequently unity check through time. It is possible to lay 31 rows of bricks before the analysis shows the strengths have been exceeded. In figure 11.3 a couple of these analyses are shown. The figures show the stresses and strengths throughout the whole structure after placement of the row of bricks 3, 17, 31 and the failure at 32. See chapter 7 for further explanation of the graph.

The construction sequence shows a pattern in line with the a.s.a.p. preference. Save for a few exceptions, results the placement of a row in the possibility of placing the next row in the wythe following. The order $0 \rightarrow 1 \rightarrow 2 \rightarrow 0$ is the standard. Deviation is likely due to the longer $\{u\}$ of the second wythe ($\alpha = 0°$). The sequence is shown in figure 11.2. This sequence results in 303 bricks being placed.

Table 11.2: Base parameter input: $Comp_0$

Parameter	Value	Unit
Robot	Kawasaki	-
Config.	1	-
B_{Vault}	3,6	m
L_{Vault}	1,2	m
$Factor$	1,1	-
$\angle\alpha_0$	-45	°
$\angle\alpha_1$	0	°
$\angle\alpha_2$	45	°
h_{Mortar}	0,005	m
l_{Unit}	0,210	m
b_{Unit}	0,100	m
h_{Unit}	0,020	m
ρ	1750	kg/m^3
f_b	20	MPa
$f_{m,init}$	0,1	MPa
$f_{m,final}$	>f_b	MPa
f_{vk0}	0,30	MPa
$f_{m,t,init}$	0,1	MPa
$f_{t,final}$	2,0	MPa
$t_{potlife}$	0,5	minutes
$t_{curetime}$	40	minutes
Δx	0,04	m
w_{rule}	a.s.a.p.	-

Table 11.3: Altered parameter for computations

#	Parameter	Value	Unit
$Comp_1$	Robot	Fanuc	-
$Comp_2$	Factor	1,25	-
$Comp_3$	$\angle\alpha_0$	0	°
	$\angle\alpha_1$	-45	°
$Comp_4$	$t_{potlife}$	5	minutes
$Comp_5$	L_{Vault}	0,8	m
$Comp_6$	Config.	2	-
$Comp_7$	w_{rule}	a.l.a.p.	-

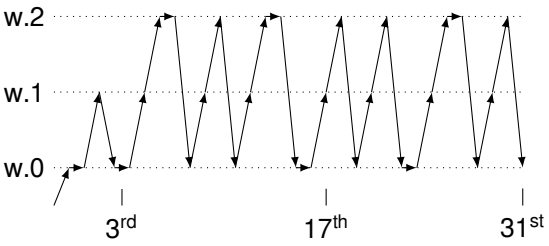
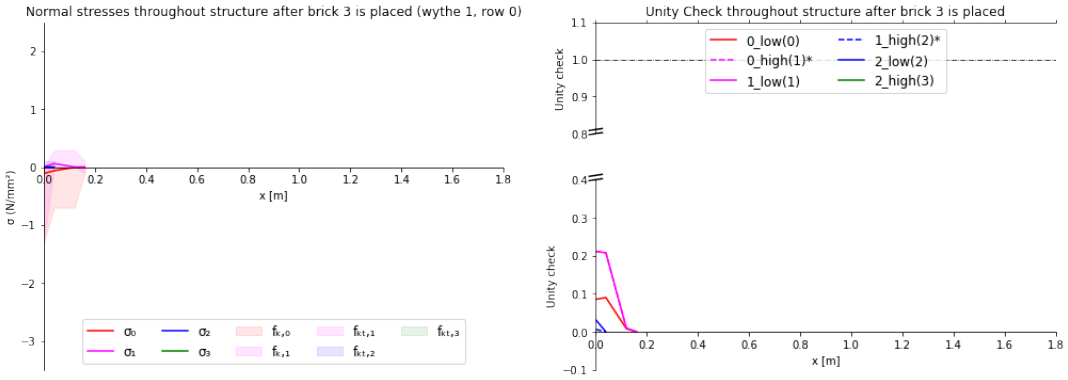
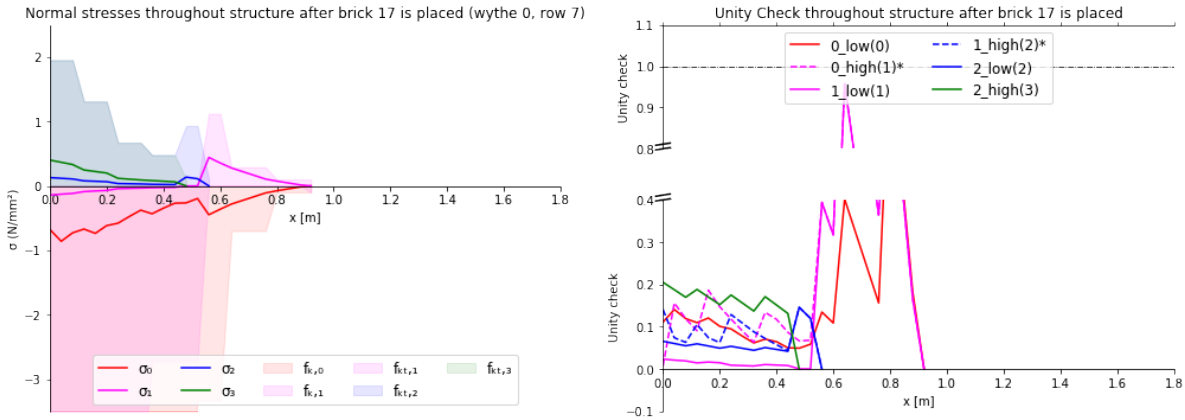


Figure 11.2: The construction sequence of the first computation.

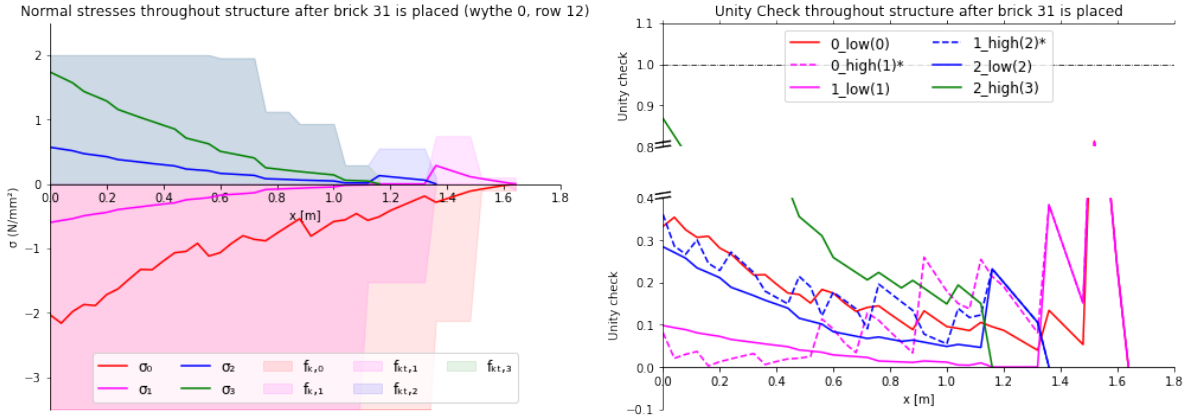


(a) Left: Stresses (lines) and strengths (area) of the structure after the third brick has been placed; right: the unity check through the whole structure, shown at the four interfaces from figure 6.5;

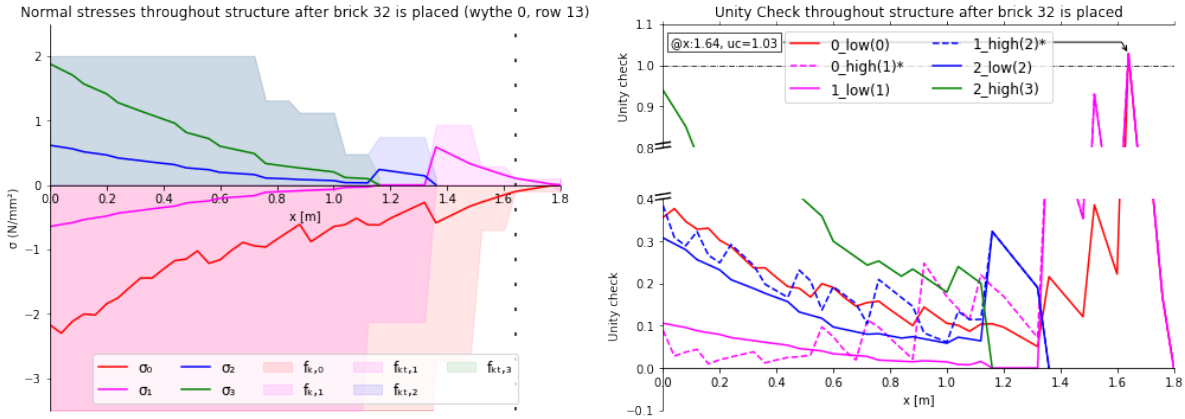
Figure 11.3: The results from the structural analysis of the first computation.



(b) The stresses, strengths and unity checks after the seventeenth brick;



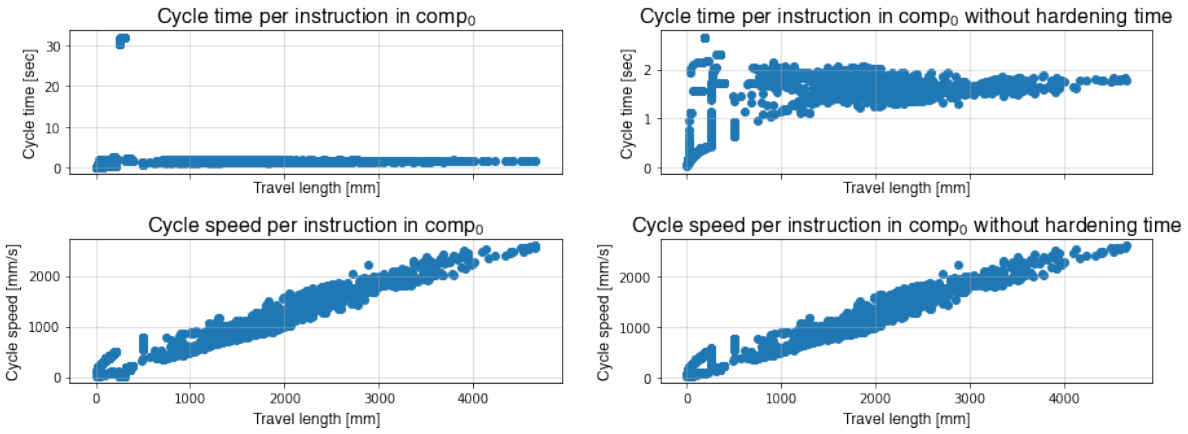
(c) The stresses, strengths and unity checks after the thirty-first brick;



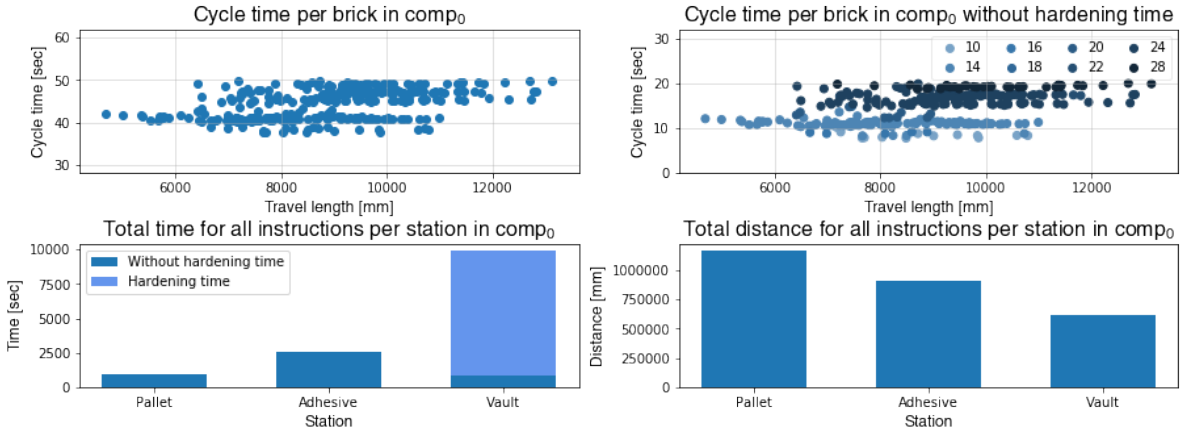
(d) The stresses, strengths and unity checks after the thirty-second brick; the dashed line in the left graph shows the x-position of failure, labelled in the right graph as well;

Figure 11.3: The results from the structural analysis of the first computation. The x-axis is the cantilevering length of the structure. On the y-axis are respectively the stresses (in MPa) and the unity check. In the left graph four lines are shown, which are the stresses (σ) at the interfaces (see figure 6.5). Additionally, five strengths (f) are shown as an area. For the second interface two strengths are shown, since both compression and tension occur at this interface. These strengths are not drawn simultaneously, but only when the stresses are compressive or tensile as well. In the right graph six unity checks are shown. Besides the four from the stresses on the left (solid lines), two additional checks are shown as well (dashed). These occur at the interfaces between the wythes, and are the approach from the wythe below, as the other ones are with the approach from the wythe on top. The colours are consistent for the interface: the bottom of the cross-section is red, the interface between the first and second wythe is purple, the interface between the second and third is blue and the top of the cross-section is green. See chapter 7 for further explanation of the graph.

Figure 11.5 shows the simulation of the construction with a robot. The construction time noted is 1 hour and 8 minutes. However, this is achieved with a wait time of 800 milliseconds. This should be half a minute (30.000 seconds). With 303 bricks, this is an increase of 2 hours and 28 minutes (8.847.600 milliseconds), resulting in a total construction time of 3 hours and 36 minutes. The construction time per instruction, per brick and per station are shown in figure 11.4. The length to travel has little to do with the time for each instruction. Excluding the hardening time, an upper and lower bound can be found for the cycle time. The little variance in cycle times with high travel length is likely due to one joint rotation taking up to 2 seconds and all other joints having a less significant or smaller rotation. The higher variance in cycle time for low travel lengths is likely due to the little rotation all joints have to take (low cycle time) and to one or more joints requiring a high rotation due to the complicated movement in close space. Another aspect is the increased cycle time due to the increase in instructions. As chapter 9 describes, the vault and pallet station only have three instructions, always. The adhesive station is dependent on the number of edges, but at least 4. Therefore, the reason for the difference in time per station also becomes clear. Minor differences in required movements won't result in major differences (as is the case between the vault and pallet station), but the increase in instructions will. However, this increase is little compared to the effect of the hardening time.



(a) The time the robot needs to perform this instruction (top) in the first computation, and its accompanying average speed doing that (bottom), with the hardening time included (left) and excluded (right, note the change in values of the vertical axis);



(b) The time the robot needs to place each brick (top) with (left) and without (right, note change in vertical axis and colouring is based on number of instructions) the hardening time;

Figure 11.4: The construction time results of the first computation. Where possible have axes stayed the same. The instructions at the vault take up a large amount of time. The cloud of points with a much higher cycle time in the top left graph of figure 11.4a are the placement of the bricks including the hardening time. The time per instruction in general has an upper and lower bound, with the increase in speed compensating for the increase in travel length. This continues with the time per brick as well, where the increase in cycle time can be attributed to the increase in instructions. The total time per station in this computation shows that the vault is the main component, but the adhesive surpasses it when the hardening time is excluded.

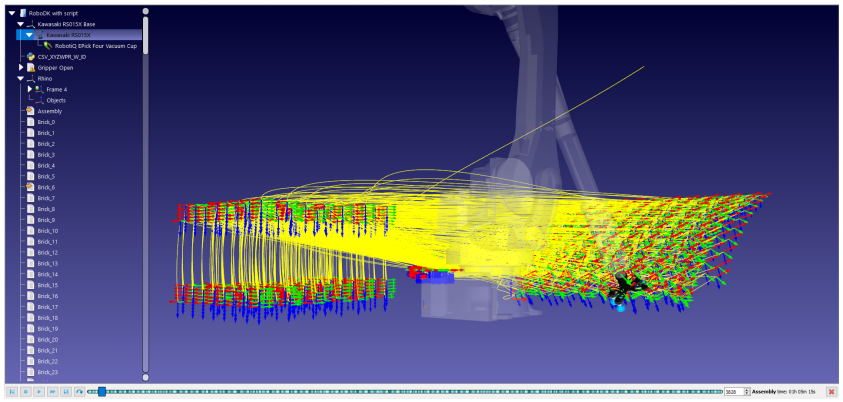
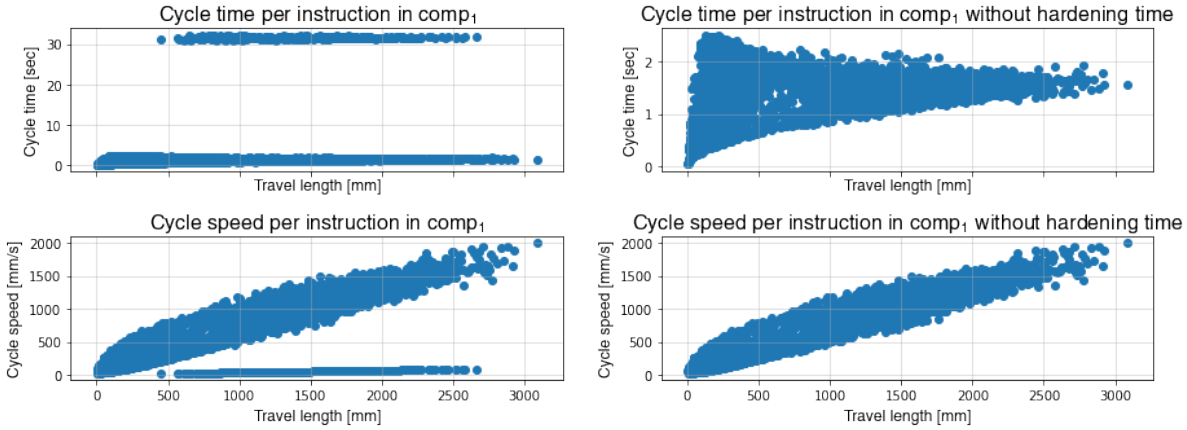


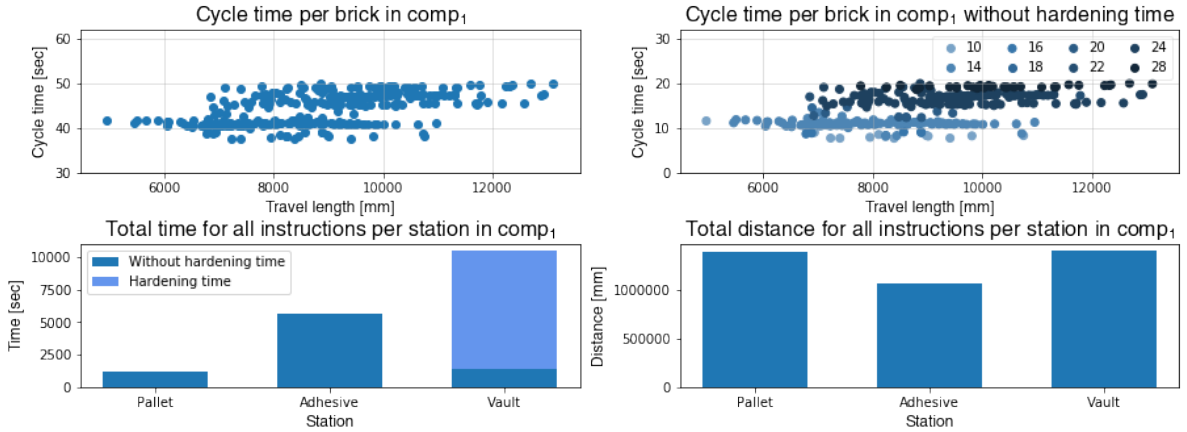
Figure 11.5: The configuration in the simulation.

11.4. Computation 1: Fanuc robotic arm

The structural analysis is the same as in section 11.3. 303 bricks can be placed. The robotic arm has changed, which has resulted in less longer travel lengths compared to the first computation. The total construction time is 3 hours and 43 minutes, which is an increase of 7 minutes (+3%) compared to the first computation.



(a) The time the robot needs to perform this instruction (top) in the second computation, and its accompanying average speed doing that (bottom), with the hardening time included (left) and excluded (right, note the change in values of the vertical axis);

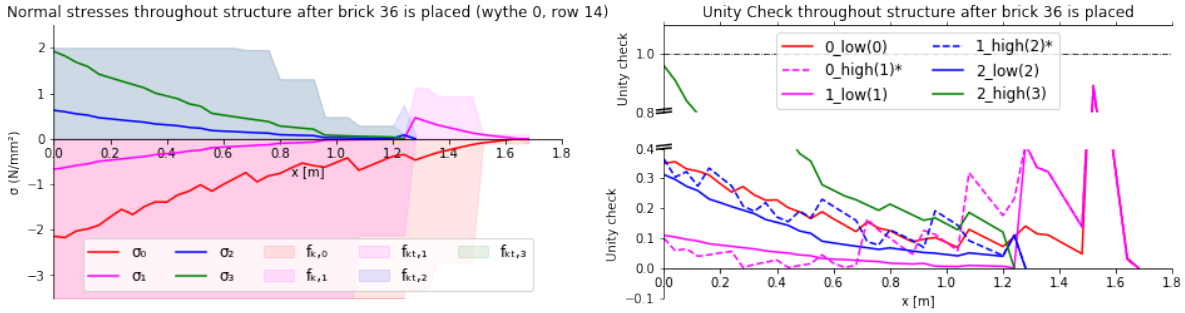


(b) The time the robot needs to place each brick (top) with (left) and without (right, note change in vertical axis and colouring is based on number of instructions) the hardening time;

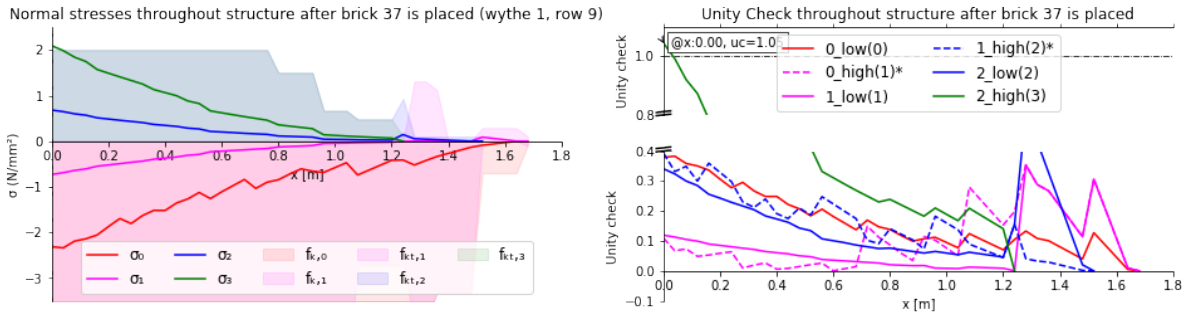
Figure 11.6: The construction time results of the second computation. Where possible have axes stayed the same. Points of interest visible here, but similar to figure 11.4, are not repeated. Using a different robotic arm has resulted in less extreme travel lengths. The maximum travel length of any instruction has reduced with 33%.

11.5. Computation 2: Factor = 1.25

The structural analysis of a changed design of the vault has resulted in more bricks possible to place: 36 rows compared to 31. However, this does not result in an increase of the cantilever length, since the curve length of the vault has increased as well. In other words, the same length of the bricks in a row, result in less length in the global Cartesian coordinates. 351 bricks are constructed. The total construction time is 4 hours and 19 minutes. This is an increase of 43 minutes (+20%).

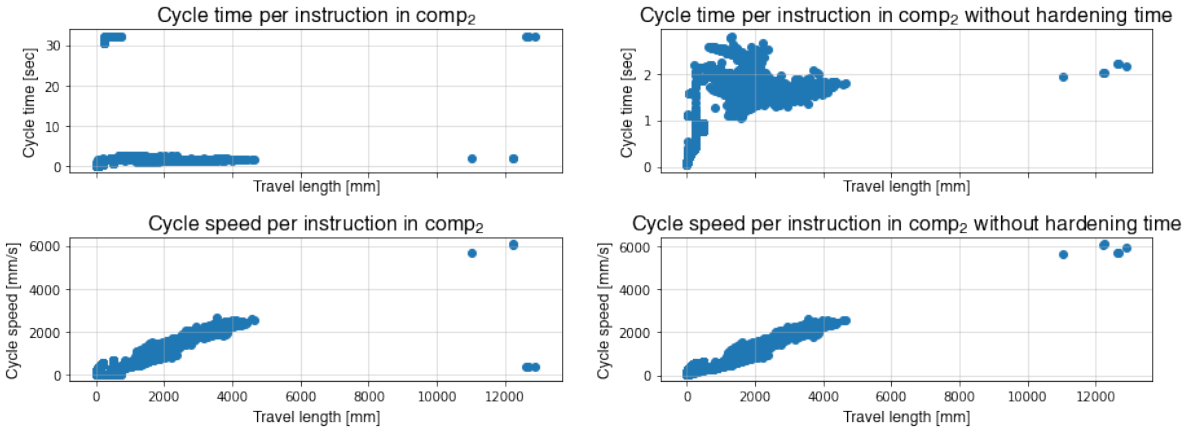


(a) The stresses, strengths and unity checks after the thrity-sixth brick;



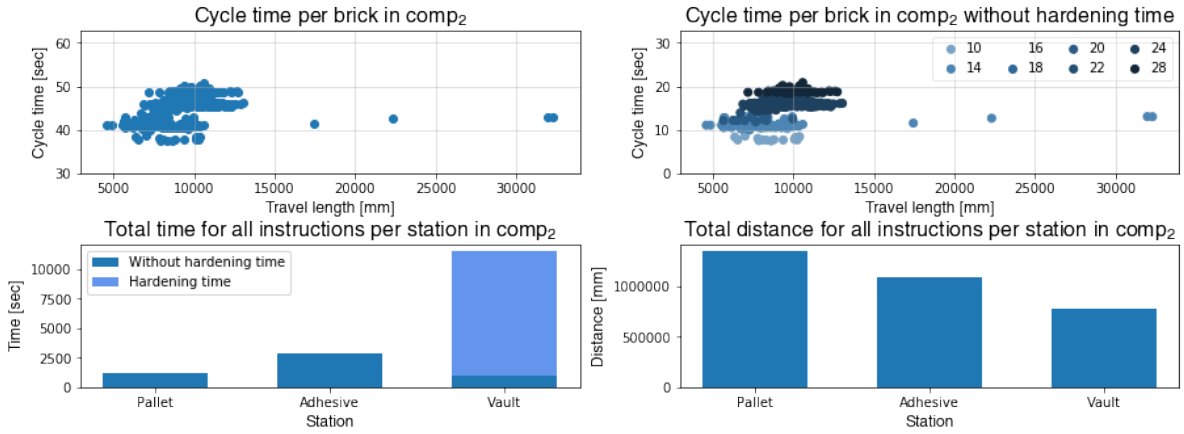
(b) The stresses, strengths and unity checks after the thirty-seventh brick; the dashed line of failure in the left graph is invisible since x=0, labelled in the right graph as well;

Figure 11.7: The results from the structural analysis of the third computation. The x-axis is the cantilevering length of the structure. On the y-axis are respectively the stresses (in MPa) and the unity check. In the left graph four lines are shown, which are the stresses (σ) at the interfaces (see figure 6.5). Additionally, five strengths (f) are shown as an area. For the second interface two strengths are shown. In the right graph six unity checks are shown. Besides the four from the stresses on the left (solid lines), two additional checks are shown as well (dashed).



(a) The time the robot needs to perform this instruction (top) in the third computation, and its accompanying average speed doing that (bottom), with the hardening time included (left) and excluded (right, note the change in values of the vertical axis);

Figure 11.8: The construction time results of the third computation



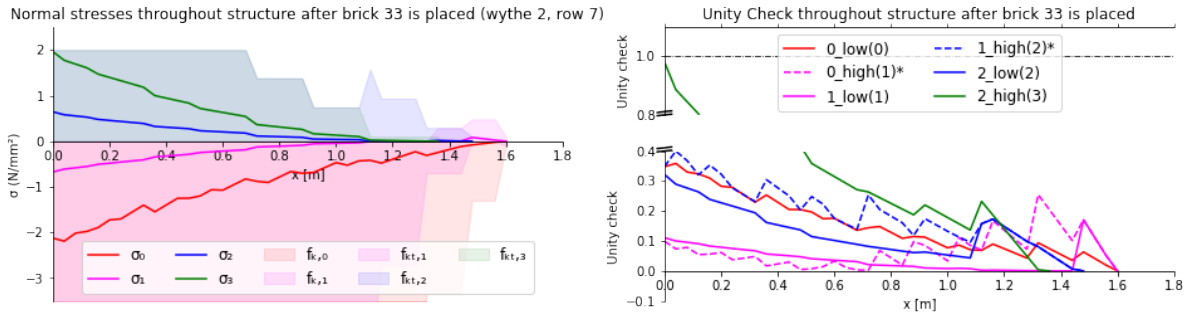
(b) The time the robot needs to place each brick (top) with (left) and without (right, note change in vertical axis) the hardening time;

Figure 11.8: The construction time results of the third computation. Where possible have axes stayed the same. Points of interest visible here, but similar to previous figures, are not repeated. The different shape of the vault has resulted in some extreme manoeuvres for execution. Manual alteration for these movements would be necessary, see 12.

11.6. Computation 3: Wythe orientations swapped

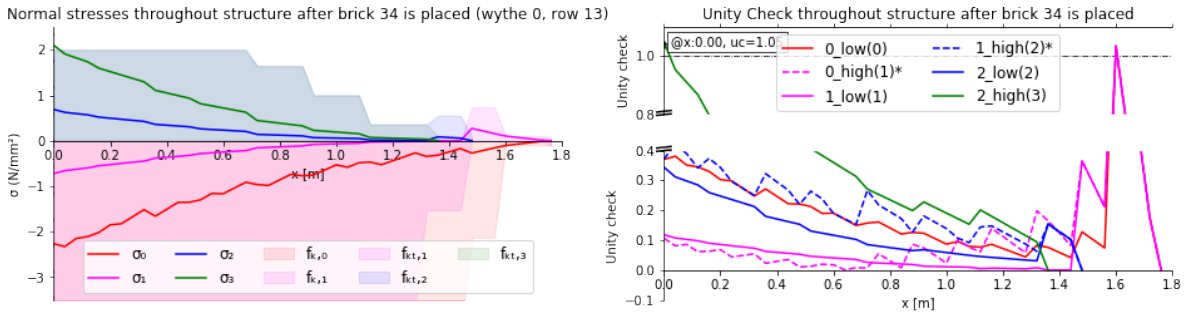
First wythe and second wythe swapped: $\alpha_0 = 0$ **&** $\alpha_1 = -45$**.** The extend of the first row of the first wythe is important (how much is cut off from the bricks to make the cut bricks of the first row?). The further the first row(s) are (the larger the bricks in the first row), the lower the number of bricks can be placed. However, even in the most optimal form, only six bricks can be placed, without changing other parameters as stated in table 11.3.

Second wythe and third wythe swapped: $\alpha_1 = 45$ **&** $\alpha_2 = 0$**.** The other possibility (the last one is trivial with same, but opposite α) results in a similar structural analysis as the first computations. Now, 33 rows of bricks can be placed, where the second wythe, with its smaller length increments, results in more rows to be placed. 321 bricks can be constructed. The total construction time is 4 hours and 0 minutes, an increase of 24 minutes (+11%).



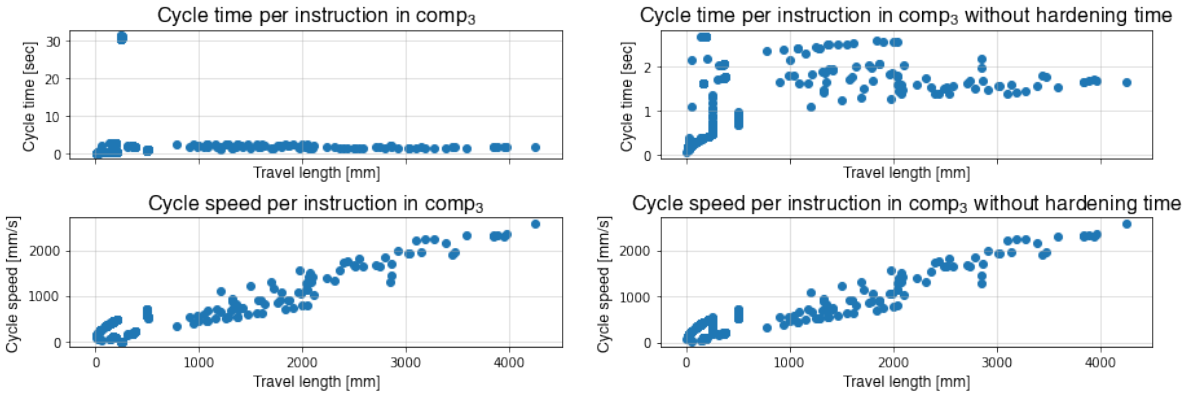
(a) The stresses, strengths and unity checks after the thrity-third brick;

Figure 11.9: The results from the structural analysis of the fourth computation

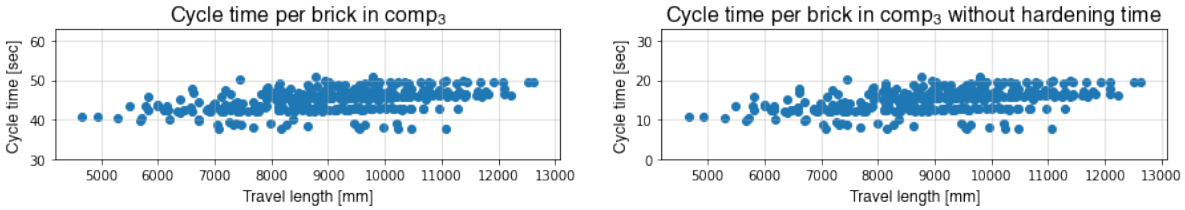


(b) The stresses, strengths and unity checks after the thirty-fourth brick; the dashed line in the left graph shows the x-position of failure, labelled in right graph as well;

Figure 11.9: The results from the structural analysis of the fourth computation. The x-axis is the cantilevering length of the structure. On the y-axis are respectively the stresses (in MPa) and the unity check. In the left graph four lines are shown, which are the stresses (σ) at the interfaces (see figure 6.5). Additionally, five strengths (f) are shown as an area. For the second interface two strengths are shown. In the right graph six unity checks are shown. Besides the four from the stresses on the left (solid lines), two additional checks are shown as well (dashed).



(a) The time the robot needs to perform this instruction (top) in the fourth computation, and its accompanying average speed doing that (bottom), with the hardening time included (left) and excluded (right, note the change in values of the vertical axis);

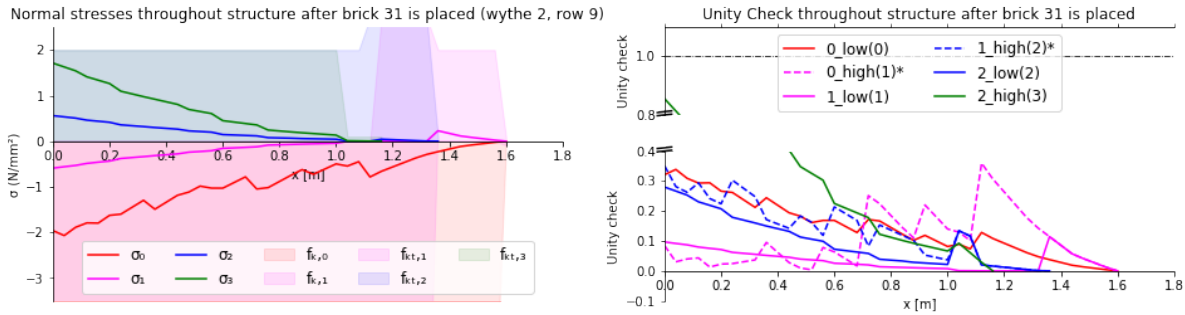


(b) The time the robot needs to place each brick (top) with (left) and without (right, note change in vertical axis) the hardening time;

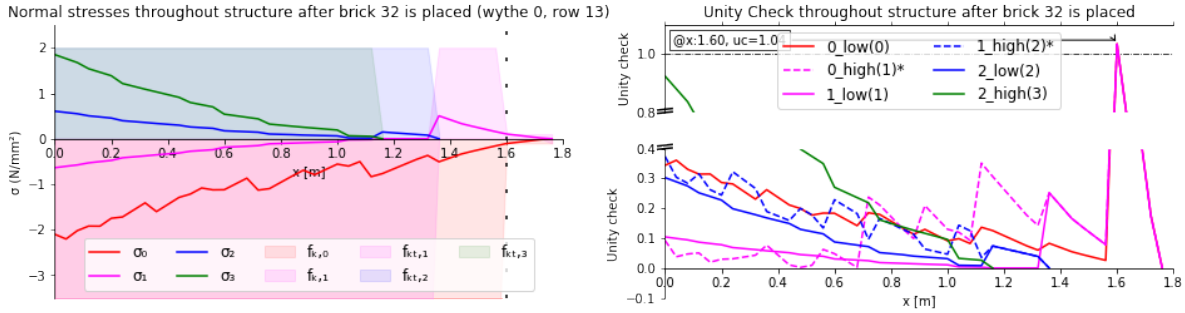
Figure 11.10: The construction time results of the fourth computation. Where possible have axes stayed the same. Points of interest visible here, but similar to previous figures, are not repeated. To reduce the calculation time of each computation, computations have a reduced number of instructions (10% compared to the first three computations). This also results in a lack of data to continue the analysis between the three stations.

11.7. Computation 4: Potlife = 5 minutes

The structural analysis with an increased pot life has not resulted in any significant change compared to the first computation. This is reflected in the number of bricks that can be constructed: 303. The total construction time, however, has increased to 26 hours and 30 minutes, a change of 22 hours and 54 minutes (+736%).

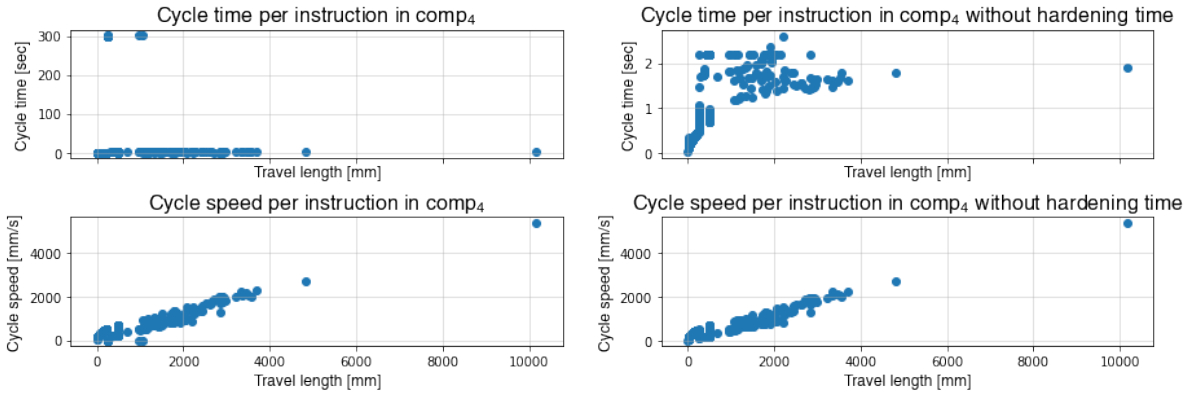


(a) The stresses, strengths and unity checks after the thirty-first brick;

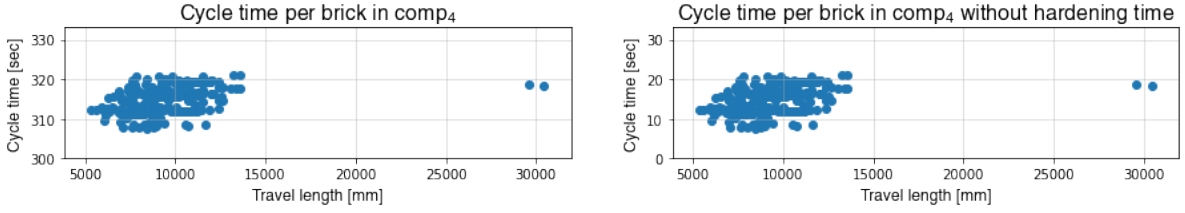


(b) The stresses, strengths and unity checks after the thirty-second brick; the dashed line in the left graph shows the x-position of failure, labelled in right graph as well;

Figure 11.11: The results from the structural analysis of the fifth computation. The x-axis is the cantilevering length of the structure. On the y-axis are respectively the stresses (in MPa) and the unity check. In the left graph four lines are shown, which are the stresses (σ) at the interfaces (see figure 6.5). Additionally, five strengths (f) are shown as an area. For the second interface two strengths are shown. In the right graph six unity checks are shown. Besides the four from the stresses on the left (solid lines), two additional checks are shown as well (dashed).



(a) The time the robot needs to perform this instruction (top) in the fifth computation, and its accompanying average speed doing that (bottom), with the hardening time included (left) and excluded (right, note the change in values of the vertical axis);

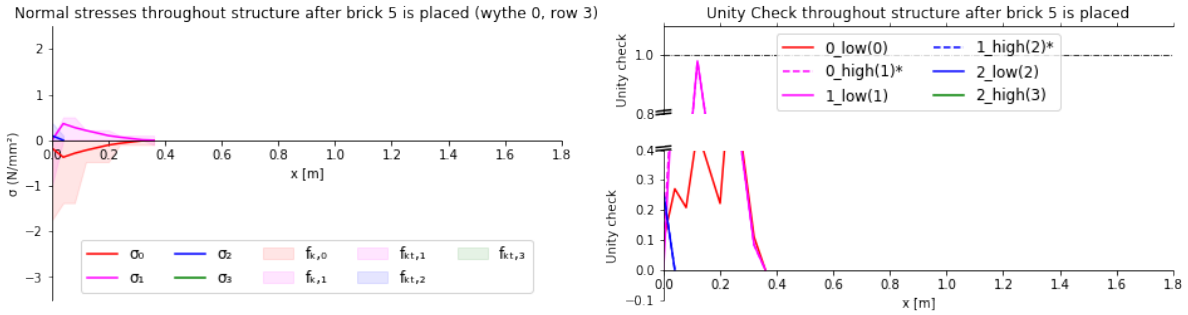


(b) The time the robot needs to place each brick (top) with (left) and without (right, note change in vertical axis) the hardening time;

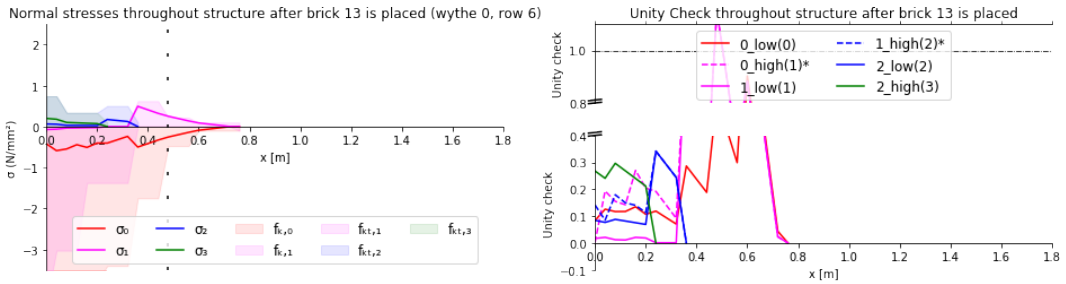
Figure 11.12: The construction time results of the fifth computation. Where possible have axes stayed the same. Points of interest visible here, but similar to previous figures, are not repeated. The graph to show the time per station is omitted, as mentioned in the previous section. The cycle times of the instructions including the hardening have changed, but that is to be expected.

11.8. Computation 5: L=0.8m

The reduced length of the course of the vault has resulted in less time for the strength to develop before a next row/load is added. Only 12 rows of bricks can be placed, resulting in 78 bricks in total. The construction time has also reduced because of this: 59 minutes. This is a reduction of 2 hours 37 minutes (-73%).

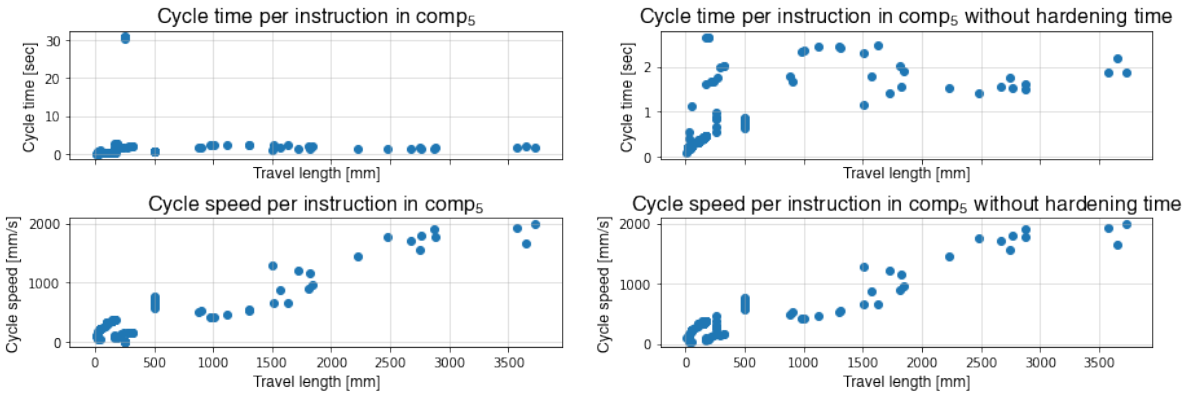


(a) The stresses, strengths and unity checks after the fifth brick;



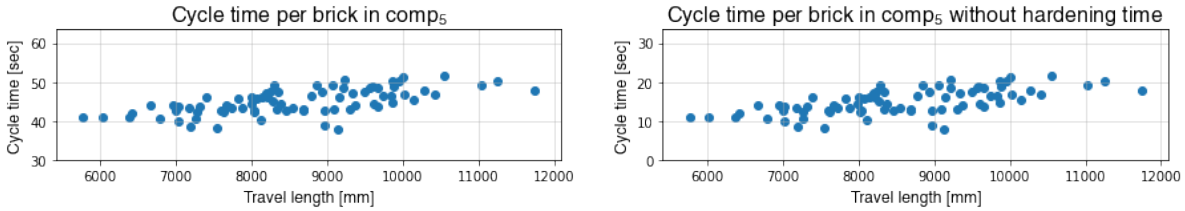
(b) The stresses, strengths and unity checks after the thirteenth brick; the dashed line in the left graph shows the x-position of failure, labelled in right graph as well; the stresses in the final construction are trivial;

Figure 11.13: The results from the structural analysis of the sixth computation. The x-axis is the cantilevering length of the structure. On the y-axis are respectively the stresses (in MPa) and the unity check. In the left graph four lines are shown, which are the stresses (σ) at the interfaces (see figure 6.5). Additionally, five strengths (f) are shown as an area. For the second interface two strengths are shown. In the right graph six unity checks are shown. Besides the four from the stresses on the left (solid lines), two additional checks are shown as well (dashed).



(a) The time the robot needs to perform this instruction (top) in the sixth computation, and its accompanying average speed doing that (bottom), with the hardening time included (left) and excluded (right, note the change in values of the vertical axis);

Figure 11.14: The construction time results of the sixth computation



(b) The time the robot needs to place each brick (top) with (left) and without (right, note change in vertical axis) the hardening time;

Figure 11.14: The construction time results of the sixth computation. Where possible have axes stayed the same. Points of interest visible here, but similar to previous figures, are not repeated.

11.9. Computation 6: Configuration 2

The structural analysis is the same as in section 11.3. 303 bricks can be placed. The results from this configuration differ from all other computations. The upper and lower bound with the cycle time are less clear, while the cycle speeds are organized in two widening lines, like tongs. The total construction time is 4 hours and 2 minutes, which is an increase of 26 minutes (+12%). Figure 11.15 shows the configuration in the simulation.

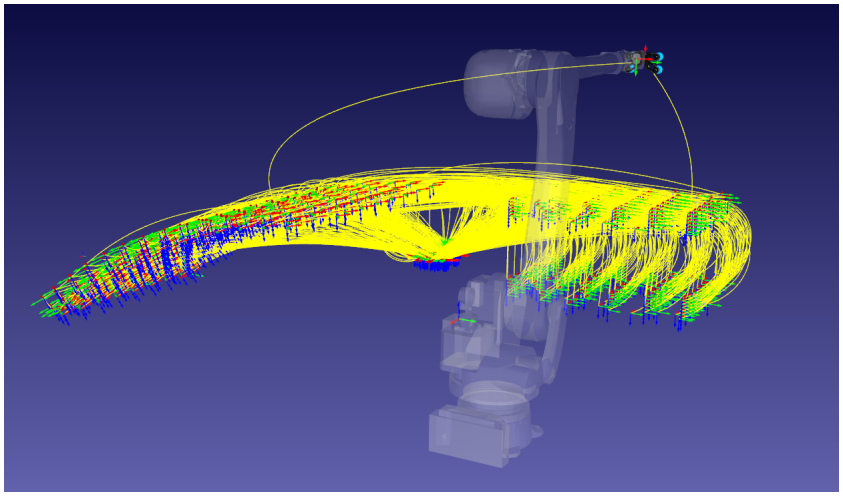
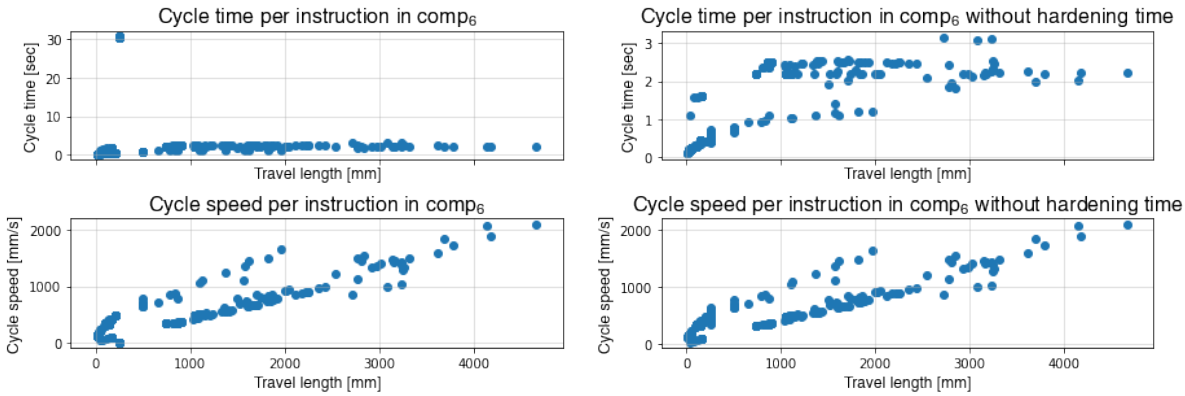
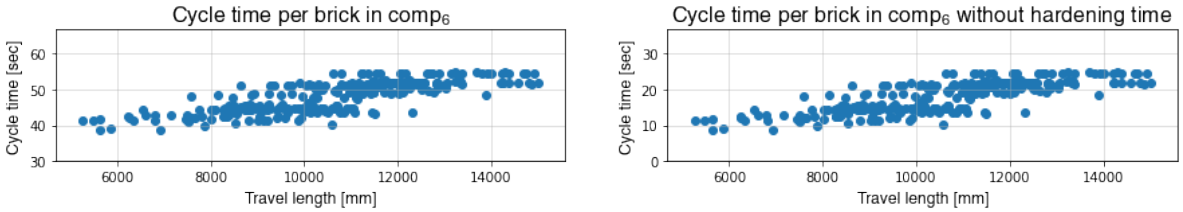


Figure 11.15: The configuration in the simulation.



(a) The time the robot needs to perform this instruction (top) in the seventh computation, and its accompanying average speed doing that (bottom), with the hardening time included (left) and excluded (right, note the change in values of the vertical axis);

Figure 11.16: The construction time results of the seventh computation



(b) The time the robot needs to place each brick (top) with (left) and without (right, note change in vertical axis) the hardening time;

Figure 11.16: The construction time results of the seventh computation. Where possible have axes stayed the same. Points of interest visible here, but similar to previous figures, are not repeated.

11.10. Computation 7: preferred sequence: a.l.a.p.

The structural analysis results in 31 rows to be constructed. This is similar to the first computations and results in the same number of bricks: 303. However, the resulting stresses, especially at the support, are much lower than the ones found in the first computation. This suggests it could have gone further, had it not stopped due to the low strengths and high stresses at the end of the cantilever. The total construction time is 3 hours 44 minutes, which is an increase of 8 minutes (+4%).

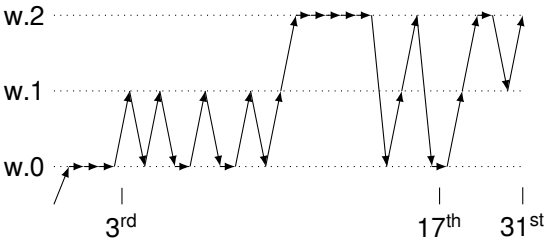
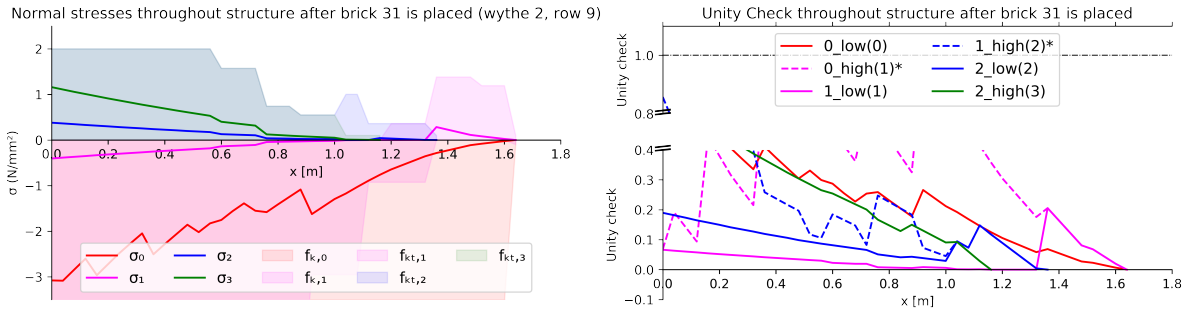
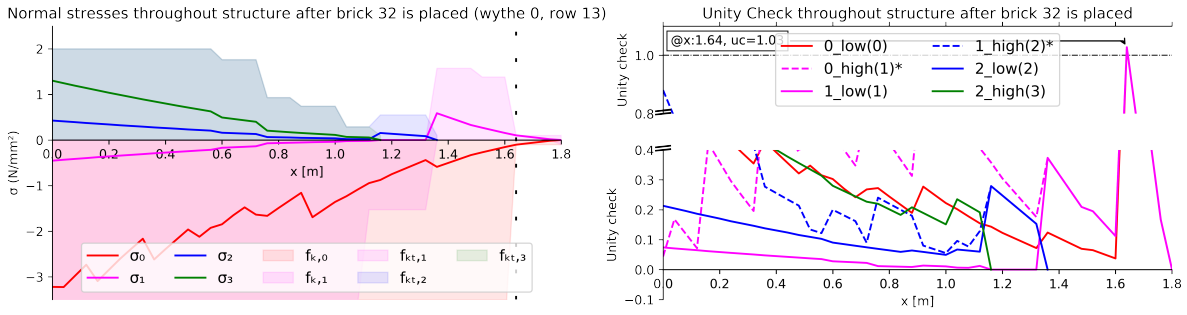


Figure 11.17: The construction sequence of the eigth computation.

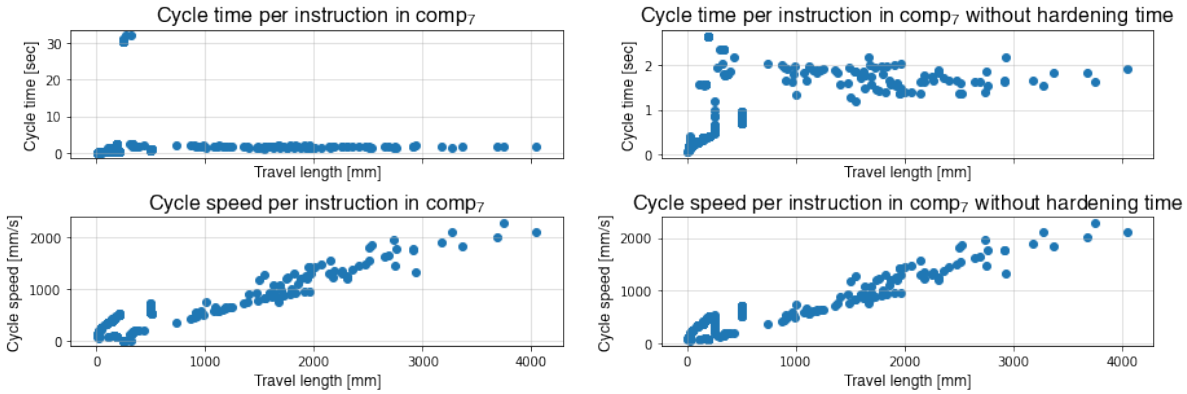


(a) The stresses, strengths and unity checks after the thirty-first brick;

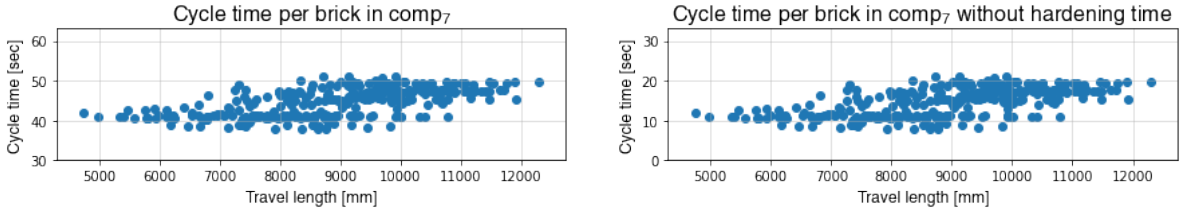


(b) The stresses, strengths and unity checks after the thrity-second brick; the dashed line in the left graph shows the x-position of failure, labelled in right graph as well;

Figure 11.18: The results from the structural analysis of the eighth computation. The x-axis is the cantilevering length of the structure. On the y-axis are respectively the stresses (in MPa) and the unity check. In the left graph four lines are shown, which are the stresses (σ) at the interfaces (see figure 6.5). Additionally, five strengths (f) are shown as an area. For the second interface two strengths are shown. In the right graph six unity checks are shown. Besides the four from the stresses on the left (solid lines), two additional checks are shown as well (dashed).



(a) The time the robot needs to perform this instruction (top) in the eighth computation, and its accompanying average speed doing that (bottom), with the hardening time included (left) and excluded (right, note the change in values of the vertical axis);



(b) The time the robot needs to place each brick (top) with (left) and without (right, note change in vertical axis) the hardening time;

Figure 11.19: The construction time results of the eighth computation. Where possible have axes stayed the same. Points of interest visible here, but similar to previous figures, are not repeated.

11.11. Overview & final cantilevers in design model

In figure 11.20 all vaults from the design model, based on each computation, are shown. They show the completion of the vault and each wythe. Table 11.4 shows the results of the full assembly of the vaults per computation.

Table 11.4: An overview of all computations and their total assembly results. Additionally representative values from literature are presented as well.

Computation	Bricks	Time [min]	Length [m]	Time per brick [sec]	Length per brick [m]	Bricks per workday	” ” without hardening
$Comp_0$	303	216	2.702	42,8	8,92	673	2.250
$Comp_1$	303	223	2.704	44,3	8,92	650	2.014
$Comp_2$	351	259	3.223	44,3	9,18	650	2.014
$Comp_3$	321	240	2.903	44,9	9,04	641	1.933
$Comp_4$	303	1.590	2.881	314,8	9,51	91	1.946
$Comp_5$	78	59	670	45,1	8,59	639	1.907
$Comp_6$	303	242	3.148	47,9	10,39	601	1.609
$Comp_7$	303	224	2.705	44,4	8,93	649	2.000
2 masons						110[sec. 3.3]	1200-2000

(a) Final vault from the design model, with computation 0;

(b) Final vault from the design model, with computation 1;

(c) Final vault from the design model, with computation 2;

(d) Final vault from the design model, with computation 3;

(e) Final vault from the design model, with computation 4;

(f) Final vault from the design model, with computation 5;

(g) Final vault from the design model, with computation 6;

(h) Final vault from the design model, with computation 7;

Figure 11.20: Design models after each computation and their full construction.

12

Discussion

In this chapter the three models, the used values and the results are discussed. The discussion focusses on three points: the applicability of the models, the reliability of the values and the remarks on the results.

12.1. The applicability of the models

This research has been split into three parts, each with their own model. The design model defines the structure and the bricks position and orientation. The engineering model analyses the strength and stresses occurring in the structure and determining in which order construction has to commence. The robotics model translates the CAD-model for CAM-purposes. The model also simulates the robotic movements to retrieve the construction time and length.

12.1.1. Design model

The design model was originally intended to be handled by additional software in Grasshopper. Based on a preliminary literature review, promising plug-ins like BrickDesign seemed applicable for further use. However, the thin-tile vault is a complicated structure and masonry is little applied beyond walls. Two difficulties in this research, which have not been fully explored, are the positioning of the brick pattern along non-primary curvatures and the reassurance the bricks have a proper distance in-between.

The design model is therefore limited to single-curvature vaults only. The definition of the thin-tile vault in this design model, however, may allow an easy expansion into more complex curvatures and eventually free-form vaults. The vaults in the research by ETH Zürich are based on Rhinovault and therefore catenary curves. When the (free-form) vault is partitioned in sections with consistent curvature, the same procedure as presented in this research is applicable. The model also allows for an expansion into more wythes than the used three. In one research at ETH, ribs were even applied to the vault as well. As long as the vault base surface is shaped to allow these ribs, the author deems it possible this can be implemented in this design model as well, with relative ease. It is unclear how the Block Research Group from the ETH made their brick patterns, with some indication in their research on voussoirs. With the use of skilled masons, it may not be necessary to provide a brick pattern in such detail as in this research. Additionally, mathematics may allow for even better definitions or widen the range of complex geometries that can be used with this model. An example is to combine this brick pattern definition with Gaussian curvatures.

The single-curvature allowed for an easy inspection of the voids between the bricks. It is clear that the thickness of the adhesive is suitable for the dimension ranges of this model. However, two improvements would allow a more conssistent gap between the bricks. First of all, if each wythe is created from their own vault base surface, the distance between points may not have increased with each next wythe placed on top. Furthermore, the curvature of the vault can provide a maximum and minimum thickness of that wythe's mortar beds, before the bricks collide with each other due to that curvature. Additionally, this can also provide the difference in distance between the top of the bricks and the bottom, both per course as well as between courses. This is true as well for the distance between the bricks per wythe.

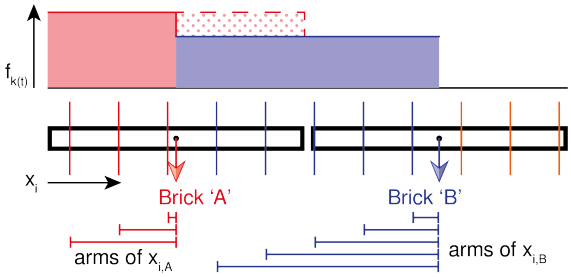


Figure 12.1: X-positions 'belonging to' the bricks 'A' and 'B' where the assigned strengths ($f_{k(t)}$ in MPa) for those x-positions is shown above and the 'long' arms for the stresses. The dashed red area is the actual strength of that section, if the strength and stresses were assigned separately.

12.1.2. Engineering model

The engineering model is a final product based on an exploration and analysis in a multitude of software. In the end the information from Grasshopper has been combined with Python to produce the results. Using Python within Grasshopper has proven to give a slow computation speed. Thus, moving to external Python software was necessary. The benefit of this interaction, which may have been possible as well with other software, is the ease to transfer data between the two programs with the help of csv-files.

The structure in the engineering model is a simplification of the design model. This leaves a lot of room for further research. Three points especially are of interest: the simplification to an arch, the discrete analysis of the cross-sections and the assignment of the strengths per cross-section.
The vault has one curvature, has a thin cross-section and is both wide enough to distribute local force concentrations and small enough to be homogeneous. This makes the simplification to an arch justifiable. However, during construction asymmetry occurs along the vault's path when half of the bricks of a row have been placed. This could result in additional stresses that have not been accounted for in this model. Furthermore, the simplification will create problems when other design types of the thin-tile vault are used. Double-curvature, thicker cross-sections and larger dimensions will not allow this simplification.

In the engineering model the bricks have been reduced to one row and these rows again to a set of points with which the structure is analysed for collapse. The reduction is part of the simplification of the arch, but collapsing the bricks of one row to one point may result in less favourable situations. To give two examples: the wythes analysed have an angle of -45°, 0° and 45°. The design model has had a limited test and seems to be working for more angles, like 30° and 60°. This will result in a problem for the engineering model, though. Since bricks in one row may deviate a lot relative to the average centre point of the entire row of bricks, the occurring stresses will differ greatly per brick. The distance of these bricks to the support may differ too much compared to the average centre point. The second example stems from the overlapping of the bricks. In the model this is done by the starting and end points of the entire row. If a next row of bricks is analysed for placement, it could be denied, since one of these bricks would have a greater cantilever than one of the bricks already placed. However, these two bricks may not be close to each other. Thus, the deny is invalid. In this same alley, by using a set of cross-sections to test the unity check, it is possible an exceedance has occurred, but it did not happen in any of the provided cross-sections. Although the linear-elastic analysis has proven to be the correct analysis to use, a more extensive analysis will provide a better reassurance the strength has not been surpassed by the stresses.

The structural analysis in Python also leaves room for improvement. The stresses and strengths are assigned to these average centre points as well. A placed brick results in stresses in a given cross-section when the centroid of that brick is further from the support than the cross-section. This results in lower stresses in a given cross-section when the centroid of a brick is not further than this cross-section, but the cross-section is through that brick (see the first two blue lines in figure 12.1). This is of little concern, since the largest part of the stresses is due to the cantilevering length, which is small in this case. However, what goes for stresses, also applies to strengths. Again the cross-section is further than the centroid. This results in no stresses and no strength for this cross-section. Now the next row of bricks is placed. This cross-section will receive the stresses and strengths based on this row of bricks, making it similar to the other blue lines in figure 12.1. Although this does not matter for the stresses, it does create a problem for the strength. Now the cantilevering length is relatively large, while the strength is relatively low.

In reality, it would have a higher strength, since the cross-section is actually in a brick which has been placed earlier, and therefore has a higher strength. This makes the analysis more conservative. In

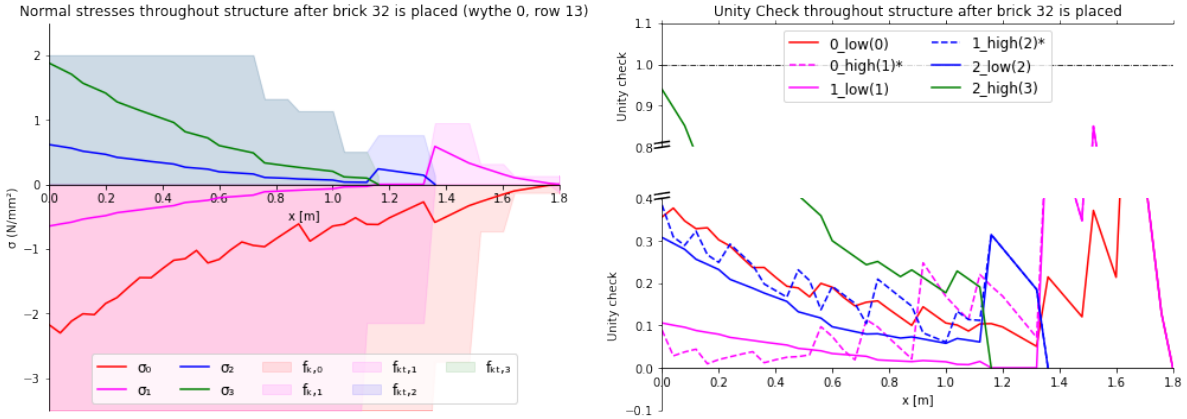
figure 12.2 the first computation is done with a slightly higher initial strength. Now 'brick 32' can be placed (wythe 0, row 13). This was not possible in the first computation, see figure 11.3 One more row hereafter and the tensile stresses at the support would become too high, which is independent of the problem described in this paragraph. However, with this last row placed, the first wythe would have reached the apex of the vault. Qs stated in section 7.1, the construction sequence based on a.l.a.p. gives a longer construction sequence. Figure 12.3 and section 11.10 indicate this as well.

With a.l.a.p. up to 36 rows can be placed, instead of the 32 with a.s.a.p. and the 31 with the lower initial strengths. 36 is not the maximum due to the structural analysis, though. Here the first wythe has reached the apex, and the other wythes cannot be placed without overlap. Thus, even more bricks can be placed than required with a.l.a.p. and higher initial strengths, but the structural analysis has not been checked if it could take the downwards curved part of the vault.

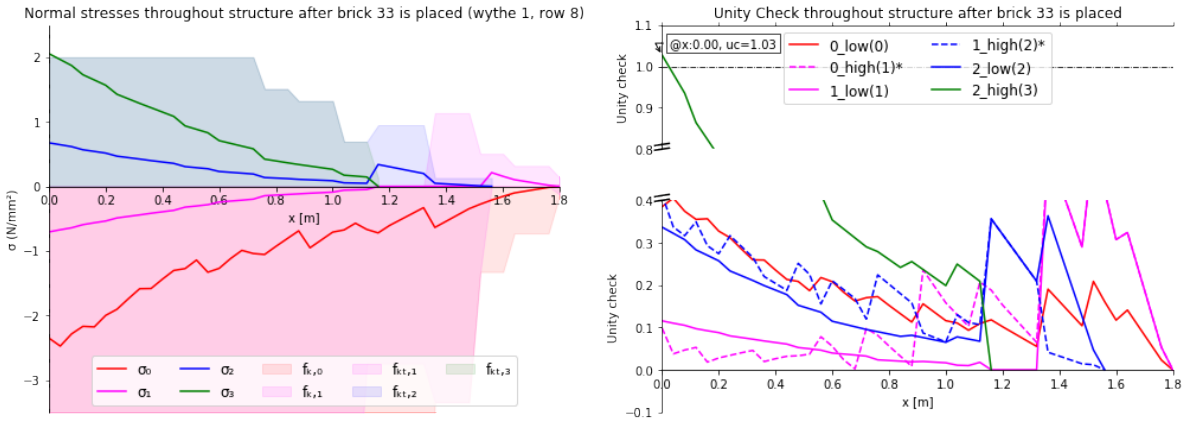
12.1.3. Robotic model

The robotic model provides all the poses to be applied on the robot. Each pose is made from the plane describing the geometry. The robotic instructions and programs are created while importing the poses. This results in simulations per instruction and per program. In both the robotic model and in the simulation improvements can be made.

The current script importing the poses and creating the instructions and programs, automatically makes everything joint moves. Joint moves have the benefit that they are the fastest movement possible, since the joints move as little as possible. However, with certain instructions it is not the robotic movement that is governing, but the movement of the object. The instructions for picking and placing, for instance, require

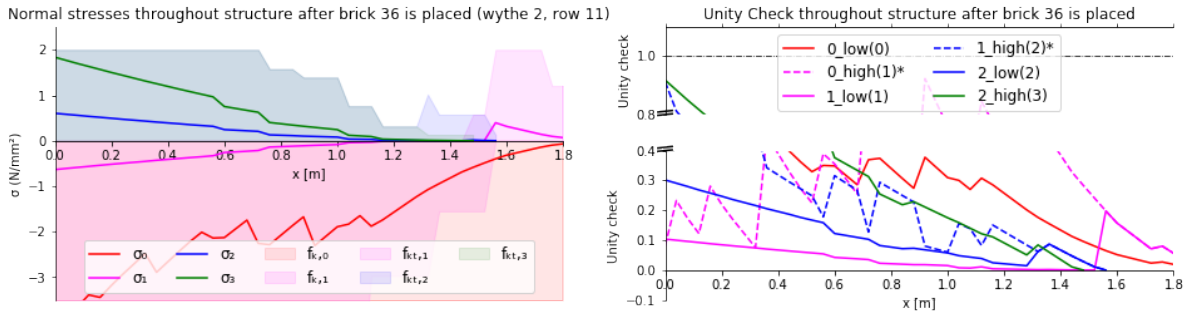


(a) Left: Stresses (lines) and strengths (area) of the structure after the thirty-second brick has been placed; right: the unity check through the whole structure, shown at the four interfaces from figure 6.5;

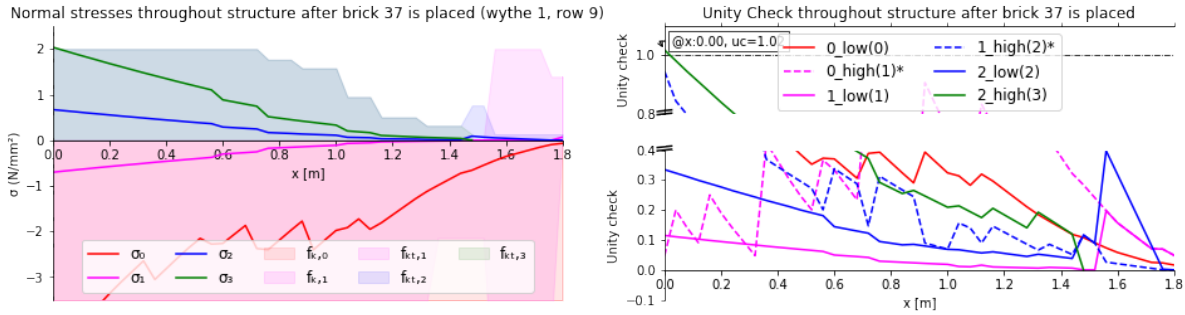


(b) The stresses, strengths and unity checks after the thirty-third brick;

Figure 12.2: The stresses, strengths and unity checks of the first computation with the initial strengths raised to 0,13 MPa (+0,03 MPA). Shown are the last possible row of bricks to be placed and the row placement resulting in failure.



(a) Left: Stresses (lines) and strengths (area) of the structure after the thirty-sixth brick has been placed; right: the unity check through the whole structure, shown at the four interfaces from figure 6.5;



(b) The stresses, strengths and unity checks after the thirty-seventh brick;

Figure 12.3: The stresses, strengths and unity checks of the seventh computation with the initial strengths raised to 0,13 MPa (+0,03 MPA). Shown are the last possible row of bricks to be placed and the row placement resulting in failure.

the object to move in a predictable manner, to ensure it does not collide with the other objects. Therefore, linear movements should be used here instead. However, that requires a manual alteration in the robotic simulation after the automatic creation of the instructions. The joint movement of the robot is only optimized from one pose to the next. This may result in additional movement (like a full 360° rotation around one joint axis) for the next instruction that would not have happened if the first instruction rotated more favourably. Joint moves are able to solve this by rotating around that axis, or performing a 'reset' where the robot moves the end effector far away from both source and target, before approaching the target again. Linear movements are constrained and may result in an invalid move. Thus, the joint move is used to ensure a simulation can happen. The model does allow the generation of linear movements for the appropriate instructions, but this has been turned off currently for each simulation. It would require a lot of manual fixing to solve this issue.

One way to solve this, is to calculate the movements beforehand, and to analyse which joints rotate around their axis (almost) completely. Then, the previous instruction should be altered to end that instruction where its next one no longer has this full turn. Of course, this altered instruction should be feasible as well. This operation could be automated with the help of programming and could be implemented in the import script used in the robotic model.

To continue with the movement of the robot, the speed of the robotic arm is also an important factor at play. In general, the movement of the end effector or the rotation of the joints is limited by their acceleration and velocity. The acceleration of the joints is generally not provided by the specifications of the machine. Only the maximum speed (°/sec) and the maximum torque (Nm) (and based on that the moment of inertia (kgm^2)) are provided. These may be used to determine the maximum acceleration, but the best way is to test this empirically per robot and per (full) brick.

The movement of the end effector can be quite high if no confinement is present. In RoboDK the maximum velocity and acceleration of the robot are set as 200°/sec and 150°/sec^2, respectively. As figure 12.4 shows, the cycle times found in the results follow this theoretical cycle time. The figure shows that when a joint makes a full rotation (360°), the time taken is a little over 3 seconds. The maximum velocity is of little importance, though. Only with rotations large enough, is this velocity reached. In this case, rotations need to

be larger than 267° (time taken more than 2,67 seconds). With the Kawasaki RS015X, the lowest maximum speed is 180°/sec. This is however in joints (the first and second) which move little due to the configuration: no rotation around the base is large enough to reach 180°/sec. It is therefore useful, when a simulation is present, to test which joints will reach their maximum velocity and use that to determine the lowest maximum velocity to be used for the robotic simulation (200°/sec for this simulation).

Furthermore, the simulation is limited to the extend of software development. In that regard major steps have been taken to ease the transition from digital architecture to digital fabrication. This process required less work than the original intention, which was: the robotic movements had to be done and calculated more before any simulation would be possible. In that regard, the robotic model and simulation turned out to be quite the opposite in terms of its state of the art, compared to the design model. The articulated six-axis robotic arms are clearly well developed and implemented in the field of robotics. These have also been implemented in the construction industry, though still in its developmental stage, see section 8.2. Of course, besides the development of software for simulation, empirical testing and case studies are required as well to improve a theoretical model.

It also shows further research with other robot types may result in better outcomes. With the maximum reach of robotic arms currently topped at 4000 millimetres and their weight at hundreds of kilos, it is questionable whether these would be useful for construction sites. Two approaches currently made ready for commercial use (little arm on a wagon and an off-site truck) show how this could be overcome, but as figure 1.3 illustrates, robotics is still a developing technology. The improvements in drones and insect-like robots may result in an improved feasibility for masonry construction as well. Additionally, the current construction time is primarily limited by the hardening time of the adhesive. A development in infrastructure construction may improve this process. Figure 12.5 shows the use of suction cups to attach, move and place multiple bricks at once. Using this method, more attention is required for the pallet station, where more specifications on the arrangement are needed to allow this method. Moreover, the suction cup tool needs to be designed with this purpose in mind: as the structural analysis has shown, it is best to group the bricks together based on their wythe and distance to the support, if they are grouped together. This has to be taken into account for the design.

When using robotic arms stays for the foreseeable future, more configurations can be explored as well. In this research the robot has been placed outside the final used space by the vault and underneath the vault. With more complex or larger structures, the reach of 4000 millimetres (or 3150 for the Kawasaki) becomes too constricting. The use of conveyor belts or other robotic support systems may increase their work space. Simultaneously, the initialization and calibration of such a work space will take more time as well. Configurations for more complex structures or more complex work spaces could be to use multiple robotic arms in a similar fashion as figure 1.6. Furthermore, when the weight of the robotic system will be less, placing the robotic

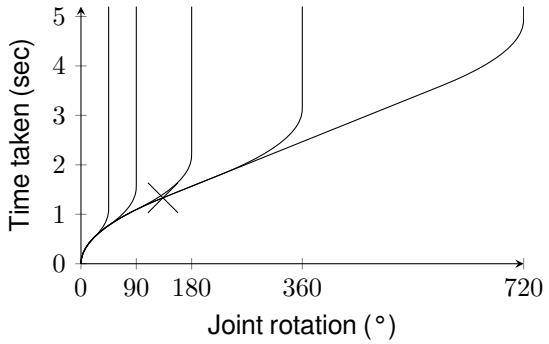


Figure 12.4: The theoretical time of a joint rotation when the rotation is 45°, 90°, 180°, 360° and 720°. Indicated is the moment when maximum speed has been reached.

Figure 12.5: The placement of paving bricks with the help of a vacuum suction cup.

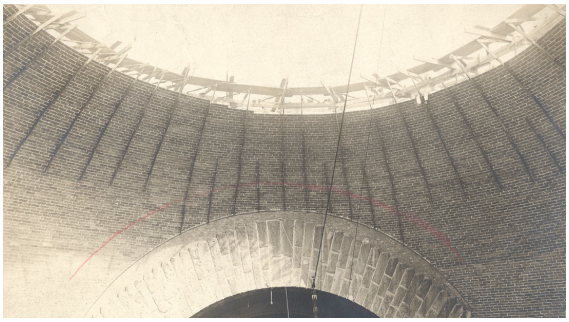


Figure 12.6: The construction of the dome of Saint John the Divine. On top are the work stations for the masons.

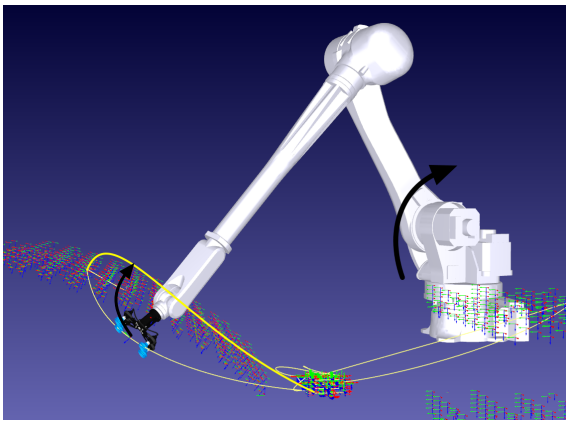


Figure 12.7: Avoiding object collision by changing the orientation of the second joint (rotation indicated with black arrow), which results in a slightly changed path (indicated with black arrow and second yellow path).

arm on top of the part of the vault already constructed becomes an option as well. This has been done a lot by Guastavino, see figure 12.6. However, such configurations only influence the total travelling length and less the total construction time. This is still primarily dependent on the hardening time. Nevertheless, when an optimization for the travelling length is considered, changing the configuration and the robot used may reduce the required motions of the robot and reduce the distance travelled by the objects, since the stations can be placed closer.

The design model has been set up favourably for the robotic model. The definition of the planes of the bricks is based on the centroid of the brick projected onto the surface where the robot needs to grab it. The normal of this plane is always pointed to the intrados. However, when the design model can be omitted and a provided design is the starting point, the bricks may need to be redefined to ensure these planes are oriented favourably for the robotic model.

In the simulations used to get the results, the bricks were absent. This has been done due to computational performance, but would have had a significant impact on the tool paths found in the simulations. The most notable change can be expected in and near the vault station. In the results, the end effector approaches the vault station partially from underneath the vault. In other words, the robotic arm moves through what has been constructed already. Additionally, the approach path for placement of the bricks will be more complex as well. The program cannot use a strange move, described earlier with the joint moves, to get to its target. This would have little influence on the total construction time, due to the hardening time, but may have changed the joint moves slightly to accommodate the objects. Although the change should not be exaggerated. The arc-like tool path due to the joint move would remain similar and with the change of one joint, the problem can be solved. The second joint (just above the base) may retract more, which allows the end effector to be higher, above the vault, see figure 12.7. This will likely result in (small) changes of the other joints as well.

Already the computation speed of the simulation can take quite long. Importing the poses and creating a program for each instruction, brick and the total assembly takes quite some time. Reducing this computation time has taken place by reducing the number of instruction programs for the simulation and by omitting the brick objects from the program. As stated previously, this may give different results when it was simulated and computed to such an extend. It would require more time though to make one simulation than it would to use a robot and build the structure. Hence, why such computation times would be counter-productive. When the structure or the configuration would become larger or more complex, this could become a problem again. Alternative computation strategies should be sought after to solve this problem. Otherwise the current computation strategy should become more optimized and sophisticated from a computer science approach.

Last but not least, there's some major differences at the moment between the robotic model & simulation and if it were to be constructed. One part of the simulated time, for instance, has not been taken into account. It can take a long time to prepare the site for construction, see section 8.2. This time is entirely case-dependent though and therefore no estimation can be given. This has an impact on any comparison with other construction

scenarios.

Another difference is the real-time scanning of the environment and process. The simulation, as is common with simulations, does not include the tolerances of the objects & adhesive nor of the robotic movements. For each a strategy can be proposed. The objects can have a stricter tolerance threshold to reduce the deviance of the objects from the assumed and simulated dimensions. Furthermore, the adhesive station can apply a thick layer, more than enough to cover the beds + any tolerance difference. This will require more cleaning and finishing of the vault, to ensure no adhesive will be dripping down. The tolerances in the robotic movements depend on the instruction. The bigger and faster movements, where tolerances are more prone to happen, are free from any object. This is not the case for their start and finish, where the speed, and thereby the tolerance difference, is lower.

Scanners and other measuring equipment may be of use to ensure the tolerance remains as little as possible. For instance, at the pallet station the position of the end effector relative to the brick can be measured right before the robot picks the brick. This can include the rotation of the last joint axis, and therefore the end effector, to accommodate any unintended misplacements of the bricks in the pallet. The tolerance issues at the adhesive station have been covered already, although a problem may arise if the stacking of tolerances becomes too high to solve with a thickened adhesive layer. At the vault two solutions can be used. The first is to perform a scan similar to the pallet station, but this involves a scan with 3 degrees of freedom to check, making it more complex. Alternatively, the robot can also place a brick, after which a scan informs whether the next brick has to compensate any tolerances or not. Again, the thickness of the adhesive plays a major role in compensating tolerances.

The strategy with traditional masonry is quite similar, see figure 12.8. Here, the mason places a thread from one side to the other. This thread shows the mason where the next course of bricks should reach to. If the brick is over this line, the mason presses on the brick to reduce the mortar bed. If the brick is under this line, the mason either tries again, or makes a thicker mortar bed for the next course. The thread here is similar to scan technology. Laser technology comes to mind especially. Additional pressure from the robot to reduce the mortar bed in any direction could be a solution as well. However, this would require additional structural analyses.

12.2. The reliability of the parameter values

The design model solely uses geometric properties as its values, or relative geometric properties like the overlength factor. Issues of tolerances could occur, as described in the previous section, but these should be no different than would be the case for other structures. A higher requirement of significant figures for the geometric properties reduces this problem. Similarly, adaptive design with real-time data could compensate for any tolerances arising. The lack of a test for the maximum and minimum occurring adhesive width is missed in the design model for further testing and verification.

Literature on the materials used in this study has been scarce. The interest for engineers, understandably, has been on finished products and when that finished product has been reached. However, when robots are to be used, a better understanding is required, where the experience of a skilled craftsman, like masons, should be known for the construction model. The construction industry often relies on 'best practices' or directives, especially when applied to the construction phase itself. The gap between theory and practice within the construction industry has, as of yet, not been closed. Especially structures like masonry have gathered more attention, but the material still has a deficit to make up compared to concrete and steel.

Three aspects in this study are related to the material properties. The initial strength of the material is set as 0,1 MPa, which aligns with the

Figure 12.8: A thread used for alignment of the brick course to the plans.

brick experiment in appendix C. This strength represents the strength to withstand the weight of one brick added, while the time to reach this

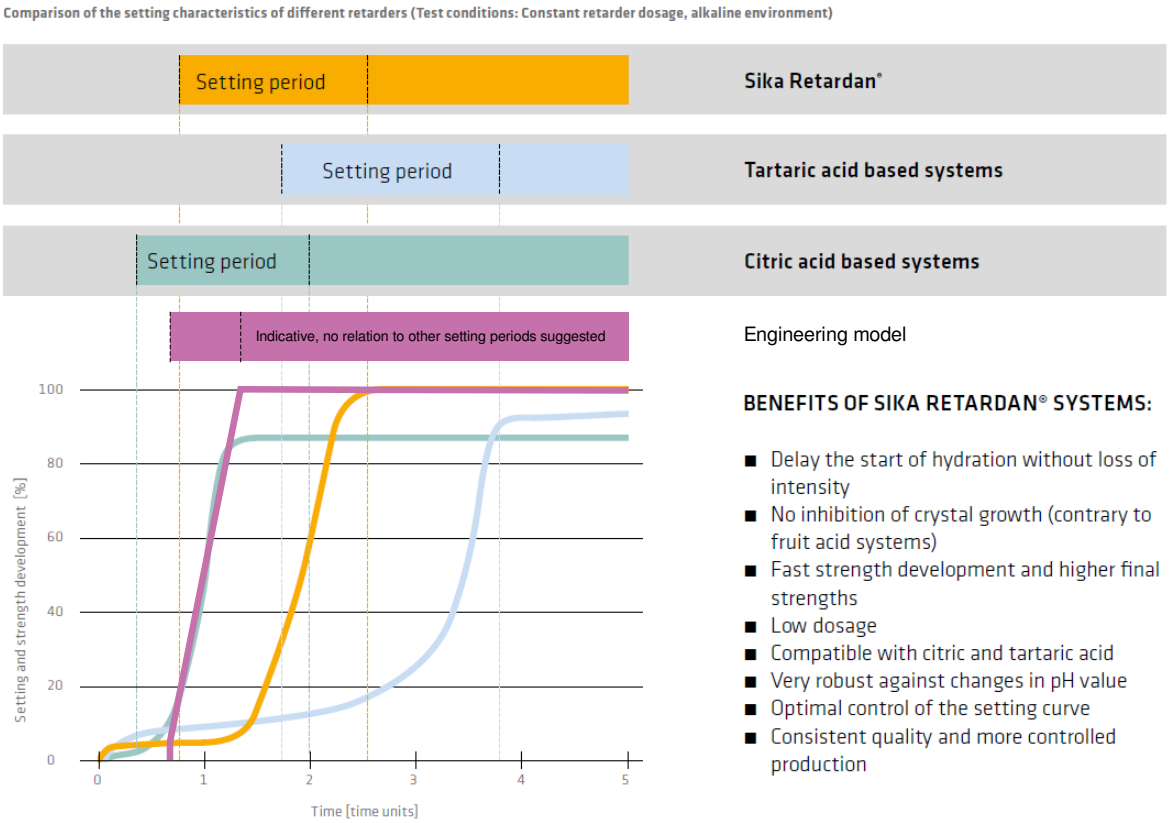


Figure 12.9: The strength development of various retarders, with an indication of how the shape of the curve of strength development used in the model relates to those shapes of the retarders. | (SIKA, 2021) (altered)

initial strength is set similar to the pot life of the epoxy, as a safety and ensurance precaution, also based on the brick experiment. With both values further research could provide a better validation and even optimization for the used values. In other research the time aspect of epoxy hardening has been of less importance, which may have increased values found there, like the 12-15 minutes per brick for the glass vault in figure 1.6 (Parascho et al., 2020). On the other hand, videos of construction of the Catalan vault or dome have shown that merely seconds would be enough for gypsum plaster or cement mortar to work (Davis et al., 2012). *'When has the strength of the adhesive developed enough to withstand the self-weight of the attached object'* remains a mostly unanswered question.

The second aspect is the strength development between the initial and the final strength. The formation process for plaster has two clear stages, while epoxies seem to have an S-shaped curve for their strength development, see figure 12.9. If a rapid strength development occurs in the beginning, the failures found in the structural analysis of the first and last computation may be even less of a problem. This shifts the focus of the preferred construction sequence entirely to the stress development in the later stages of the construction, mainly at the support. If the beginning shows a slow strength development, this focus is reversed and the stresses in the entire structure should be kept as low as possible. Alternatively, the hardening time could be expanded, which raises the initial strength as well. That would mean an increase in construction time as well.

The last aspect is that of the final strength of the structure. Although research like Guastavino's has shown that a tensile strength of 2 MPa can be expected for the final structure, more understanding is required. This is most important for the reason behind the increase in tensile strength. Although it is obvious and easy to assume this to be due to the alternating layers (removing those mortar-brick interface planes spanning the entire cross-section), it seems no experiment has been conducted to proof this undoubtedly. If the hypothesis is true, it is interesting to expand this property to other structural typologies as well, besides the vaults.

Masonry walls capable of withstanding more flexural strength, and maybe more shear strength as well, could be beneficial for the applicability of masonry. Similarly, repairs and maintenance could be improved by adding a layer of thin-tile bricks to the existing structure. In that regard, the resistance of dynamic forces may be the most urgent interest to be looked into for the Netherlands. The result could be the removal of the need for complex (steel) reinforcements or complex repairs of historic masonry walls. Further research is required in this area.

The robotic model and simulation do not have any numeric parameters for which a reliability issue could arise. The given values from the robotic arms, like their maximum speed, their weight and so forth, are the only values used. However, as stated before, testing a robot before using it should always be done. Therefore, a test with the used robotic arms may have given slightly different results, but this should have little impact on the overall results.

12.3. Remarks on the results

The results from chapter 11 show that the limiting factor is the tensile strength of the structure. The maximum span, which has been limited due to the extend of the robotic arm, should be limited even further to achieve the reach of the apex and therefore the completion of one half of the vault. However, as section 12.1.2 shows, this could be reached with an improvement on the structural analysis. With the preferred construction sequence *a.s.a.p.*, this would not give a major improvement, but with *a.l.a.p.* the possible bricks to construct even go further than the apex. The increased time differs little and could be due to small differences between the design models and automated judgement of the simulation program on the perceived best movement. Additionally, the importance of the maximum dimensions outweighs the importance of such little time differences.

Additionally, changing the shape (increasing the factor) has little consequences for the extend of the cantilevering vault. However, even higher or lower factors could give different results. After all, the first wythe may have extended similarly, but the second wythe did not. The increase in weight may not be downplayed entirely by the increase in normal force and the decrease in cantilever length per brick. From a building engineering approach, the reduction of the factor would be beneficial. From a structural or from an architectural approach the increased factor could be sought after. The desired overlength depends on the desired outcome and function of the vault, but it is clear that at least for the range of 1,1 to 1,25 for the factor of overlength, the maximum span of the vault is similar.

Although only the first computations have the results on the time per station, these already make it clear that a majority of the time is spent at the vault station. Additionally, the adhesive station requires the most time for all the movements to be made. This is no surprise given the higher number of movements required at the adhesive station (at least four compared to three at the other stations). The first and third computation (base & factor) show little difference between the times per station. However, using a different robot does result in larger differences. Including the lengths per station in this comparison, it seems the Fanuc requires more complicated movements than the Kawasaki. The Fanuc has lower maximum speeds for the third and fourth joint ('elbow'), but higher for the fifth and sixth ('hand'). The Fanuc performs better on range as well. No more 'reset' movements should be expected with the Fanuc as with the Kawasaki. One explanation could be that the fifth and sixth joints never reach their maximum speed, while the third and fourth may do. Additionally, the slightly different dimensions of the robotic arms may result in more complicated moves for the Fanuc. These two explanations seem likely, but it has not been and cannot be proven with this study.

The cycle times per brick show an increased time for an increase in instructions. This is not surprising given it is the joints rotation combined with large travel lengths that give the total cycle time. This suggests the cycle time for bricks with a lot of instructions (28) could be half if these would have the least number of instructions (10) as well.

The cycle times per computation have resulted in an area somewhat similar to a triangle: with a minimal length boundary, an upper boundary and a lower boundary. No reason can be found from the research for this result. However, it could be that this would be the case for other robotic constructions and applications as well. The convergence to one point at maximum travel length suggests that the longer the end effector travels, the more likely it is finished with the joint rotations, with the longest taking joint rotation always at the same joint. This longest joint rotation determines the cycle time, which would vary little over the increasing travel lengths with a high speed and low additional rotation. The

linearity of the speeds suggests this as well. Furthermore, the change in the spread at the other configuration (computation 6) suggests that each configuration may have its own spread of cycle times. It would be interesting to see if any relation or prediction can be found of the spread of cycle times related to the configuration of the work space.

In the robotic simulation only a part of the instructions could be simulated, and therefore shown. It could be that due to this certain instructions are not shown that otherwise would have been a point of interest.

13

Conclusion

The main question of this research has been:

How does a robotic construction of a parametrically designed thin-tile vault perform based on step-wise structural analyses? To answer this question the set of sub-questions have to be answered first:

1. What is the parametric model of a thin-tile vault?
2. What stresses (can) develop in the construction of the thin-tile vault?
3. How does a robot construct a thin-tile vault?

What is the parametric model of a thin-tile vault? In part II a parametric model of a thin-tile vault has been created. A thin-tile vault is a masonry structure that consists of multiple layers of brick and mortar (or other adhesives) where the bricks are placed flat and have a relative low thickness and where the mortar (or other adhesive) is fast-setting and provides support during construction. The thin-tile vault is different from other forms of masonry, due to its capability to endure tension. The multiple layers create a mixture of brick and adhesive in any cross-section. This increases the adhesion or bond strength between brick and adhesive, which is generally considered weak. This allows the thin-tile vault to be calculated with a linear-elastic analysis.

The thin-tile vault in this parametric model is a barrel vault of single curvature made of three wythes. The shape of the vault is similar to that of a catenary arch, which is in line with recent research on thin-tile vaults and free-form vaults. The bricks are placed in a stretcher bond in directions of -45°, 0° and 45° relative to the non-trivial primary curvature. The form of the catenary vault is based on the overlength versus length of an arch.

In the model the centre point approach is used to place the bricks adequately in the vault. This has resulted in a filled surface with gaps neither too small nor too large for an adhesive to be applied within. The bricks used in the model are based on the common Waal format bricks (210×100×50). The thin bricks are made in a similar way as slip bricks for façades are made.

What stresses (can) develop in the construction of the thin-tile vault? The thin-tile vault from the parametric model has been simplified to an arch. During construction the arch behaves like a cantilever, until a thrust line can be created. The load of the structure is its own weight, which is non-uniform distributed along global Cartesian coordinates. The most significant force is the normal stresses due to a moment in the structure. The normal force and shear force, both derived from a vertical directed shear force, have little influence on the occurring stresses in the structure. A phased structural analysis is done after each set of bricks with similar cantilevering length (a row of bricks) had been placed.

The key locations where the stresses may exceed the strength depend on the construction sequence. Two possible locations have been found. The tensile capacity has been reached at the support, which is likely to occur in the upper wythe, but may also occur earlier, before a brick of the upper wythe has been placed. The strength has not adequately developed, which is likely to occur at the second to last row of bricks placed, where exceedance of the tensile strength is more likely.

In literature, video evidence and through buildings still standing it is clear that bricks and Plaster of Paris (quick-setting gypsum plaster) are able to provide a tensile capacity for the

construction of thin-tile vaults. Literature does not provide any quantitative evidence and thus only empirical evidence exist. This makes the use of Plaster of Paris for the construction of thin-tile vaults unsuitable when applied in a theoretical framework. Epoxies are a quick-setting adhesive as well, with good bond strength and tensile capacity. The function type/shape of hardening of epoxies is known, at least better than that of gypsum plaster.

It is unknown how gypsum plaster hardens over time. Literature shows research on the strength development in common time stamps for mortar and cement materials. These stamps are trivial for the construction of any structure, due to the high construction time. Some epoxies have a much shorter setting time and research shows time stamps closer to usable time periods for any construction. Although values are not present, the function type of epoxy hardening is that of a logistic function, or an s-shaped curve. Brick has time-independent material properties when it comes to the construction phase. Due to the higher tensile capacity, the combination of brick and adhesive will likely result in a higher flexural strength than found for traditional masonry, and as found in the Eurocode.

How does a robot construct a thin-tile vault? Articulated six-axis robotic arms are common in the construction industry and provide a flexibility required for the construction of the thin-tile vault. Specifications of the arms differ greatly, but the weight and the reach of the robot are its greatest limitations in use. The weight of robots with a high reach is similar to or even greatly surpassing common values used for most structures as a distributed load. The reach is also limited to 4 meters in any direction. With the help of linear tracks this can be increased in one direction. The speed of robotic arms is a compound of their angular joint speeds, and is limited to the maximum torque the arm can handle.

A CAD model can be adjusted for the use of CAM. For robotic arms or other end effector machines, the CAD model needs to provide the poses of the end effector. A pose consists of six coordinates, three are the Cartesian coordinates, three are based on Euler angles. More information, like the speed or pause time can be used as well between CAD and CAM.

How does a robotic construction of a parametrically designed thin-tile vault perform based on step-wise structural analyses? A robotic arm can place roughly 2000 bricks per day in a thin-tile vault. This makes it faster than placement by masons, and on par with other masonry robotics like SAM100. This does not take into account the time it takes for the adhesive to harden and it excludes the site preparation before construction can commence. Including the hardening time, the number of bricks reduces to over 600 bricks per day, which is on par with slow masons constructing a wall and exceeds the 110 bricks placed by two masons for the temporary pavilion at ETH Zürich. Although the pot life of 5 minutes has shown that the more realistic or mid-range value of the hardening time evaporates this difference completely.

The hardening time is the primary influence of the total construction time. Any measurement or strategy used to reduce the total time waiting on the hardening will have a (major) impact on the total construction time. If gypsum plaster has been proven to have a favourable strength development, its setting time of 5 to 20 seconds could already reduce the total construction time by 25 to 60% compared to the computed results, and even 94 to 99% compared to the increased pot life of 5 minutes, which represented the more realisitc pot life of epoxies. The minimal construction time, in the used configurations, remains to be at minimum 12 seconds per bricks, though.

The time spent per instruction is dependent on either the minimal joint rotation or the total distance to be travelled. The last one has been reduced in this research with a thought through positioning of the different stations, but different configurations could reduce the work space of the robot even further. Most important is to avoid any full rotation of the robotic arm around its base. This increases the time that instruction takes significantly. The minimal joint rotation is the fastest movement possible between two poses. However, this optimization is a local optimum. The poses could be altered to find a better minimal joint rotation. In the current configurations this has resulted in the minimum of 12 seconds. This could be reduced even further with a more extensive optimization.

The linear-elastic analysis of a thin-tile vault to an arch simplified by grouping the bricks per row has shown that quite a large cantilever can be created with thin-tile vaults. It is best to construct the wythes of the vault from the inside out: first the intrados and ending with the extrados. Additionally, two placement strategies can be used: the a.s.a.p. method and the a.l.a.p. method. The a.s.a.p. method starts with the intrados wythe,

but places the next row in the wythe above as soon as that row is fully covered on the bottom side by its inner or prior wythe. The added benefit is that it is easy to reach up and over the vault when the robot is positioned beneath the vault. The a.l.a.p. method also starts with the intrados wythe, but tries to postpone placement in the next wythe as much as possible. This is to reduce the tensile stresses in the upper or extrados wythes as much as possible. The benefit is that more bricks can be placed with this method, while using the same design.

13.1. Recommendations

This study has looked at the most important aspect of the thin-tile vault from an engineering perspective: the construction phase. Previous work from Ochsendorf and the Block Research Group has mostly been focused on the history; renovation of existing vaults; and the free-form design of vaults. For each study where a full-scale model was built, experienced masons from Catalonia had to be hired. With this study a first step into the robotic construction of a thin-tile vault has been made. The completion of this process has two major benefits. The amount of research on masonry (vaults) can be increased, since (expensive) masons and long construction times can be cut. Another benefit is that one of the factors resulting in the decline of thin-tile vaults, high labour costs, may become obsolete. With the first benefit (more research) the other factor for decline could be eliminated as well: a lack of understanding compared to concrete and steel. Furthermore, the author wishes the materials used in this study, brick and epoxy, may be replaced with (even) more sustainable materials, showcasing the progress of innovation.

The use of robotics for digital architecture has become easier as well and the integration with digital fabrication has come a long way. The three models (design, structural and robotics) all show further research is required. In terms of design a further expansion of plugins like Brickdesign could encourage more construction with masonry in the future. Research both in this thesis and at universities like the TU Delft and ETH Zürich show the geometric challenge of filling free-form spaces can be quite the challenge, but also very rewarding. With the engineering model and structural analysis it is clear a translation from practice to theory is still required with masonry structures and materials. If research into Plaster of Paris or other adhesives proves to be fruitful, this research is best to be repeated with these materials, likely to result in better outcomes. The robotics has shown that integrating a design & engineering model with robotics requires some attention, but can be easily implemented. The challenge here is to practice and improve on brick-laying, with the help of masons if possible. Especially further development in fit-for-purpose masonry robots could improve the feasibility of masonry construction with robotics even more.

References

Benfratello, S., Caiozzo, G., D'avenia, M., & Palizzolo, L. (2012). TRADITION AND MODERNITY OF CATALAN VAULTS: HISTORICAL AND STRUCTURAL ANALYSIS. *Meccanica dei Materiali e delle Strutture*, *4*(1), 44–54. https://www.unipa.it/dipartimenti/dicam/.content/MMS/MEMS{_}S-BENFRATELLO{_}4.pdfhttps://www.unipa.it/dipartimenti/ingegneria/meccanica-dei-materiali-e-delle-strutture/

Bock, T. (2007). Construction robotics. *Springer*, *22*, 201–209. https://doi.org/10.1007/s10514-006-9008-5

Ochsendorf, J., & Freeman, M. (2010). *Guastavino Vaulting The Art of Structural Tile*. Princeton Architectural Press.

Dugum, H. (2013). *Structural Assessment of the Guastavino Masonry Dome of the Cathedral of Saint John the Divine* (Master Thesis). Massachusetts Institute of Technology. Cambridge (Massachusetts). https://dspace.mit.edu/handle/1721.1/82711

Haas, C., Skibniewski, M., & Budny, E. (1995). Robotics in Civil Engineering. *Microcomputers in Civil Engineering*, *10*(5), 371–381.

Li, R. Y. M. Robots for the Construction Industry. In: *An economic analysis on automated construction safety*. Hong Kong: Springer Singapore, 2018. Chap. 2, pp. 23–46. ISBN: 978-981-10-5771-7. https://doi.org/10.1007/978-981-10-5771-7_2.

Haidegger, T., Barreto, M., Gonçalves, P., Habib, M. K., Ragavan, S. K. V., Li, H., Vaccarella, A., Perrone, R., & Prestes, E. Applied ontologies and standards for service robots. In: *Robotics and autonomous systems. 61*. (11). Elsevier, 2013, 1215–1223. https://doi.org/10.1016/j.robot.2013.05.008.

Bloss, R. (2014a). Robots have come to architecture to model, construct, fabricate and offer new approaches to create innovative designs, elements and structures. *The Industrial Robot*, *41*(5), 403–407. https://doi.org/10.1108/IR-06-2014-0359

Parascho, S., Han, I. X., Walker, S., Beghini, A., Bruun, E. P. G., & Adriaenssens, S. (2020). Robotic vault: a cooperative robotic assembly method for brick vault construction. *Construction Robotics*. https://doi.org/10.1007/s41693-020-00041-w
Interesting links over the history of bricklaying robotics. Paper consists of three parts: 1. Development of the fabrication method 2. Method underlying the design 3. Sequencing strategy State o/t Art Focus on brick construction and robots cooperating together 1.

Huerta, S. U. P. d. M. The Mechanics of Timbrel Vaults: A Historical Outline (A. Becchi, M. Corradi, F. Foce, & O. Pedemonte, Eds.). In: In *Essay on the history of mechanics* (A. Becchi, M. Corradi, F. Foce, & O. Pedemonte, Eds.). Ed. by Becchi, A., Corradi, M., Foce, F., & Pedemonte, O. Genova: Birkhaüser, Basel, 2003. Chap. 5, pp. 89–134. ISBN: 978-3-0348-8091-6. https://link.springer.com/chapter/10.1007/978-3-0348-8091-6{_}5

Guastavino, R. (1893). *ESSAY ON THE THEORY AND HISTORY OF COHESIVE CONSTRUCTION, APPLIED ESPECIALLY TO THE TIMBREL VAULT*. (second). TICKNOR; CO MPANY. http://www.antichefornaci.it/files/biblioteca/Guastavino{_}Essay{_}on{_}the{_}theory{_}and{_}history{_}of{_}cohesive{_}construction{_}applied{_}especially{_}to{_}the{_}timbrel{_}vault.pdf

Hartsuijker, C., & Welleman, J. W. (2007). *Engineering Mechanics: Stresses, Strains, Displacements*. Springer Netherlands. https://doi.org/10.1007/978-1-4020-5763-2

Como, M. (2017). *Statics of Historic Masonry Constructions: An Essay* (M. Frémond & F. Maceri, Eds.; 3rd ed., Vol. 9). Springer. https://doi.org/10.1007/978-3-319-13003-3_3

DIANA. (2008). DIANAFEA Manual - D.4 Historical Notes. Retrieved May 31, 2021, from https://dianafea.com/manuals/d93/GetStart/node104.html

Michael H. Ramage. (2004). Building a Catalan vault. Retrieved February 25, 2021, from http: //web.mit.edu/cron/Backup/project/guastavino/www/resources/writings/ramage{_}text.pdfhttp: //web.mit.edu/cron/Backup/project/guastavino/www/resources/resources{_}writings.htm{\#}

Block Research Group. (n.d.). Prof. Dr. Philippe Block. Retrieved October 21, 2021, from https://block.arch.ethz.ch/brg/people/philippe-block

Davis, L., Rippmann, M., Pawlofsky, T., & Block, P. (2012). Innovative funicular tile vaulting: A prototype vault in Switzerland. *The Structural Engineer*, *90*(11), 46–56. www.thestructuralengineer.orghttps://block.arch.ethz.ch/brg/files/2012{_}J-IStructE{_}Davis-Rippmann-Pawlofsky-Block.pdf

Block, P. (2009). *Thrust Network Analysis Exploring Three-dimensional Equilibrium Submitted to the Department of Architecture in Partial Fulfillment of the Requirements for the Degree of* (Doctoral dissertation). Massachusetts Institute of Technology. https://block.arch.ethz.ch/brg/content/publication/thrust-network-analysis-exploring-three-dimensional-equilibrium

Philippe Block, & Matthias Rippmann. (2013). The Catalan Vault – A Historical Structural Principle with a Bright Future. *DETAIL: Review of Architecture*, *53*(5), 528–536. https://block.arch.ethz.ch/brg/files/2013-detail-block-rippmann-katalanisches-gewoelbe{_}1396861318.pdf

López López, D., Domènech Rodríguez, M., & Palumbo Fernández, M. (2014). "Brick-topia", the thin-tile vaulted pavilion. *Case Studies in Structural Engineering*, *2*, 33–40. https://doi.org/10.1016/j.csse.2014.09.001

Blanco, Z. G. (2014). The Thin Tile Vaulting Manual. www.unhabitat.orghttps://www.hamk.fi/wp-content/uploads/2018/09/Thin-tile-Vaulting-Manual-Draft{_}ZB2018.pdf

Lourenco, P. (1996). *Computational strategies for masonry structures* (Doctoral dissertation). Delft University of Technology. https://repository.tudelft.nl/islandora/object/uuid{\%}3A4f5a2c6c-d5b7-4043-9d06-8c0b7b9f1f6f

D'Altri, A. M., Sarhosis, V., Milani, G., Rots, J., Cattari, S., Lagomarsino, S., Sacco, E., Tralli, A., Castellazzi, G., & de Miranda, S. (2020). Modeling Strategies for the Computational Analysis of Unreinforced Masonry Structures: Review and Classification. *Archives of Computational Methods in Engineering*, *27*(4), 1153–1185. https://doi.org/10.1007/s11831-019-09351-x

Zhang, Y., Yang, J., & Cao, X. (2020). Effects of several retarders on setting time and strength of building gypsum. *Construction and Building Materials*, *240*, 117927. https://doi.org/10.1016/J.CONBUILDMAT.2019.117927

Lewry, A. J., & Williamson, J. (1994b). The setting of gypsum plaster - Part II The development of microstructure and strength. *Journal of Materials Science*, *29*(21), 5524–5528. https://doi.org/10.1007/BF00349943

Hashempour, M., Samani, A. A., & Heidari, A. (2021). Essential improvements in gypsum mortar characteristics. *International Journal of Engineering, Transactions B: Applications*, *34*(2), 319–325. https://doi.org/10.5829/IJE.2021.34.02B.03

Cambridge. (2020). VIDEO-ON-DEMAND | meaning in the Cambridge English Dictionary. Retrieved August 10, 2021, from https://dictionary.cambridge.org/dictionary/english/video-on-demand?fbclid=IwAR1taABjqQt6wSdu12pzBGCQDVznFPdCjLwukvRVm48pH244Lp1QYF0GhIo

NBVG. III Gebrand gips en Stukadoorsgips (1st ed.). In: *Alles over gips* (1st ed.). 1st ed. Nederlandse Branche Vereniging Gips, 2006. Chap. 3, pp. 17–33. www.stabu.nl

Lewry, A. J., & Williamson, J. (1994a). The setting of gypsum plaster - Part I The hydration of calcium sulphate hemihydrate. *Journal of Materials Science*, *29*(20), 5279–5284. https://doi.org/10.1007/BF01171536

Yu, Q., Brouwers, H., & de Korte, A. Gypsum hydration: a theoretical and experimental study (H. Fischer & K. Bode, Eds.). In: In *Proceedings 17th ibausil, international conference on building materials (internationale baustofftagung)* (H. Fischer & K. Bode, Eds.). Ed. by Fischer, H., & Bode, K. Weimar: Bauhaus-Universitat Weimar, 2009, 6. ISBN: 978-3-00-027265-3. https://josbrouwers.bwk.tue.nl/publications/Conference51.pdf

Karni, J., & Karni, E. (1995). Gypsum in construction: origin and properties. *Materials and Structures*, *28*(2), 92–100. https://doi.org/10.1007/BF02473176

SIKA. (2021). Dry Mortar - Sika Performance Additive Technologies. https://www.sika.com/content/dam/dms/corporate/n/glo-dry-mortar-additives.pdfhttps://www.sika.com/en/construction/gypsum-dry-mortar.html

Pham, H. Q., & Marks, M. J. (2005). Epoxy Resins. *Ullmann's Encyclopedia of Industrial Chemistry*. https://doi.org/10.1002/14356007.A09_547.PUB2

Huntsman. (2020). THE ADHESIVES YOU NEED NO MATTER YOUR INDUSTRY: Araldite 2000. https://huntsman-pimcore.equisolve-dev.com/Documents/Araldite2000corerangebrochure{_}EU.pdf

SIKA. (2012). Sikadur ®-51. *SIKA*. www.sika.nlhttps://www.viba.nl/media/files/tds/tds0000566.pdf

Huntsman. (2017). Araldite ® 2015-1 Two component epoxy paste adhesive Key properties. *Huntsman*. www.aralditeadhesives.com.https://www.viba.nl/media/files/tds/tds0001457.pdf

Eurocodes. (n.d.). Retrieved August 19, 2021, from https://www.nen.nl/en/bouw/constructieve-veiligheid/eurocodes

Eurocode 6 - Design of masonry structures - Part 1-1: General rules for reinforced and unreinforced masonry structures. (2013). https://connect.nen.nl/standard/openpdf/?artfile=557416{\&}RNR=185662{\&}token=d81f5e3c-e553-47d7-ae6d-0c90a05ce027{\&}type=pdf{\#}pagemode=bookmarks

Epoxy Products Ltd. (2020). Outstanding Adhesive ▲ Rapid Strength Development ▲ Twin Pack-50/50 Mix-No Waste ▲ Excellent Chemical Resistance THE RANGE Manufacturers of High Performance Floor Coatings and Re-Surfacing Screeds. *Epoxy Products Ltd*. https://www.epoxyproducts.co.uk/BrickAdhesive.pdf

Mojsilović, N. (2011). Tensile strength of clay blocks: An experimental study. *Construction and Building Materials*, *25*(11), 4156–4164. https://doi.org/10.1016/J.CONBUILDMAT.2011.04.052

Chang, L. Z., Messali, F., & Esposito, R. (2020). Capacity of unreinforced masonry walls in out-of-plane two-way bending: A review of analytical formulations. *Structures*, *28*, 2431–2447. https://doi.org/10.1016/J.ISTRUC.2020.10.060

Estrella, G. P. (2017). Robotics in Architecture Potential applications and current limitations Robotics in Architecture Potential applications and current limitations. https://www.academia.edu/34623222/Robotics{_}in{_}Architecture{_}Potential{_}applications{_}and{_}current{_}limitations{_}Robotics{_}in{_}Architecture{_}Potential{_}applications{_}and{_}current{_}limitations

Bidgoli, A. (2015). *Towards an integrated design making approach in architectural robotics* (Doctoral dissertation). Pennsylvania State University. https://etda.libraries.psu.edu/catalog/27238

ISO/ASTM. (2017). ISO/ASTM 52900, https://connect.nen.nl/Family/Detail/62069?compId=10037{\&}collectionId=0

RoboDK. (2021). RoboDK. Retrieved August 10, 2021, from https://robodk.com/blog/robot-machining-rhinocam-robodk/

RoboFold. (2013). RoboFold. Retrieved August 10, 2021, from https://robofold.com/make/software

RoboDK. (n.d.). Robot Library | RoboDK. Retrieved August 10, 2021, from https://robodk.com/library

Madsen, A. J. (2019). *The SAM100: Analyzing Labor Productivity* (tech. rep.). California Polytechnic State University. https://digitalcommons.calpoly.edu/cmsp/243/

Bogue, R. (2018). What are the prospects for robots in the construction industry? *Industrial Robot: An International Journal*, *45*(1), 1–6. https://doi.org/10.1108/IR-11-2017-0194

Bloss, R. (2014b). Robots have come to architecture to model, construct, fabricate and offer new approaches to create innovative designs, elements and structures. *Industrial Robot*, *41*(5), 403–407. https://doi.org/10.1108/IR-06-2014-0359

Van Sante, M. (2020). Industrialisatie in de bouw. Retrieved August 6, 2021, from https://www.ing.nl/media/EBZ{_}ING-Industrialisactie-in-de-bouw{_}mei-2020{_}tcm162-194858.pdf

Technical reference manual - RAPID Overview (tech. rep.). (2019). ABB Robotics. https://abb.sluzba.cz/Pages/Public/OmniCoreRoboticsDocumentationRW7/Controllers/RobotWare/RAPID/en/3HAC065040-001.pdf

Motoman. (2019). SPECIFICATIONS: MH50 II-20. Retrieved August 21, 2021, from https://www.motoman.com/getmedia/e24fd3d6-107b-4825-b6f8-50201e3d0c9d/MH50II-20.pdf.aspx

Fanuc. (2019). FANUC M-710iC/20L industriële robot. Retrieved August 21, 2021, from https://www.fanuc.eu/be/nl/robots/robot-filter-pagina/m-710-serie/m-710ic-20l

Fanuc. (2017). FANUC M-710iC/12L industriële robot. Retrieved August 21, 2021, from https://www.fanuc.eu/be/nl/robots/robot-filter-pagina/m-710-serie/m-710ic-12l

Kawasaki. (2020). RS015X Robot | R series | Small-Medium Payload | Kawasaki Robotics. Retrieved August 21, 2021, from https://robotics.kawasaki.com/en1/products/robots/small-medium-payloads/RS015X/index.html?language{_}id=4

Appendices

A

The Grasshopper model: from design to work procedure

This appendix describes the model made in grasshopper. The model is a combination of three models as described in the report. The first part consists of the design model, the second part is the engineering model, the third part is the robotic model. The input is shown on the left as purple, the models are shown as red, green and blue respectively, and the output is shown on the right in orange.

The input parameters can be found in table 11.1, with a few additional inputs for the transition between models and programs, like the file imports. The output consists of the visualisation of the vault's design, the information for each following model, the input for the structural analysis in Python and the poses and additional information for RoboDK.

The following images are present in this appendix to illustrate and show the Grasshopper model:

- Figure A.1: The entire canvas in the Grasshopper model, also showing the work space created in Grasshopper.
- Figure A.2: The canvas is represented as a process in a diagram.
- Figure A.3: A detail of the map projection cluster from chapter 4.
- Figure A.4: The entire canvas again, with enougb resolution for visible components.

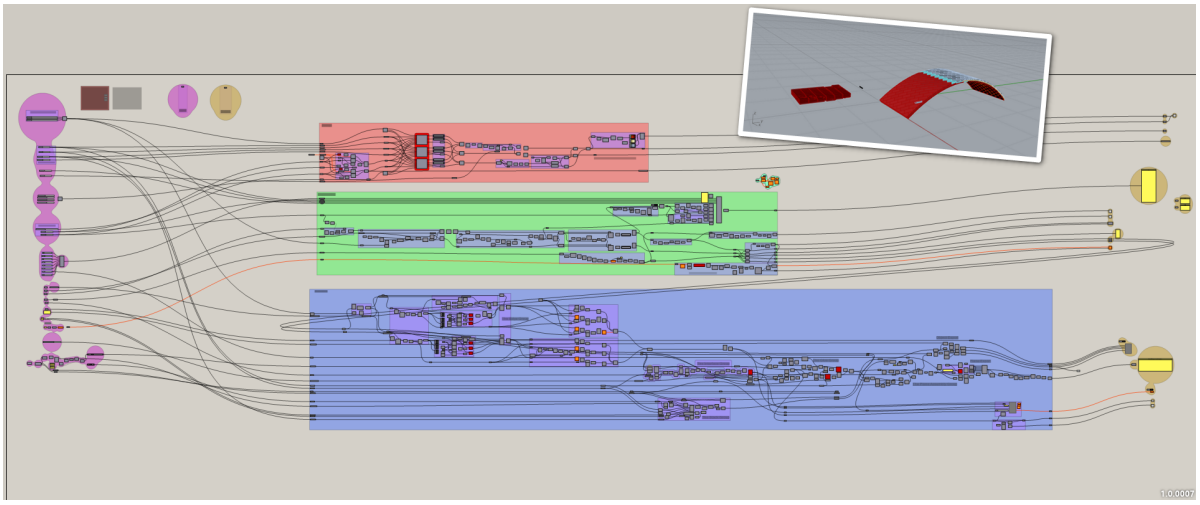


Figure A.1: The complete canvas in Grasshopper. On the left is the input, on the right is the output. The red box is the design model, the green box is the engineering and the blue box is the robotics.

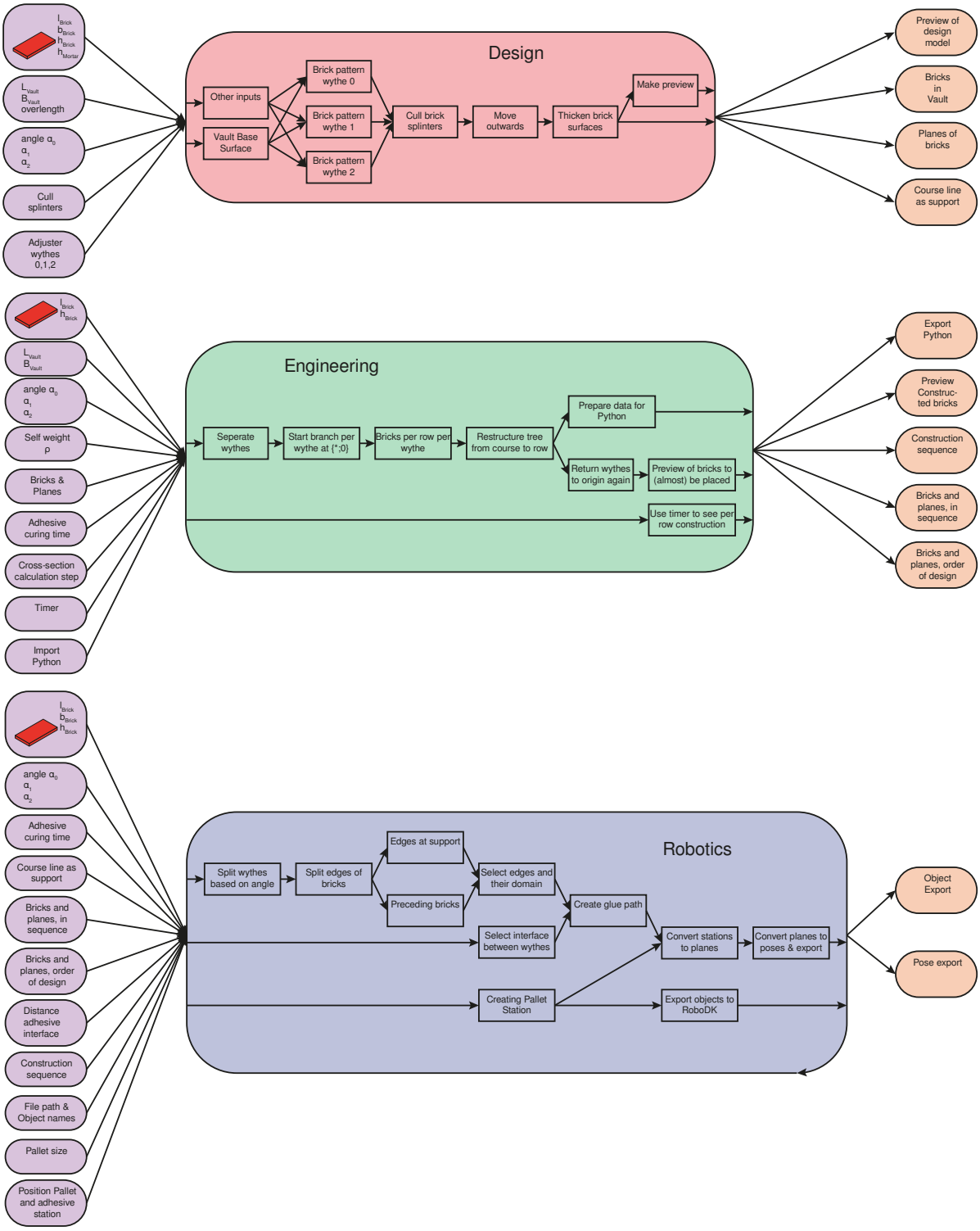
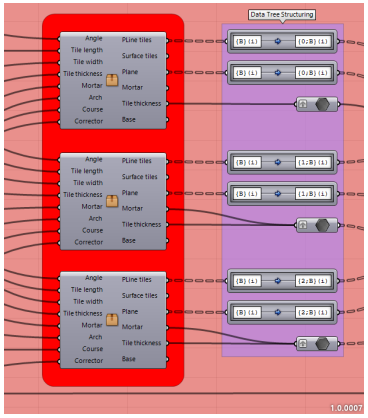
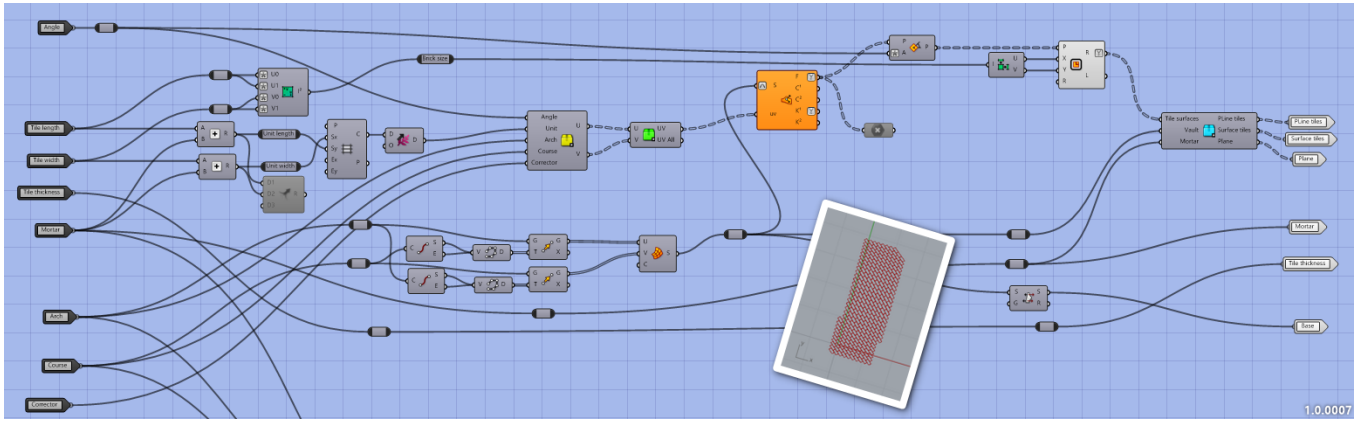


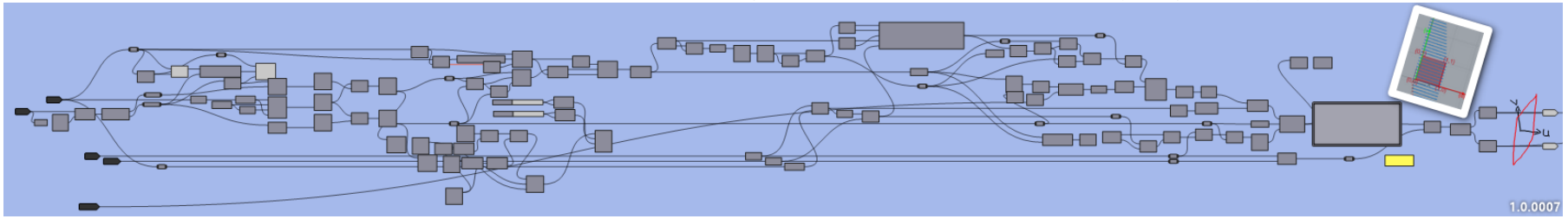
Figure A.2: The workflow of the Grasshopper model based on the groups created on the Grasshopper canvas



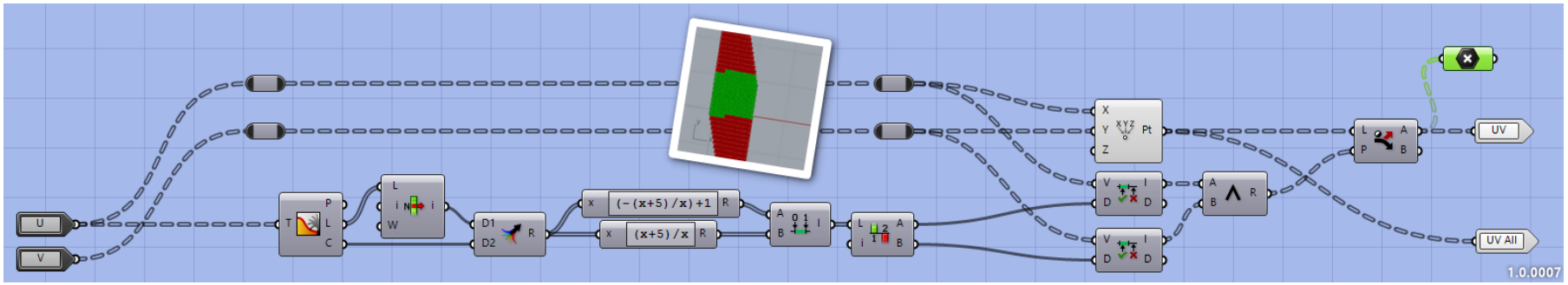
(a) The tiling cluster within the GH canvas;



(b) The tiling cluster exploded, with three more clusters visible in yellow, green and cyan;

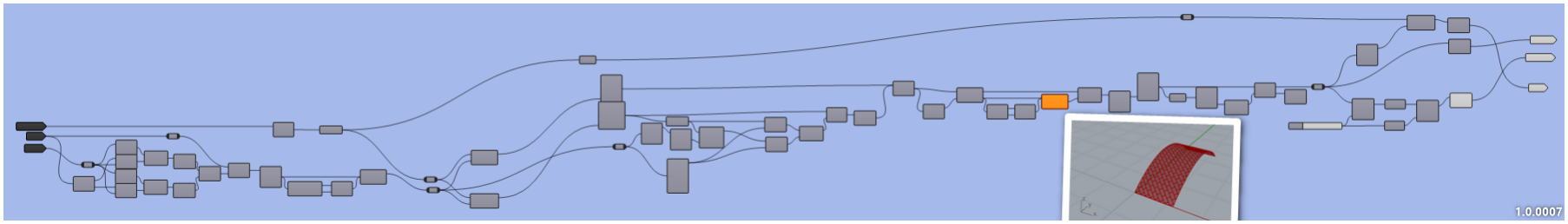


(c) Tiling the surface: the yellow cluster;



(d) Reducing points to compute: the green cluster;

Figure A.3: The tiling cluster exploded.



(e) Project points from unit surface to base vault surface: the cyan cluster;

Figure A.3: The tiling cluster exploded.

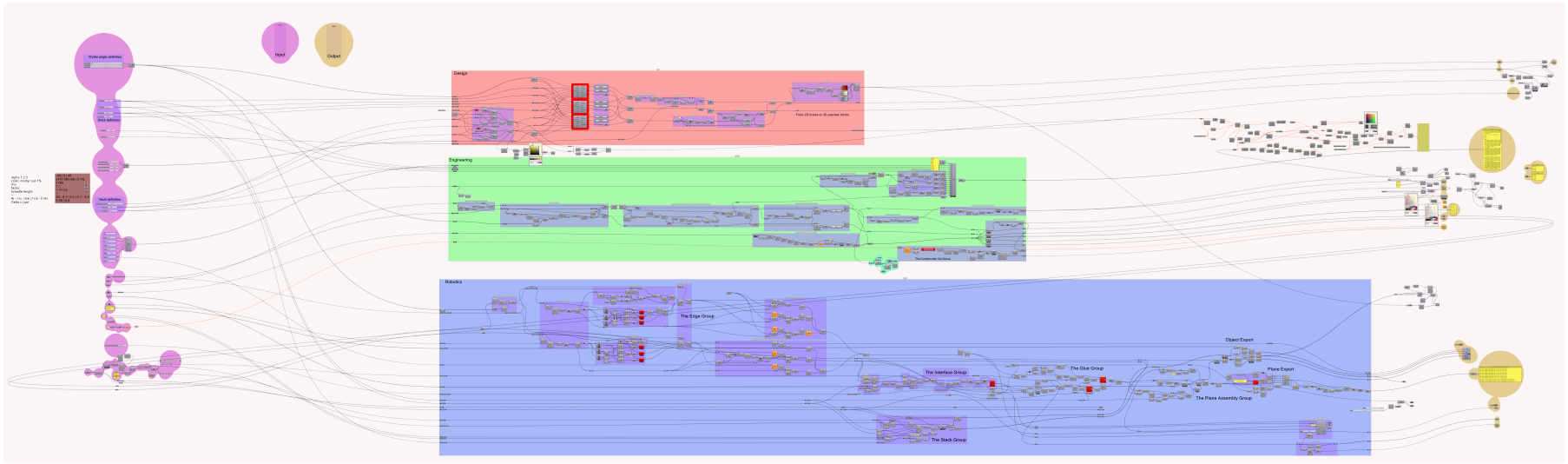


Figure A.4: A detailed and (digitally) zoomable figure of the canvas.

B

DIANA: modelling the monolithic property

This chapter provides the full DIANA model used for section 6.4.1

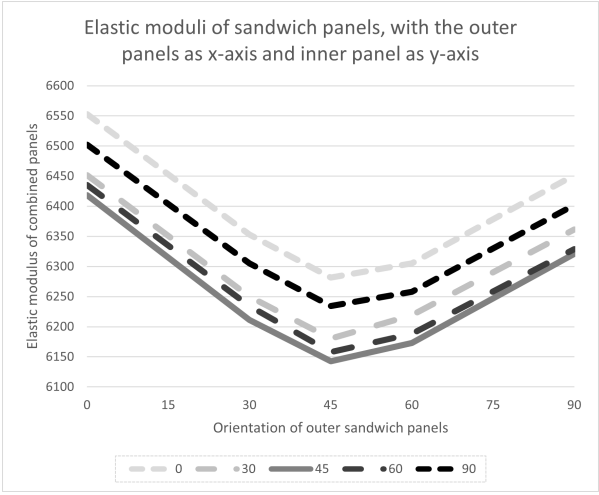


Figure B.1: The elastic modulus at different orientations in a three wythe sandwich panel found with DIANA.

Report 3 panels 1

Contents

Chapter 1

Project information

Diana project name	W:/student-homes/w/jwelles/Documents/Master_Thesis/Rhino Files/Untitled.dpf
Analysis aspects	[‘Structural’]
Model dimension	[‘Three dimensional’]
Default mesher type	HEXQUAD
Default mesher order	QUADRATIC

Diana version	Diana 10.4, Latest update: 2021-03-05 13:13:13
System	Windows NT 6.2 Build 9200
Model sizebox	10.0

Units

The following units are applied

Quantity	Unit	Symbol
Length	meter	m
Mass	kilogram	kg
Force	newton	N
Time	second	s
Temperature	kelvin	K
Angle	radian	rad

Directions

The following directions are defined:

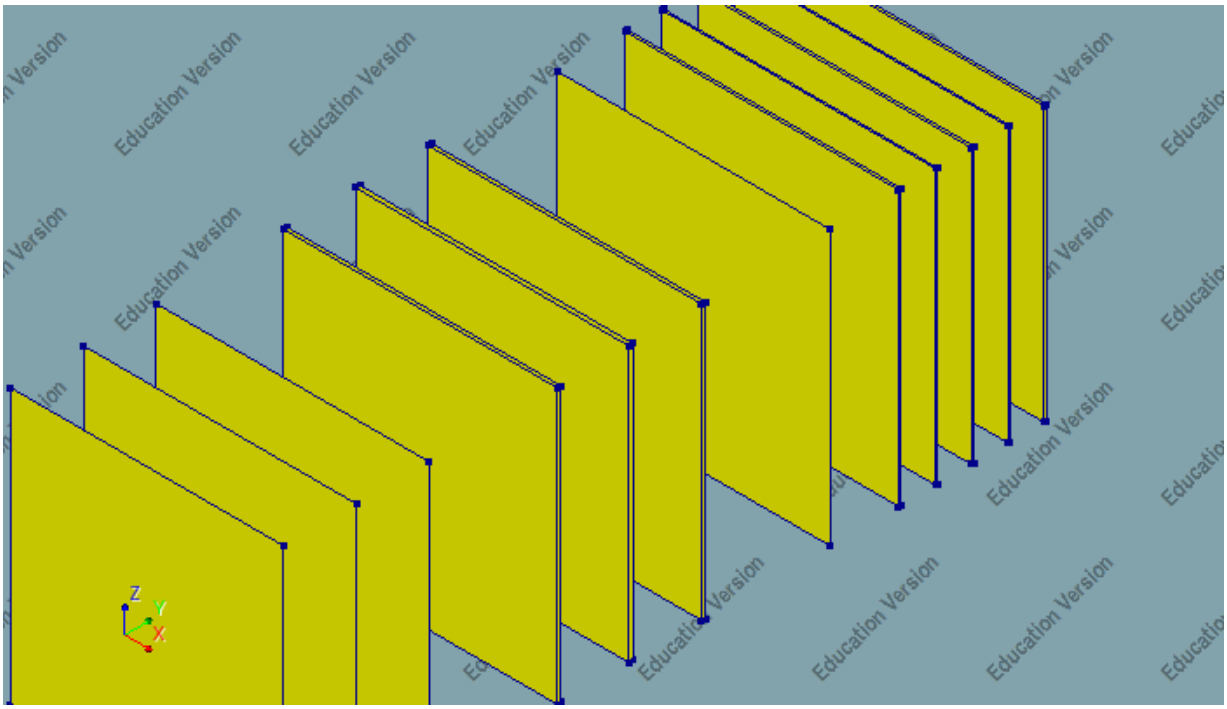
Name	X	Y	Z
X	1	0	0
Y	0	1	0
Z	0	0	1

Definitions

Name	Value
Acceleration of gravity	-9.81 m/s2
Fluid density	1000 kg/m3
Reference point for total head	0 0 0

Rayleigh damping coefficients	a: 0 b: 0
Design safety factor concrete compressive strength	1
Design safety factor concrete uniax. tensile strength	1
Design safety factor concrete stiffness	1
Design safety factor steel yield stress	1
Design safety factor steel stiffness	1
Direction of gravity	Z

Chapter 2



The model consists of the following shapes, reinforcements, piles and interfaces:

Shapes

Name	Set	Element Class	Material	Geometry	Seeding method	Element size [m]	Division
Wythe 1	Shapes	CURSHL	Brick 1	Masonry wall x=(1;0;0)	Divisions	0	19
Wythe 2	Shapes	CURSHL	Brick 1	Masonry wall x=(1;0;var)	Divisions	0	19
Wythe 3	Shapes	CURSHL	Brick 1	Masonry wall x=(1;0;0)	Divisions	0	19
Sheet 1	Shapes 1	CURSHL	Brick	Masonry wall x= 1	Divisions	0	19
Block 1	Shapes 2	STRSOL	Brick no gap	Wall no gap		0	0
Block 2	Shapes 2	STRSOL	Brick no gap	Wall no gap		0	0
Block 3	Shapes 2	STRSOL	Brick no gap	Wall no gap	Divisions	0	9
Block 4	Shapes 3	STRSOL	Brick no gap	Wall no gap		0	0
Block 5	Shapes 3	STRSOL	Bed no gap	Wall no gap		0	0
Block 6	Shapes 3	STRSOL	Brick no gap	Wall no gap		0	0
Block 7	Shapes 3	STRSOL	Bed no gap	Wall no gap		0	0
Block 8	Shapes 3	STRSOL	Brick no gap	Wall no gap		0	0

Interfaces

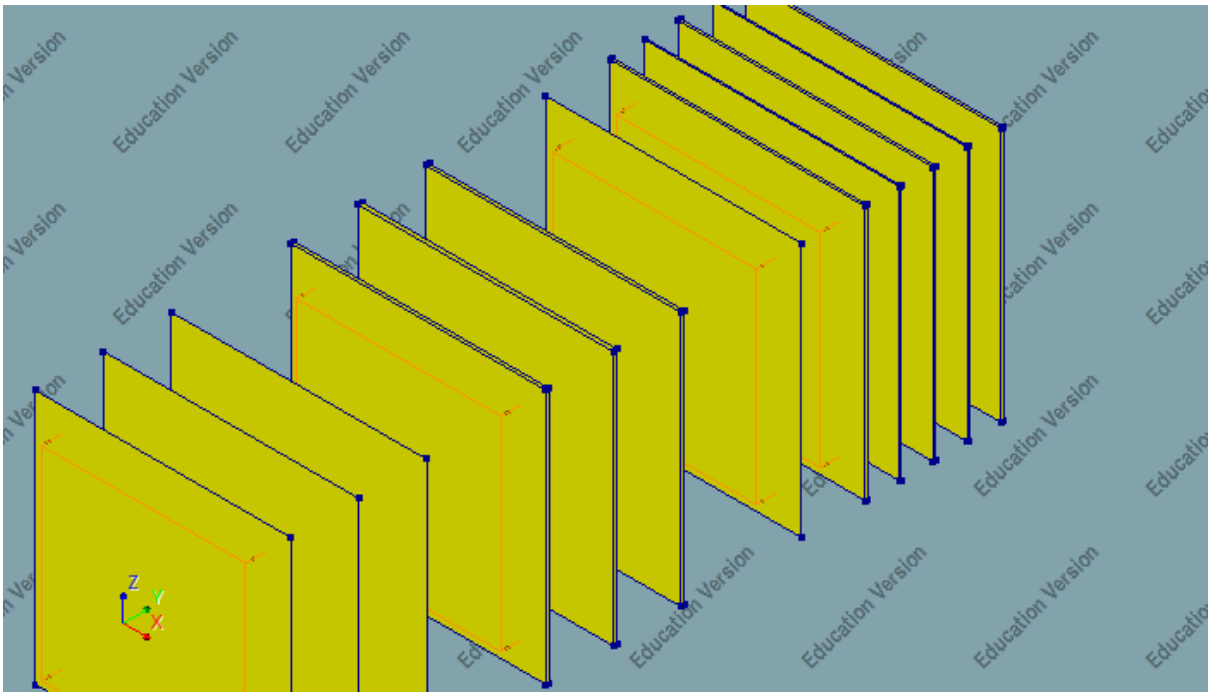
Name	Interface Type	Element Class	Material
Wythe 1	Interface	STPLIF	Wythe bed
Wythe 2	Interface	STPLIF	Wythe bed
Wythe 3			
Block 1	Interface	STPLIF	Wythe bed
Block 2			
Block 2	Interface	STPLIF	Wythe bed
Block 3			

Dimensions

Axes	Minimum coordinate [m]	Maximum coordinate [m]
X	-1	1
Y	-2	2.6
Z	-1	1

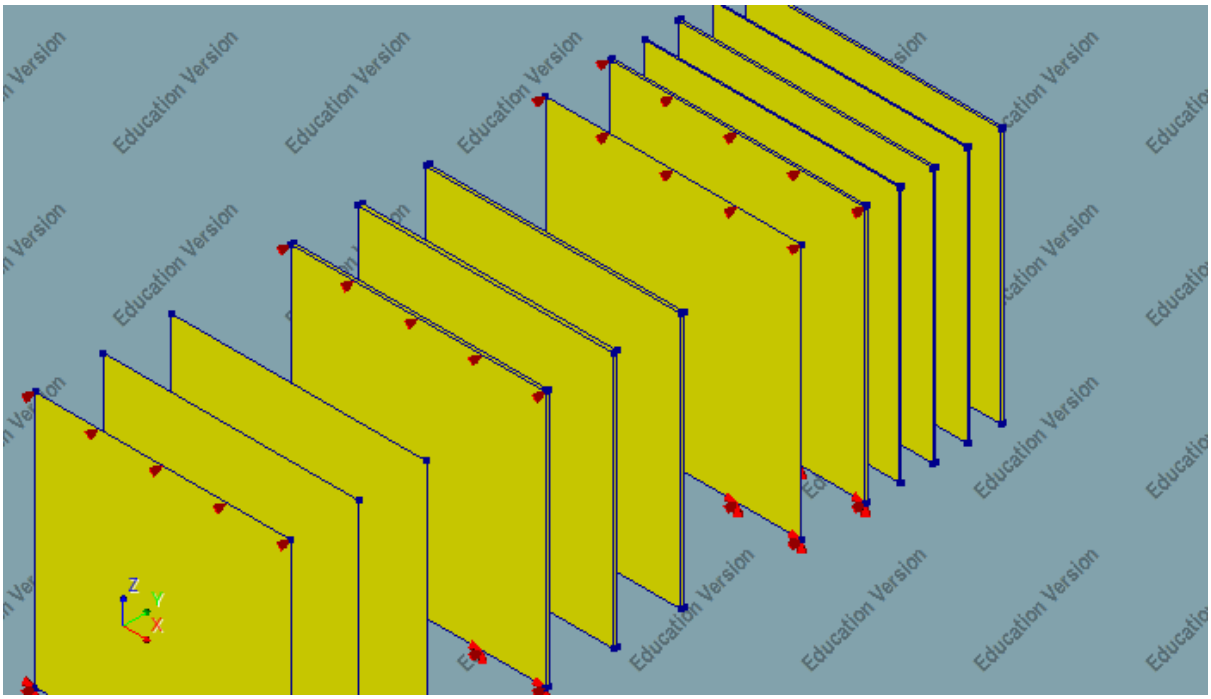
Geometry load ‘Bending’

Name	Target	Type	Direction	DOF	Value	Unit
Load Bending wythe 1	SURFAC	FORCE	NORMAL		1000	N/m2



Geometry support ‘Supports’

Name	Target	Translation	Rotation
Roller	LINE	Y	
Simple support	LINE	X,Y,Z	



Geometry: Masonry wall x=(1;0;0)

Name	Value
Geometry class	Sheets
Geometry model	Regular curved shell elements
Thickness	0.028 m
Element x axis	1 0 0

Geometry: Masonry wall x=(1;0;var)

Name	Value
Geometry class	Sheets
Geometry model	Regular curved shell elements
Thickness	0.028 m
Element x axis	1 0 0

Geometry: Wythe bed

Name	Value
Geometry class	Sheets
Geometry model	Structural surface interface elements
Element x axis	1 0 0

Geometry: Masonry wall x= 1

Name	Value
Geometry class	Sheets
Geometry model	Regular curved shell elements
Thickness	0.084 m
Element x axis	1 0 0

Material: Brick

Name	Value
Material class	Concrete and masonry
Material model	Linear elastic orthotropic
Color	grey
Element type	Curved shell
Young’s modulus	6.5e+09 6.5e+09 6.5e+09 N/m2
Poisson’s ratio	0.15 0.15 0.15
Shear modulus	2.82e+09 2.82e+09 2.78e+09 N/m2
Mass density	1800 kg/m3

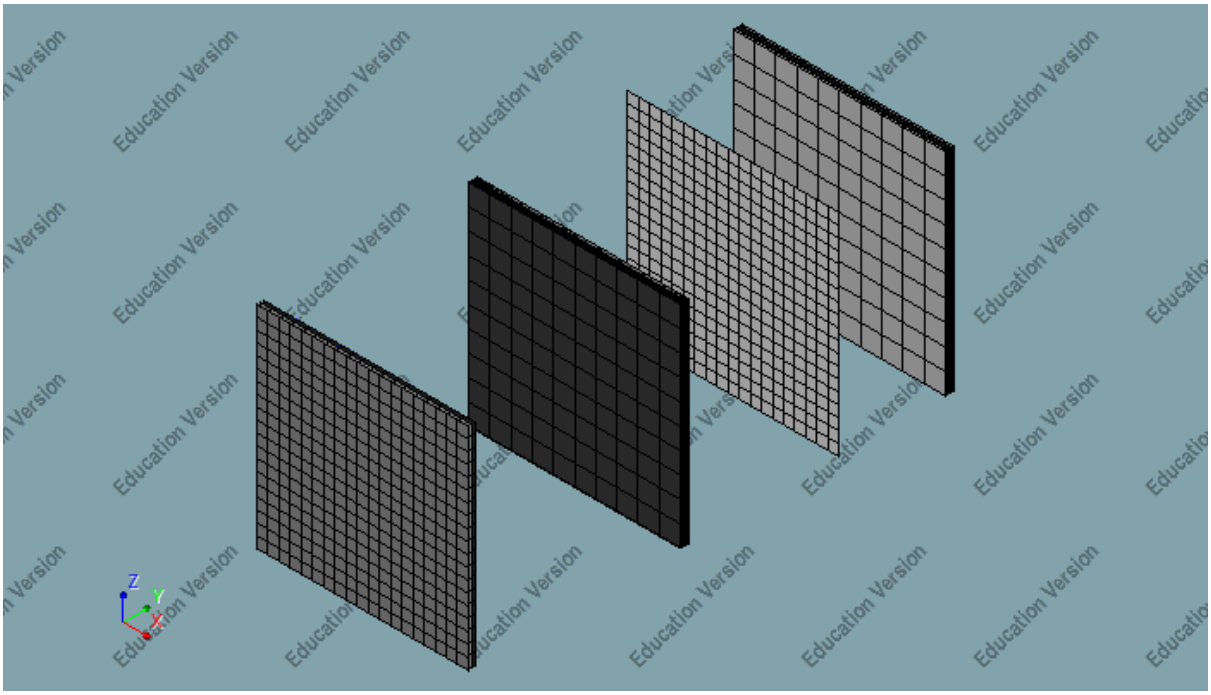
Material: Wythe bed

Name	Value
Material class	Interface elements
Material model	Linear elasticity
Color	silver
Type	3D surface interface
Normal stiffness modulus-z	9.17e+10 N/m3
Shear stiffness modulus-x	3.36e+10 N/m3
Shear stiffness modulus-y	3.36e+10 N/m3

Data: Element data 1

Name Value —— ——- —— ——-

Chapter 3



The Mesh consists of the following parts:

Mesh Sets

Name	# Elements	Material	geometry	Data
Wythe 1	361	Brick 1	Masonry wall x=(1;0;0)	
Wythe 2	361	Brick 1	Masonry wall x=(1;0;var)	
Wythe 3	361	Brick 1	Masonry wall x=(1;0;0)	
Sheet 1	361	Brick	Masonry wall x= 1	
Block 1	390	Brick no gap	Wall no gap	
Block 2	474	Brick no gap	Wall no gap	
Block 3	534	Brick no gap	Wall no gap	
Block 4	100	Brick no gap	Wall no gap	
Block 5	544	Bed no gap	Wall no gap	
Block 6	100	Brick no gap	Wall no gap	
Block 7	471	Bed no gap	Wall no gap	
Block 8	100	Brick no gap	Wall no gap	
Connection 1-2	361	Wythe bed	Wythe bed	
Connection 2-3	361	Wythe bed	Wythe bed	
Con12	100	Wythe bed	Wythe bed	
Con23	100	Wythe bed	Wythe bed	

Chapter 4

Analysis: Analysis1

Definition

```
Structural linear static
 Structural linear static
  Evaluate model
   Evaluate elements
    Average nodal normals
     Tolerance angle  = 0.349066
    Evaluate composed elements
   Assemble Elements
    Degree of freedom tolerance  = 1e-06
   Setup element stiffness matrices
   Setup load vectors
  Solve the system of equations
   Method  = Parallel Direct Sparse
   Convergence tolerance  = 1e-08
   Parallel Direct Sparse
    Factorization
  Output linear static analysis
   Output linear static analysis
    Device  = DIANA native
    Option  = Binary
    Seltyp  = USER
    Select
     Blknam  = OUTPUT
     Modsel  = Complete
     Loads
      User selection  = ALL
     Casety  = LOADS
    User
     Displacements
      Displacements
       Form  = TRANSL
       Oper  = GLOBAL
       Type  = TOTAL
       Total
      Displacements
       Form  = TRANSL
       Oper  = LOCAL
       Type  = TOTAL
       Total
     Strains
     Stresses
```

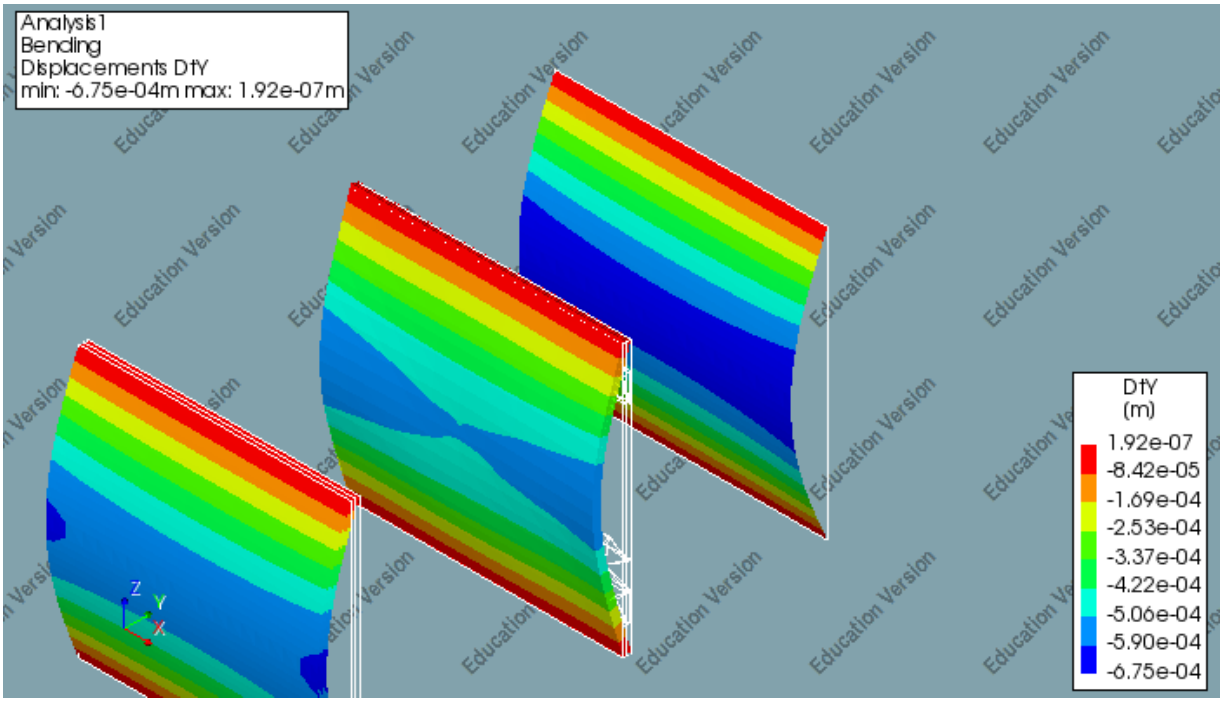
```
        Stresses
         Form  = CAUCHY
         Oper  = GLOBAL
         Type  = TOTAL
         Total
        Stresses
         Form  = CAUCHY
         Oper  = LOCAL
         Type  = TOTAL
         Total
        Stresses
         Form  = DISFOR
         Oper  = LOCAL
         Type  = TOTAL
         Total
        Stresses
         Form  = DISFOR
         Oper  = LOCAL
         Type  = ELEMEN
         Element contribution
       Forces
       Model parameters
       Element forces
```

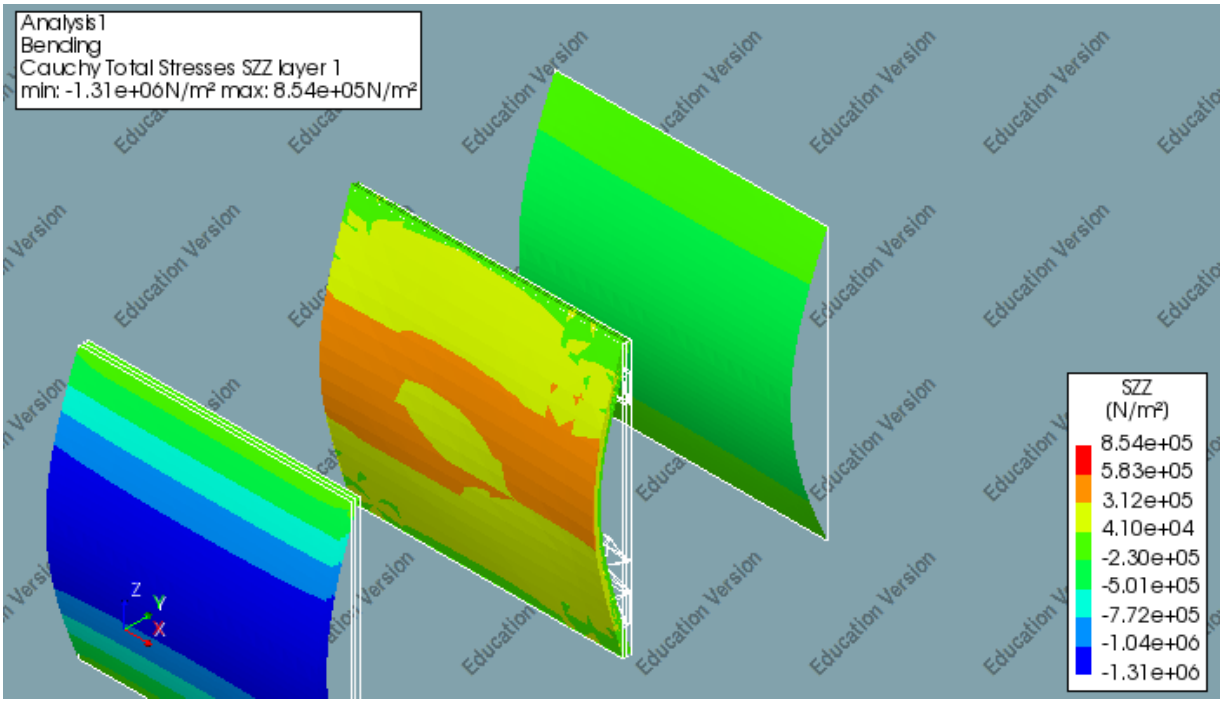
DCF Commands

```
*LINSTA LABEL="Structural linear static"
  MODEL EVALUA CHECK OFF
  SOLVE PARDIS
  BEGIN OUTPUT
    TEXT "Output linear static analysis"
    BINARY
    SELECT LOADS ALL /
    DISPLA TOTAL TRANSL GLOBAL
    DISPLA TOTAL TRANSL LOCAL
    STRESS TOTAL CAUCHY GLOBAL
    STRESS TOTAL CAUCHY LOCAL
    STRESS TOTAL DISFOR LOCAL
    STRESS ELEMEN DISFOR LOCAL
  END OUTPUT
*END
```

Phases

Chapter 5


Analysis1
Bending
Displacements DtY
min: -6.75e-04m max: 1.92e-07m
DtY
(m)
1.92e-07
-8.42e-05
-1.69e-04
-2.53e-04
-3.37e-04
-4.22e-04
-5.06e-04
-5.90e-04
-6.75e-04
Z
Y
X
Education Version


Analysis1
Bending
Cauchy Total Stresses SZZ layer 1
min: -1.31e+06N/m² max: 8.54e+05N/m²
SZZ
(N/m²)
8.54e+05
5.83e+05
3.12e+05
4.10e+04
-2.30e+05
-5.01e+05
-7.72e+05
-1.04e+06
-1.31e+06
Z
Y
X
Education Version

C

Brick experiment: testing the bond strength development

This chapter provides the experiment done on standard bricks from the hardware store and epoxy glue provided by the TU Delft. The epoxy glue is a Siko-Clearbond glue of 50 mL, with a 2K-Methyl methacrylaat. The following figures are present in this appendix:

- Figure C.1: Properties of the glue.
- Figure C.2: Weather of the days of experimentation.
- Figure C.3: The first brick experiment on the 24th of May.
- Figure C.4: The second brick experiment.
- Figure C.5: The third brick experiment.
- Figure C.6: The fourth brick experiment.
- Figure C.7: The fifth brick experiment.
- Figure C.8: The sixth brick experiment on the 28th of May, with cobblestone bricks.
- Figure C.9: The seventh brick experiment with hardware store bricks again.

2-component adhesive product name	Siko Clearbond
Type	Methyl methacrylate
Gap fill	2mm
Colour	Clear
Temperature resistance	-50C to +120C
Viscosity mPa.s	60000-90000
Fixture time	4 min
Final curing	24 h
Tensile strength Mpa	13-17
UV-resistance	Unknown
Shore D hardness	
Comments	It's high viscosity makes it easy to apply. Cures within 3min. Results in a clear adhesive layer.

Figure C.1: Siko-Clearbond properties

The brick did not fail with a hardening time of 7 minutes and 33 seconds (figure C.5), 9 minutes and 15 seconds (figure C.7), 7 minutes and 30 seconds (figure C.8) and 4 minutes and 23 second (figure C.9).

Weerstatistieken mei 2021 in Delft

Bekijk hieronder de weerstatistieken van mei 2021 in Delft. Lees onderaan de pagina hoe deze data is samengesteld. Je vindt hier de data per dag. Verder staat er onderaan nog een tabel met bijzondere dagen.

Let op: deze gegevens zijn niet gevalideerd. Eventuele meetfouten en/of storingen zijn niet uitgefilterd. Gebruik deze data daarom indicatief!

Datum	Tmax	Tmin	Neerslag	Max Windstoot	Opmerkingen
2021-05-31	23.8 °C	11.7 °C	0,0 mm	15.0 km/u	data tot 11:55u
2021-05-30	22.6 °C	9.4 °C	0,0 mm	14.0 km/u	data tot 11:55u
2021-05-29	18.3 °C	9.9 °C	0,0 mm	12.2 km/u	data tot 11:55u
2021-05-28	20.9 °C	7.1 °C	0,2 mm	10.0 km/u	data tot 11:55u
2021-05-27	13.2 °C	9.8 °C	0,8 mm	19.1 km/u	data tot 11:55u
2021-05-26	13.2 °C	8.7 °C	10,8 mm	22.0 km/u	data tot 11:55u
2021-05-25	13.1 °C	7.8 °C	2,4 mm	21.0 km/u	data tot 11:55u
2021-05-24	15.4 °C	10.1 °C	5,8 mm	22.0 km/u	data tot 11:55u

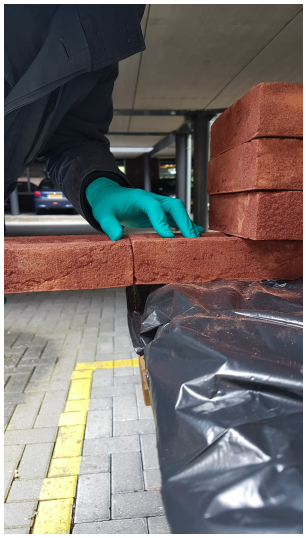
Figure C.2: The weather on the days of experimentation | https://www.hetweeractueel.nl/ weer/delft/historie/2021/05/.

(a) T=14:33:15, set up;

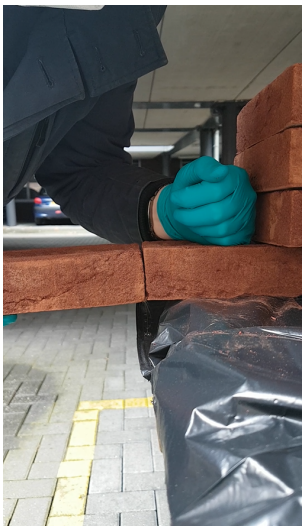
(b) T=2:11, applying adhesive;

(c) T=2:18, attaching;

(d) T=4:13, pressure for placement;



(e) T=6:24, pressure for placement, sideways;



(f) T=7:42, failure;

(g) T=7:51, check failure;

(h) T=8:51, ending the experiment;

(i) T=9:14, headers of bricks;

Figure C.3: First brick experiment.

(a) T=14:51:09, applying adhesive;

(b) T=1:06, pressing for placement;

(c) T=4:54, failure;

Figure C.4: Second brick experiment.

(a) T=14:58:45, applying adhesive;

(b) T=0:20, hardening;

(c) T=7:33, place horizontally;

(d) T=7:47, extra loading;

(e) T=7:57, failure;

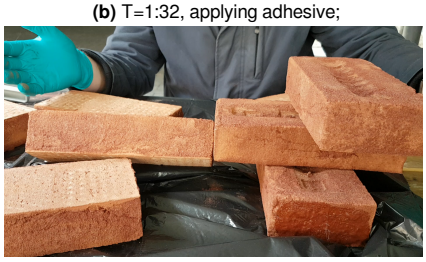
(f) T=8:29, headers;

Figure C.5: Third brick experiment.

(a) T=15:27:40, set up;

(b) T=1:32, applying adhesive;

(c) T=4:53, removing support;



(d) T=5:05, failure;

Figure C.6: Fourth brick experiment.

(a) T=15:35:11, applying adhesive;

(b) T=5:36, trying to remove support;

(c) T=5:39, put support back before failure;

(d) T=9:15, removing support;

(e) T=10:28, failure;

(f) T=10:30, failure, close up;

(g) T=12:19, headers;

(h) T=21:31, ripping off adhesive;

Figure C.7: Fifth brick experiment.

(a) T=11:17:18, applying adhesive;

(b) T=1:03, applying additional adhesive;

(c) T=7:30, removing support;

(d) T=8:28, adding weight, but more counterweight is needed;

(e) T=9:04, adding the weight again, with additional counterweight;

(f) T=9:17, failure;

(g) T=10:12, headers, close up;

(h) T=10:47, headers;

Figure C.8: Sixth brick experiment.

(a) T=11:31:02, sanding;

(b) T=5:15, applying adhesive;

(c) T=9:38, removing support;

(d) T=13:09, additional load;

(e) T=16:22, failure with two bricks;

(f) T=16:27, failure, overview;

(g) T=21:52, headers;

(h) T=22:37, comparison with figure C.8;

Figure C.9: Seventh brick experiment.

D

Phased Structural Analysis in Python

The phased structural anlysis has been done with an import file from Grasshopper, resulting in an export file to Grasshopper. The workflow is given. In the script afterwards not only the calculation is included, but also the making of the visualisations.

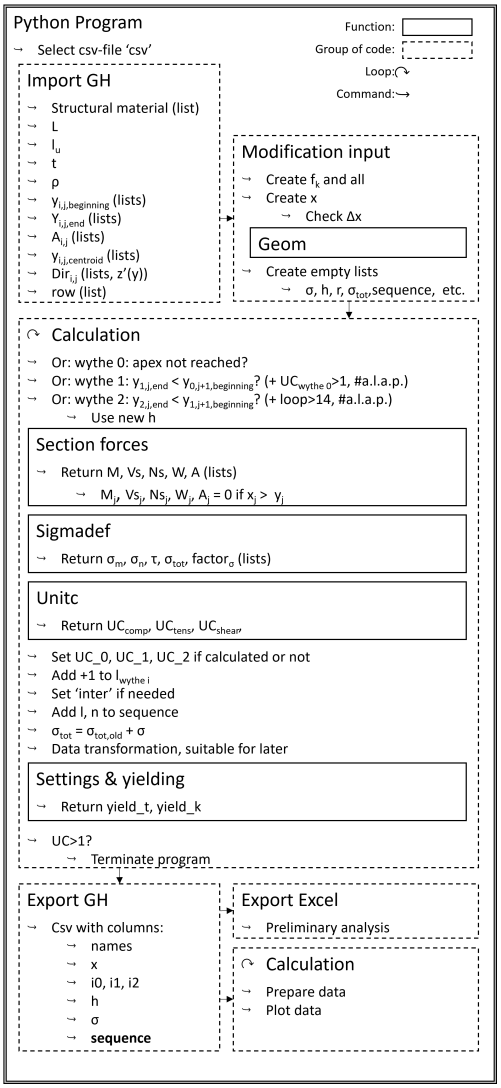


Figure D.1: Python workflow of the structural analysis.

In [1]:
```
__author__ = "Joris"
__version__ = "2021.08.26"
#
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import matplotlib.animation as ani
from matplotlib.animation import FuncAnimation as Fani
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:45% !important; }</style>"))
# %config InlineBackend.figure_formats = ['retina']
mpl.rcParams['figure.dpi'] = 600
print(mpl.rcParams['figure.dpi'])
import csv
print("INPUT:")
```

```
<IPython.core.display.HTML object>


600.0
INPUT:
```

In [2]:
```
#
GH = pd.read_csv(
    "C:/Users/Joris/Documents/Master Thesis quick access/GH_to_Python.csv",
    error_bad_lines=False)
GH = pd.read_csv(
    "C:/Users/Joris/Documents/Master Thesis quick access/GH_to_Python.csv",
    header=None, sep='\n')
GH = GH[0].str.split(';', expand=True)

GH = GH.values.tolist()
GH = [[ele for ele in GH[ind] if str(ele) != 'nan'] for ind,ele in enumerate(GH)]
GH = [[ele for ele in GH[ind] if str(ele) != 'None'] for ind,ele in enumerate(GH)]

GH_top = GH[0]
GH = GH[1:]

for i in range(len(GH_top)):
    if len(GH[i])>1:
        exec(GH_top[i] + " = [float(ele) for ele in GH[i]]")
    else:
        exec(GH_top[i] + " = float(GH[i][0])")
    exec("print(" + "GH_top[i])\nprint(" + GH_top[i] + ")\nprint()")

print("INPUT^")
print("FUNCTIONS:")
```

```
sm
[20.0, 0.1, 0.3, 0.1, 2.0, 0.04, 0.5, 40.0]

L
3.6

lu
0.21

t
0.02

rho
1750.0

y0b
[0.001942, 0.004363, 0.12204, 0.24266, 0.366599, 0.494935, 0.626936, 0.762191, 0.900692, 1.042161, 1.186268, 1.332786...

y1b
[-0.013907, -0.001186, 0.117076, 0.239484, 0.364786, 0.494101, 0.627385, 0.764148, 0.90285, 1.045686, 1.191479, 1.339...

y2b
[-0.029768, 0.017589, 0.190165, 0.371265, 0.559851, 0.756109, 0.959366, 1.169021, 1.383607, 1.601516, 1.821049, 2.040...

y0e
[0.067261, 0.185694, 0.307957, 0.433913, 0.564524, 0.698096, 0.834715, 0.974734, 1.117627, 1.262972, 1.410713, 1.5597...

y1e
[0.061141, 0.180306, 0.304216, 0.430439, 0.562536, 0.698043, 0.836294, 0.977588, 1.122094, 1.268772, 1.418042, 1.5688...

y2e
[0.096149, 0.272971, 0.457876, 0.649016, 0.848577, 1.055176, 1.266654, 1.482252, 1.700828, 1.92046, 2.139227, 2.35532...
```

```
A0
[0.040683, 0.139165, 0.139165, 0.139165, 0.139165, 0.139165, 0.139165, 0.139165, 0.139165, 0.139165, 0.139165, 0.1391...

A1
[0.052435, 0.139165, 0.139165, 0.139165, 0.139165, 0.139167, 0.139172, 0.139178, 0.139183, 0.139186, 0.139184, 0.1391...

A2
[0.103116, 0.1995, 0.1995, 0.1995, 0.1995, 0.1995, 0.1995, 0.1995, 0.1995, 0.1995, 0.1995, 0.1995, 0.1995, 0.1995, 0.1...

y0
[0.021823, 0.095505, 0.215188, 0.338746, 0.465814, 0.596443, 0.730637, 0.868174, 1.00878, 1.152152, 1.297881, 1.445516...

y1
[0.008651, 0.087761, 0.208485, 0.332874, 0.461473, 0.593586, 0.729094, 0.868036, 1.010159, 1.154967, 1.302475, 1.45177...

y2
[0.012579, 0.144002, 0.323122, 0.509154, 0.703541, 0.904715, 1.112227, 1.325243, 1.542039, 1.76096, 1.980261, 2.198153...

Dir0
[0.817056, 0.778442, 0.726475, 0.666186, 0.589992, 0.5287, 0.472582, 0.411248, 0.349787, 0.288724, 0.220057, 0.159523,...

Dir1
[0.81852, 0.776537, 0.713992, 0.669917, 0.591574, 0.52023, 0.460946, 0.401482, 0.339271, 0.282871, 0.214852, 0.151768,...

Dir2
[0.817891, 0.754174, 0.646288, 0.568826, 0.469041, 0.379546, 0.29128, 0.198586, 0.10745, 0.017397, -0.073506, -0.16695...

row
[8.0, 8.0, 11.0]

INPUT^
FUNCTIONS:
```

In [3]:
```python
#
def lim(y,l,i,j=0):
    try:
        return y[l[i]+j]
    except IndexError:
        return y[0]
#return:    either y[l[i]+j] or y[0]

def ll(l,i):
    try:
        return l[i]
    except IndexError:
        return 0
#return:    either l[i] or 0

def geom(x,yi, diri):
    ii = []; dsdxi = []; dzdxi = []
    ii.append( [np.argmax(yi > ele) +(len(yi))*(yi[-1] <  ele) for ele in x] )
    ii = ii[0]
    dzdxi.append([-lim(diri,ii,ele[0]) for ele in enumerate(x)])
    dzdxi = dzdxi[0]
    dsdxi.append([(1+dzdxi[ele[0]]**2)**0.5 for ele in enumerate(x)])
    dsdxi = dsdxi[0]
    ia = ii
    if max([ele<0 for ele in yi]):
        ia.extend( [ele for ele, a in enumerate([not(ele>0) for ele in yi]) if a] )
    if not ( all( x in ia for x in range(len(yi)) ) ):
        print('not every brick is represented in linspace x')
        print('x:')
        print(x)
        print(ia)
        print(set(ia).difference(range(len(yi))))
        exit()
    return ii,dsdxi,dzdxi
#return:    ii,dsdxi,dzdxi

def zero(i):
    zeros = np.zeros((i,len(x) ))
    return zeros
#return:    zeros

def new(l,h_old):
    n = min( len([l0,l1,l2][l]) , len([y0,y1,y2][l])-1 )
    ii = [i0,i1,i2][l]
    h = np.array([ ele+t if ii[ind]==n else ele for ind,ele in enumerate(h_old) ])
    return(h,n)

def section_forces(n,yi,Ai,dsdx,dzdx,h):
    weight = rho*9.81*t #N/mš
    y=yi[n] #m
```

```
    A=Ai[n] #mš/mž (Sum(l*b)/L)

    Vx = np.piecewise(x, [x<=y,x>y], [weight*A,0])
    M = weight*(y-x)*A
    M = [0 if ele < 0 else ele for ele in M]

    Vs = Vx/dsdx
    Ns = Vx*dzdx/dsdx

    W = 1/6 * h**2
    A = h
    return M,Vs,Ns,W,A
#return:    M,Vs,Ns,W,A

def sigmadef():
    dist = [[
        np.linspace( -1,1,int(h[j]/t+1) )[i]
        if int(h[j]/t)>=i
        else 0
        for j in range(len(x))]
        for i in range(lmax+1)]
    sigma_m = np.array(
        [[
        M[j]/W[j] *10**(-6) * dist[i][j]
        if h[j] != 0
        else 0
        if h[j]/t>=i
        else 0
        for j in range(len(x))]
        for i in range(lmax+1)] )
    sigma_n = np.array(
        [[
        Ns[j]/A[j] *10**(-6)
        * (h[j]/t>=i)
        if h[j] != 0
        else 0
        for j in range(len(x)) ]
        for i in range(lmax+1)] )
    tau =  np.array(
       [[
       Vs[j]*(6*dist[i][j]**2 - 3/2)/h[j] *10**(-6)
       if (h[j] != 0 and h[j]/t>=i)
       else 0
       for j in range(len(x)) ]
       for i in range(lmax+1)] )
    return dist,sigma_m,sigma_n, tau
#return:    dist, sigma_m, sigma_n, tau

def settings(l,n,setting):
    setting = [[
        min(i+[r0,r1,r2][l],1) #number of bricks
                #placed since this brick has been placed. similar to i+= row placed
        if i !=0
        else i
        for i in setting[j] ]
        for j in range(3)] #setting has shape of wythe Œ x
    setting[l] = [ [r0,r1,r2][l] if n==ele else setting[l][ind]
                  for ind,ele in enumerate([i0,i1,i2][l]) ]
    setting = np.array(setting)
    return setting
#return: setting

def yielding(setting,h_old,t,set):
    for j in range(4):
        for i in range(len(h_old)):
            if (int(j <= h_old[i]/t)) and (h_old[i] >0):
                sump = max( int(h_old[i]/t) -1,0)
                set[j][i] = setting[min(sump,j)][i]
    yield_t = [[fti+ele*(ftk-fti) for ele in set[j]] for j in range(4)]
    yield_k = [[fm+ele*(fk-fm) for ele in set[j]] for j in range(4)]
    yield_t,yield_k = np.array(yield_t),np.array(yield_k)
    return yield_t,yield_k
#return: yield_t,yield_k

def unitc(sigma,inter,strength):
    unitsc = np.array([sigma[0]/strength[0],
             sigma[1]/strength[0],
             (sigma[1]-inter[1])/strength[1],
             sigma[2]/strength[1],
             (sigma[2]-inter[2])/strength[2],
             sigma[3]/strength[3]])
    return unitsc
#return: uccomp,uctens,uctau
```

```
        def annot_max(x,y, ax=None,textp=[0.75,0.95]):
            xmax = x[np.argmax(y)] #xvalue
            ymax = y.max() #yvalye
            text= "@x:{:.2f}, uc={:.2f}".format(xmax, ymax) #text
            textpo = [textp[0],textp[1]] # textbox position
            if not ax:
                ax=plt.gca()
            bbox_props = dict(boxstyle="square,pad=0.3", fc="w", ec="k", lw=0.72) # textbox
            arrowprops=dict(arrowstyle="->",connectionstyle="angle,angleA=0,angleB=120")
                    # arrow
            kw = dict(xycoords='data',textcoords="axes fraction",
                      arrowprops=arrowprops, bbox=bbox_props, ha="left", va="top")
            ax.annotate(text, xy=(xmax, ymax), xytext=(textpo[0],textpo[1]), **kw)

        print("FUNCTIONS")
        print("SCRIPT")
        print("INPUT,OUTPUT AND PARAMETERS")
```

```
FUNCTIONS
SCRIPT
INPUT,OUTPUT AND PARAMETERS
```

```
In [4]: #        ##directe bewerking input
        L = L   #float
        [fb,fm,fvk,fti,ftf,deltax,pot,cure] = sm    #list of floats
        # fti,fm = .13,.13
        fk = -0.75*fb**0.7
        fm = -fm
        ftk = ftf
        fy = fk                             #####!!!!!
        fyt = ftk
        # ftk = 5
        fyv = fvk
        # y0 = [ele for ele in y0 if ele<L];    #list of floats
        # y1 = [ele for ele in y1 if ele<y0[-1]-lu/4]
        # y2 = [ele for ele in y2 if ele<y1[-1]-lu/4]
        #t & rho is also one float number   A0,A1,A2,row are lists of floats

                ##output
        seq=["layer;brick"]

                ##overige parameters

        l0=[];l1=[];l2=[]; lmax=3

        ymin = min(
        min([j-i for i, j in zip(y0[:-1], y0[1:])]),
        min([j-i for i, j in zip(y1[:-1], y1[1:])]),
        min([j-i for i, j in zip(y2[:-1], y2[1:])]))
        CHECK = (deltax < ymin)
        if CHECK==False:
            print("Reduce deltax!")
            print(ymin)

        x = np.arange(0,max(y0[-1],y1[-1],y2[-1]),deltax)
        i0,dsdx0,dzdx0 = geom(x,y0,Dir0)    #list of floats
        i1,dsdx1,dzdx1 = geom(x,y1,Dir1)
        i2,dsdx2,dzdx2 = geom(x,y2,Dir2)

        sigma_M = zero(4); sigma_N = zero(4); Tau = zero(4)
        sigma_m = zero(4); sigma_n = zero(4); tau = zero(4); dist = zero(4)
        h_old = np.zeros(len(x))

        setting = zero(3); set = zero(4)
        [r0,r1,r2] = row #number of bricks per row
        r0,r1,r2 = r0*pot/cure,r1*pot/cure,r2*pot/cure
        r = zero(4)
        yield_t,yield_k = yielding(setting,h_old,t,set)
        inter = zero(4)

        UCseries = []
        UCseries_ = []

        Distr = []
        Sigres = []
        Cstrres = []
        Tstrres = []
        Cucres = []
        Tucres = []
        ut=[]

        Sigd0 = [[],[]]
        Sigd9 = [[],[]]
```

```
        Sigd18 = [[],[]]
        Yied0 = [[],[]]
        Yied9 = [[],[]]
        Yied18 = [[],[]]
        UCd = [[],[]]
        apexl0=False
        print("INPUT,OUTPUT AND PARAMETERS^")
        print("FOR LOOP:")
```

```
INPUT,OUTPUT AND PARAMETERS^
FOR LOOP:
```

```
In [5]: #
        for runloop in range(len(y0)+len(y1)+len(y2)):
        # Layer 0 (1st) ########################################
            print('layer 0:',apexl0==False) # check if y0 is over the apex
            if apexl0==False:
        #     if lim(y0,l0,-1,1)<=(L+2*lu)/2 or len(l0)<1:
              l=0
              h,n = new(l,h_old)

              M,Vs,Ns,W,A = section_forces(n,y0,A0,dsdx0,dzdx0,h)
              dist, sigma_m, sigma_n, tau = sigmadef()

              sigma = sigma_M + sigma_m + sigma_N + sigma_n

              ttau = Tau + tau

              uccomp = unitc(sigma,inter,yield_k)
              uctens = unitc(sigma,inter,yield_t)
              uctau = ttau/fvk
            #   uccomp = (sigma-inter)/yield_k
              uccomp[uccomp < 0],uctens[uctens < 0] = 0,0
              UCmax = max( [np.max(ele) for ele in [uccomp,uctens,uctau] ] )

              ucs = UCmax; UCm1='na'; UCm2='na'; UCm0 = UCmax


            print('layer 1:',lim(y0b,l0,-1,1),' > ',
                  lim(y1e,l1,-1,1),
                  lim(y0b,l0,-1,1) > lim(y1e,l1,-1,1)) # check if new brick in
                            #2nd layer gives overlap for a new brick in 1st layer
        # Layer 1 (2nd) ########################################
            if lim(y0b,l0,-1,1) > lim(y1e,l1,-1,1):# and (UCm0>1 or apexl0==True):
                  # a.l.a.p. += and (UCm0>1 or apexl0==True)
        #     if y0[max(len(l0)-1,0)]-lu/2 > y1[max(len(l1)-1,0)] or UCmax>1:
              l=1
              h,n = new(l,h_old)

              M,Vs,Ns,W,A = section_forces(n,y1,A1,dsdx1,dzdx1,h)
              dist, sigma_m, sigma_n, tau = sigmadef()

              sigma = sigma_M + sigma_m + sigma_N + sigma_n
              ttau = Tau + tau

              uccomp = unitc(sigma,inter,yield_k)
              uctens = unitc(sigma,inter,yield_t)
              uctau = ttau/fvk

              uccomp[uccomp < 0],uctens[uctens < 0] = 0,0
              UCmax = max( [np.max(ele) for ele in [uccomp,uctens,uctau] ] )

              ucs = UCmax; UCm1 = UCmax

            print('layer 2:',lim(y1b,l1,-1,1),' > ',
                  lim(y2e,l2,-1,1),
                  lim(y1b,l1,-1,1) > lim(y2e,l2,-1,1)) # check if new brick in
                        #3rd layer gives overlap for a new brick in 2nd layer

        # Layer 2 (3rd) ########################################
            if lim(y1b,l1,-1,1) > lim(y2e,l2,-1,1):# and runloop>14:
                  # a.l.a.p. += and runloop>14
              l=2
              h,n = new(l,h_old)

              M,Vs,Ns,W,A = section_forces(n,y2,A2,dsdx2,dzdx2,h)
              dist, sigma_m, sigma_n, tau = sigmadef()

              sigma = sigma_M + sigma_m + sigma_N + sigma_n
              ttau = Tau + tau

              uccomp = unitc(sigma,inter,yield_k)
              uctens = unitc(sigma,inter,yield_t)
```

```
        uctau = ttau/fvk

        uccomp[uccomp < 0],uctens[uctens < 0] = 0,0
        UCmax = max( [np.max(ele) for ele in [uccomp,uctens,uctau] ] )

        ucs = UCmax; UCm2 = UCmax

#Apex reached?
    if runloop>30:
        print(lim(y2e,l2,-1,1),lim(y1b,l1,-1,1),' ',
              lim(y1e,l1,-1,1),lim(y0b,l0,-1,1),' ',
              lim(y0,l0,-1,1),(L+2*lu)/2,' ', len(l0)>0)
    if lim(y2e,l2,-1,1)>lim(y1b,l1,-1,1) \
    and lim(y1e,l1,-1,1)>lim(y0b,l0,-1,1) and lim(y0,l0,-1,1)>(L+2*lu)/2 \
    and len(l0)>0:
        print("Apex has been reached! No more bricks can be added!")
        break
    if lim(y0,l0,-1,1)>(L+2*lu)/2 and len(l0)>0 and apexl0==False:
        print("Apex has been reached! The first layer is done!")
        apexl0=True
# Print UC's
    if l == 0:
        if isinstance(UCm0,np.float64):
            UC_0 = np.round(UCm0,3)
        else:
            UC_0 = 'finished'
        print('UC0 = %s' % (UC_0))

        UC_0
    elif l == 1:
        if isinstance(UCm0,np.float64):
            UC_0 = np.round(UCm0,3)
        else:
            UC_0 = 'finished'
        if isinstance(UCm1,np.float64):
            UC_1 = np.round(UCm1,3)
        else:
            UC_1 = 'no try'
        print('UC0, UC1 = %s, %s' % (UC_0,UC_1))

        UC_0
    elif l == 2:
        if isinstance(UCm0,np.float64):
            UC_0 = np.round(UCm0,3)
        else:
            UC_0 = 'finished'
        if isinstance(UCm1,np.float64):
            UC_1 = np.round(UCm1,3)
        else:
            UC_1 = 'no try'
        if isinstance(UCm2,np.float64):
            UC_2 = np.round(UCm2,3)
        else:
            UC_2 = 'no try'
        print('UC0, UC1, UC2 = %s, %s, %s' % (UC_0,UC_1,UC_2))
        UC_0

    if len([l0,l1,l2][l])<len([y0,y1,y2][l]):
        [l0,l1,l2][l].append(n)
    if l!=0:                #   inter   #
        inte = [ind for ind,ele in enumerate([i0,i1,i2][l]) if ele==n]
        inter[l,inte[0]:inte[-1]+1] = sigma[l,inte[0]:inte[-1]+1]

    seq.append(str(l) + ";" + str(n))

    h_old = h

    sigma_M = sigma_M + sigma_m     #is correct sign for position on cross-section
    sigma_N = sigma_N + sigma_n     #negative is correct
    sigma_norm = sigma_M+sigma_N
    Tau = Tau + tau #positive is correct

    UCseries.append(ucs)
    UCseries_.append([runloop,UCm0,UCm1,UCm2])

    Distr.append(runloop)

    Sigres.append(sigma_norm-inter)
    Tstrres.append(yield_t)
    Cstrres.append(yield_k)
    Cucres.append(uccomp)
    Tucres.append(uctens)

    setting = settings(l,n,setting)
```

```
            yield_t,yield_k = yielding(setting,h_old,t,set)

            indc = [[0,lst,ind] for (lst,ind),ele in np.ndenumerate(uccomp) if ele==UCmax]
            indt = [[1,lst,ind] for (lst,ind),ele in np.ndenumerate(uctens) if ele==UCmax]
            indv = [[2,lst,ind] for (lst,ind),ele in np.ndenumerate(uctau) if ele==UCmax]
            ut.append((indc or indt or indv)[0])

            if UCmax>1.0:
                print("")
                if UCmax==UCm2:
                    if type(UCm1)!='float': UCm1 = 3
                    print('Check this:      ',
                          'UC0, UC1, UC2 = %s, %s, %s' % (np.round(UCm0,3),
                                                          np.round(UCm1,3),np.round(UCm2,3)))
                print("UC too big!")
                if y0[n]>(L+2*lu)/2:
                    print('Apex has been reached in layer 1!')
                l_last,n_last = l,n
                del seq[-1]
                break
            print('brick %s placed in layer %s as row %s' % (str(runloop+1),str(l),str(n)))

            nope = False
            if nope:
                Sigd0[0].append(sigma_M[0,0])
                Sigd0[1].append(sigma_M[1,0])
                Sigd9[0].append(sigma_M[0,9])
                Sigd9[1].append(sigma_M[1,9])
                Sigd18[0].append(sigma_M[0,18])
                Sigd18[1].append(sigma_M[1,18])
                Yied0[0].append(yield_k[0,0])
                Yied0[1].append(yield_t[1,0])
                Yied9[0].append(yield_k[0,9])
                Yied9[1].append(yield_t[1,9])
                Yied18[0].append(yield_k[0,18])
                Yied18[1].append(yield_t[1,18])
                UCd[0].append(uccomp[0,0])
                UCd[1].append(uctens[1,0])

            uccomp = sigma/yield_k
            uctens = sigma/yield_t
            uctau = ttau/fvk

        print()
        print("FOR LOOP^")
        print("OUTPUT:")
```

```
layer 0: True
layer 1: 0.001942  >  0.061141 False
layer 2: -0.013907  >  0.096149 False
UC0 = 0.05
brick 1 placed in layer 0 as row 0
layer 0: True
layer 1: 0.004363  >  0.061141 False
layer 2: -0.013907  >  0.096149 False
UC0 = 0.412
brick 2 placed in layer 0 as row 1
layer 0: True
layer 1: 0.12204  >  0.061141 True
layer 2: -0.013907  >  0.096149 False
UC0, UC1 = 0.696, 0.147
brick 3 placed in layer 1 as row 0
layer 0: True
layer 1: 0.12204  >  0.180306 False
layer 2: -0.001186  >  0.096149 False
UC0 = 0.696
brick 4 placed in layer 0 as row 2
layer 0: True
layer 1: 0.24266  >  0.180306 True
layer 2: -0.001186  >  0.096149 False
UC0, UC1 = 0.766, 0.241
brick 5 placed in layer 1 as row 1
layer 0: True
layer 1: 0.24266  >  0.304216 False
layer 2: 0.117076  >  0.096149 True
UC0, UC1, UC2 = 0.721, no try, 0.188
brick 6 placed in layer 2 as row 0
layer 0: True
layer 1: 0.24266  >  0.304216 False
layer 2: 0.117076  >  0.272971 False
UC0 = 0.721
brick 7 placed in layer 0 as row 3
layer 0: True
layer 1: 0.366599  >  0.304216 True
```

```
layer 2: 0.117076  >  0.272971 False
UC0, UC1 = 0.793, 0.239
brick 8 placed in layer 1 as row 2
layer 0: True
layer 1: 0.366599  >  0.430439 False
layer 2: 0.239484  >  0.272971 False
UC0 = 0.771
brick 9 placed in layer 0 as row 4
layer 0: True
layer 1: 0.494935  >  0.430439 True
layer 2: 0.239484  >  0.272971 False
UC0, UC1 = 0.846, 0.342
brick 10 placed in layer 1 as row 3
layer 0: True
layer 1: 0.494935  >  0.562536 False
layer 2: 0.364786  >  0.272971 True
UC0, UC1, UC2 = 0.846, no try, 0.267
brick 11 placed in layer 2 as row 1
layer 0: True
layer 1: 0.494935  >  0.562536 False
layer 2: 0.364786  >  0.457876 False
UC0 = 0.846
brick 12 placed in layer 0 as row 5
layer 0: True
layer 1: 0.626936  >  0.562536 True
layer 2: 0.364786  >  0.457876 False
UC0, UC1 = 0.947, 0.284
brick 13 placed in layer 1 as row 4
layer 0: True
layer 1: 0.626936  >  0.698043 False
layer 2: 0.494101  >  0.457876 True
UC0, UC1, UC2 = 0.947, no try, 0.216
brick 14 placed in layer 2 as row 2
layer 0: True
layer 1: 0.626936  >  0.698043 False
layer 2: 0.494101  >  0.649016 False
UC0 = 0.947
brick 15 placed in layer 0 as row 6
layer 0: True
layer 1: 0.762191  >  0.698043 True
layer 2: 0.494101  >  0.649016 False
UC0, UC1 = 0.979, 0.319
brick 16 placed in layer 1 as row 5
layer 0: True
layer 1: 0.762191  >  0.836294 False
layer 2: 0.627385  >  0.649016 False
UC0 = 0.784
brick 17 placed in layer 0 as row 7
layer 0: True
layer 1: 0.900692  >  0.836294 True
layer 2: 0.627385  >  0.649016 False
UC0, UC1 = 0.931, 0.423
brick 18 placed in layer 1 as row 6
layer 0: True
layer 1: 0.900692  >  0.977588 False
layer 2: 0.764148  >  0.649016 True
UC0, UC1, UC2 = 0.931, no try, 0.329
brick 19 placed in layer 2 as row 3
layer 0: True
layer 1: 0.900692  >  0.977588 False
layer 2: 0.764148  >  0.848577 False
UC0 = 0.931
brick 20 placed in layer 0 as row 8
layer 0: True
layer 1: 1.042161  >  0.977588 True
layer 2: 0.764148  >  0.848577 False
UC0, UC1 = 0.985, 0.339
brick 21 placed in layer 1 as row 7
layer 0: True
layer 1: 1.042161  >  1.122094 False
layer 2: 0.90285  >  0.848577 True
UC0, UC1, UC2 = 0.81, no try, 0.378
brick 22 placed in layer 2 as row 4
layer 0: True
layer 1: 1.042161  >  1.122094 False
layer 2: 0.90285  >  1.055176 False
UC0 = 0.81
brick 23 placed in layer 0 as row 9
layer 0: True
layer 1: 1.186268  >  1.122094 True
layer 2: 0.90285  >  1.055176 False
UC0, UC1 = 0.993, 0.464
brick 24 placed in layer 1 as row 8
layer 0: True
```

```
layer 1: 1.186268  >  1.268772 False
layer 2: 1.045686  >  1.055176 False
UC0 = 0.993
brick 25 placed in layer 0 as row 10
layer 0: True
layer 1: 1.332786  >  1.268772 True
layer 2: 1.045686  >  1.055176 False
UC0, UC1 = 1.043, 0.561
brick 26 placed in layer 1 as row 9
layer 0: True
layer 1: 1.332786  >  1.418042 False
layer 2: 1.191479  >  1.055176 True
UC0, UC1, UC2 = 0.903, no try, 0.612
brick 27 placed in layer 2 as row 5
layer 0: True
layer 1: 1.332786  >  1.418042 False
layer 2: 1.191479  >  1.266654 False
UC0 = 0.903
brick 28 placed in layer 0 as row 11
layer 0: True
layer 1: 1.481037  >  1.418042 True
layer 2: 1.191479  >  1.266654 False
UC0, UC1 = 0.986, 0.721
brick 29 placed in layer 1 as row 10
layer 0: True
layer 1: 1.481037  >  1.568822 False
layer 2: 1.339571  >  1.266654 True
UC0, UC1, UC2 = 0.823, no try, 0.784
brick 30 placed in layer 2 as row 6
layer 0: True
layer 1: 1.481037  >  1.568822 False
layer 2: 1.339571  >  1.482252 False
UC0 = 0.848
brick 31 placed in layer 0 as row 12
layer 0: True
layer 1: 1.630597  >  1.568822 True
layer 2: 1.339571  >  1.482252 False
1.482252 1.339571   1.568822 1.630597   1.7444 2.0100000000000002   True
UC0, UC1 = 1.036, 0.905
brick 32 placed in layer 1 as row 11
layer 0: True
layer 1: 1.630597  >  1.720447 False
layer 2: 1.489541  >  1.482252 True
1.482252 1.489541   1.720447 1.630597   1.7444 2.0100000000000002   True
UC0, UC1, UC2 = 1.036, no try, 0.981
brick 33 placed in layer 2 as row 7
layer 0: True
layer 1: 1.630597  >  1.720447 False
layer 2: 1.489541  >  1.700828 False
1.700828 1.489541   1.720447 1.630597   1.7444 2.0100000000000002   True
UC0 = 1.05

UC too big!

FOR LOOP^
OUTPUT:
```

```
In [6]: # export Grasshopper
        # print(np.round(sigma,2))
        sig_com = min([min(ele) for ele in sigma])
        sig_ten = max([max(ele) for ele in sigma])
        taut = max([max(ele) for ele in Tau])

        #location of highest compressive stresses compared to strength there
        sind = [[lst,ind] for (lst,ind),ele in np.ndenumerate(sigma) if ele==sig_com][0]
        print('l:',sind[0],'br:',sind[1]), print(
            'f_k',sigma[sind[0]][sind[1]]), print('f_R',yield_k[sind[0]][sind[1]])

        #location of highest tensile stresses compared to strength there
        sind = [[lst,ind] for (lst,ind),ele in np.ndenumerate(sigma) if ele==sig_ten][0]
        print('l:',sind[0],'br:',sind[1]), print(
            'f_tk',sigma[sind[0]][sind[1]]), print('f_tR',yield_t[sind[0]][sind[1]])

        #location of highest shear stresses compared to strength there
        sind = [[lst,ind] for (lst,ind),ele in np.ndenumerate(Tau) if ele==taut][0]
        print('l:',sind[0],'br:',sind[1]), print(
            'f_vk',Tau[sind[0]][sind[1]]), print('f_vR',fvk)

        maxcomp, maxtens, maxshear = np.max(uccomp), np.max(uctens), np.max(uctau)
        sind = [[lst,ind] for (lst,ind),ele in np.ndenumerate(uctens) if ele==maxtens][0]
        print(sind),print()

        print('maximum unity checks:',"maxcomp: ",np.round(maxcomp,3),
```

```
            " maxtens: ", np.round(maxtens,3),
            " maxshear: ", np.round(maxshear,3)), print()
        print(seq),print(len(seq)),print()
        print (np.round(np.array(UCseries),3)),print()

        UCseries_ = [[np.round(ele,1) if
                      type(ele)==np.float64 else ele for ele in UCseries_[j] ]
                     for j in range(len(UCseries_))]
        for i in range(len(UCseries_)):
            print(UCseries_[i])
        layer = [ele.split(';')[0] for ele in seq]
        brick = [ele.split(';')[1] for ele in seq]
        print()
        names = ['names','x','i0','i1','i2','h','sigma','seq']
        output = [names,x,i0,i1,i2,h,sigma,seq]
        output = [ele.tolist() if type(ele)==np.ndarray else ele for ele in output]

        ###                                  export

        np.savetxt('C:/Users/Joris/Documents/Master Thesis quick access/Python_to_GH.csv',
                   output, delimiter = ';', fmt = '%s')
        #print(output)

        a = [['   x','  y0','  y1','  y2','   h','sigm0','sigm1','sigm2','sigm3']]
        b = np.transpose([x,i0,i1,i2,h,sigma[0],sigma[1],sigma[2],sigma[3]])
        b = [[str("{:.2f},".format(ele)) for ele in b[j]] for j in range(len(b))]
        a += b
        # print(np.array(a))
```

```
l: 0 br: 1
f_k -2.3276665855266727
f_R -6.106357973053566
l: 3 br: 0
f_tk 2.09957120344815
f_tR 2.0
l: 0 br: 1
f_vk 0.11499622611018762
f_vR 0.3
[5, 0]

maximum unity checks: maxcomp:  1.036  maxtens:  1.05  maxshear:  0.383

['layer;brick', '0;0', '0;1', '1;0', '0;2', '1;1', '2;0', '0;3', '1;2', '0;4', '1;3', '2;1', '0;5', '1;4', '2;2', '0;6',
 '1;5', '0;7', '1;6', '2;3', '0;8', '1;7', '2;4', '0;9', '1;8', '0;10', '1;9', '2;5', '0;11', '1;10', '2;6', '0;12',
  '1;11', '2;7']
34

[0.05  0.412 0.147 0.696 0.241 0.188 0.721 0.239 0.771 0.342 0.267 0.846
 0.284 0.216 0.947 0.319 0.784 0.423 0.329 0.931 0.339 0.378 0.81  0.464
 0.993 0.561 0.612 0.903 0.721 0.784 0.848 0.905 0.981 1.05 ]

[0, 0.1, 'na', 'na']
[1, 0.4, 'na', 'na']
[2, 0.7, 0.1, 'na']
[3, 0.7, 'na', 'na']
[4, 0.8, 0.2, 'na']
[5, 0.7, 'na', 0.2]
[6, 0.7, 'na', 'na']
[7, 0.8, 0.2, 'na']
[8, 0.8, 'na', 'na']
[9, 0.8, 0.3, 'na']
[10, 0.8, 'na', 0.3]
[11, 0.8, 'na', 'na']
[12, 0.9, 0.3, 'na']
[13, 0.9, 'na', 0.2]
[14, 0.9, 'na', 'na']
[15, 1.0, 0.3, 'na']
[16, 0.8, 'na', 'na']
[17, 0.9, 0.4, 'na']
[18, 0.9, 'na', 0.3]
[19, 0.9, 'na', 'na']
[20, 1.0, 0.3, 'na']
[21, 0.8, 'na', 0.4]
[22, 0.8, 'na', 'na']
[23, 1.0, 0.5, 'na']
[24, 1.0, 'na', 'na']
[25, 1.0, 0.6, 'na']
[26, 0.9, 'na', 0.6]
[27, 0.9, 'na', 'na']
[28, 1.0, 0.7, 'na']
[29, 0.8, 'na', 0.8]
[30, 0.8, 'na', 'na']
[31, 1.0, 0.9, 'na']
[32, 1.0, 'na', 1.0]
```

```
[33, 1.0, 'na', 'na']
```

```
In [7]: # export Excel
        # distr = [Distr,Sigd0[0],Sigd0[1],Sigd9[0],Sigd9[1],Sigd18[0],Sigd18[1], Yied0[0],Yied0[1],Yied9[0],Yied9[1],
        #Yied18[0],Yied18[1],UCd[0],UCd[1]]
        # for i in range(len(distr)):
        #     distr[i].insert(0,['n',0,0,9,9,18,18,0,0,9,9,18,18,0,0][i])
        print(len(Sigres))
        print(len(Sigres[0]))
        print(len(Sigres[0][0]))
        sigres = np.transpose(Sigres)
        tres = np.transpose(Tstrres)
        cres = np.transpose(Cstrres)

        distr = []
        list = [sigres[0][0],sigres[0][1],sigres[0][2],sigres[0][3],
                cres[0][0],cres[0][1],tres[0][1],tres[0][2],tres[0][3]]
        for i in range(len(list)):
            distr.append(list[i])
        print(sigres[0][3][0:50])

        np.savetxt('C:/Users/Joris/Documents/Master Thesis quick access/Stress_dist.txt',
                   distr, delimiter = ';', fmt = '%s')

        # for i in range(len(Sigres)):
        #     distr = Sigres[i]
        #     np.savetxt('C:/Users/Joris/Documents/Master Thesis quick access/Stress_dist_%s.txt' % (i), distr,
        # delimiter = ';', fmt = '%s')
        #print(output)
```

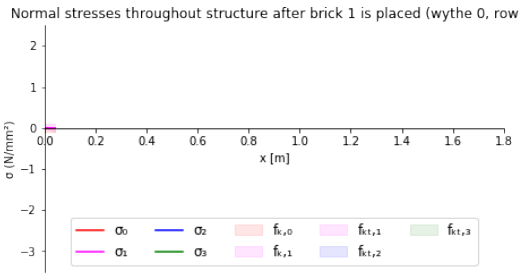
```
34
4
91
[0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00 3.68681070e-04 2.68415776e-02 4.29403158e-02
 7.95325484e-02 1.05537275e-01 1.21254326e-01 1.68249483e-01
 2.04495470e-01 2.40661554e-01 2.98343542e-01 3.45111306e-01
 4.13746351e-01 4.71307812e-01 5.28712031e-01 6.08544540e-01
 6.77174465e-01 7.56770657e-01 8.48020906e-01 9.27973034e-01
 1.03082873e+00 1.12231642e+00 1.22487941e+00 1.33949234e+00
 1.44272744e+00 1.56898081e+00 1.69546215e+00 1.81058390e+00
 1.96115600e+00 2.09957120e+00]
```

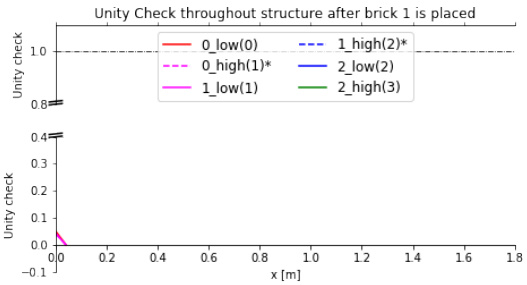
```
In [8]: def sub(x):
                normal = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+-=()"
                subs = "CDGQwZw"
                res = x.maketrans(''.join(normal), ''.join(subs))
                return x.translate(res)
```

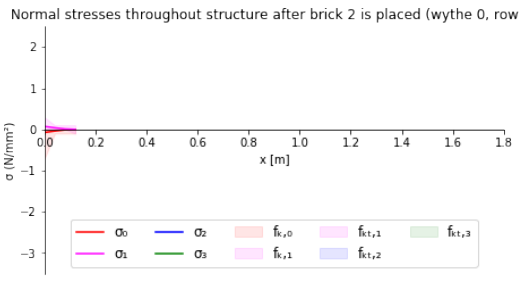
```
In [9]: # normal stresses after brick[i]
        ll = [int(ele.split(';')[0]) for ele in seq[1:]]
        nn = [int(ele.split(';')[1]) for ele in seq[1:]]
        ll.append(l_last),nn.append(n_last)
        # ll.append(1),nn.append(8)
        n0,n1,n2 = -2,-2,-2
        id0,id1,id2 = 0,0,0

        COres=np.array([[[0 if Sigres[brk][lay][idx]>0 else ele
                          for idx,ele in enumerate(Cstrres[brk][lay])]
                         for lay,Lay in enumerate(Cstrres[brk])]
                        for brk,Brk in enumerate(Cstrres)])
        TOres=np.array([[[0 if Sigres[brk][lay][idx]<0 else ele
                          for idx,ele in enumerate(Tstrres[brk][lay])]
                         for lay,Lay in enumerate(Tstrres[brk])]
                        for brk,Brk in enumerate(Tstrres)])
        UCres=np.array([[[Tucres[brk][lay][idx] if ele==0 else ele
                          for idx,ele in enumerate(Cucres[brk][lay])]
                         for lay,Lay in enumerate(Cucres[brk])]
                        for brk,Brk in enumerate(Cucres)])
        UC_maxi = np.array([[np.max(UCres[-1][:,idx])
                             for idx,ele in enumerate(UCres[-1][lst])]
                            for lst,Lst in enumerate(UCres[-1])][0])
        plotall = []
        for i in range(len(Sigres)):
        # for i in range(4):
            n0,n1,n2 = nn[i] if ll[i]==0 else n0, nn[i] if \
                ll[i]==1 else n1, nn[i] if ll[i]==2 else n2
            id0,id1,id2 = i0.index(n0+1) if n0>-1 else -1, i1.index(n1+1) if \
                n1>-1 else -1, i2.index(n2+1) if n2>-1 else -1
        #     print('n',n0+1,n1+1,n2+1)
```

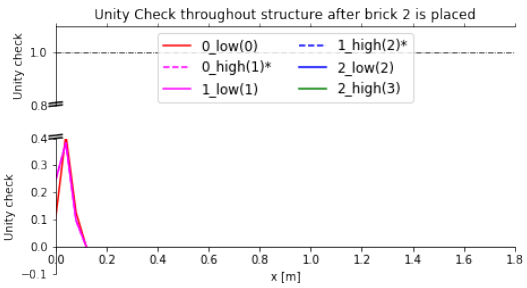
```
    plt.rcParams["figure.figsize"] = (16,4) # (w, h)
#     fig, (ax0,ax1,ax2) = plt.subplots(2,2,sharex=True,sharey=False)
    ax0 = plt.subplot(1,2,1)
    ax1 = plt.subplot(9,2,(2,6), sharex=ax0)
    ax2 = plt.subplot(9,2,(10,18), sharex=ax0)
    AX = [ax1,ax2]
    Yl = [[.8,1.1],[-.1,.4]]
#
    ax0.spines['left'].set_position('zero')
    ax0.spines['right'].set_color('none')
    ax0.yaxis.tick_left()

    ax0.spines['bottom'].set_position('zero')

    ax0.spines['top'].set_color('none')
    ax0.xaxis.tick_bottom()

    xx = x
#     print(Sigres[i][0][:id0+1])
    ax0.plot(xx[:id0+1],Sigres[i][0][:id0+1], label = '\u03C3%s' % (sub('0')),
             color='red')
    ax0.plot(xx[:id0+1],Sigres[i][1][:id0+1], label = '\u03C3%s' % (sub('1')),
             color='magenta')
    ax0.plot(xx[:id1+1],Sigres[i][2][:id1+1], label = '\u03C3%s' % (sub('2')),
             color='blue')
    ax0.plot(xx[:id2+1],Sigres[i][3][:id2+1], label = '\u03C3%s' % (sub('3')),
             color='green')

    alp = .1
    ax0.fill_between(xx[:id0+1],C0res[i][0][:id0+1], color='red',alpha=alp,
                     label = 'f%s' % (sub('k,0')))
    ax0.fill_between(xx[:id0+1],C0res[i][1][:id0+1], color='magenta',alpha=alp,
                     label = 'f%s' % (sub('k,1')))
    ax0.fill_between(xx[:id0+1],T0res[i][1][:id0+1], color='magenta',alpha=alp,
                     label = 'f%s' % (sub('kt,1')))
    ax0.fill_between(xx[:id1+1],Tstrres[i][2][:id1+1], color='blue',alpha=alp,
                     label = 'f%s' % (sub('kt,2')))
    ax0.fill_between(xx[:id2+1],Tstrres[i][3][:id2+1], color='green',alpha=alp,
                     label = 'f%s' % (sub('kt,3')))


    ax0.set_xlabel('x [m]')
    ax0.set_ylabel('\u03C3 (N/mmš)')
    ax0.title.set_text(
        'Normal stresses throughout structure after brick %s is placed (wythe %s, row %s)' % (i+1,ll[i],nn[i]))

    ax0.legend(loc=8, prop={"size":12},ncol=5)
    ax0.set_ylim([-3.5,2.5]), ax0.set_xlim([0,L/2])

    # # # # # # # # # # # # # #

    ax1.spines['bottom'].set_visible(False)
    ax2.spines['top'].set_color('none')

    ax2.spines['top'].set_visible(False)
    ax2.spines['bottom'].set_position('zero')

    ax2.xaxis.tick_bottom()
    ax1.xaxis.tick_top()
    ax1.tick_params(labeltop=False)
    ax2.spines['left'].set_position('zero')
    d,D = .014,.06
    rate = [.625,.375]

    for j in range(len(AX)):
        ax = AX[j]
        ax.spines['right'].set_color('none')
        ax.yaxis.tick_left()

        ax.plot(xx[:id0+1],Cucres[i][0][:id0+1], label = '0_low(0)',
                color='red')
        ax.plot(xx[:id0+1], UCres[i][1][:id0+1], label = '0_high(1)*',
                color='magenta',linestyle='dashed')
        ax.plot(xx[:id0+1], UCres[i][2][:id0+1], label = '1_low(1)',
                color='magenta')
        ax.plot(xx[:id1+1],Tucres[i][3][:id1+1], label = '1_high(2)*',
                color='blue',linestyle='dashed')
        ax.plot(xx[:id1+1],Tucres[i][4][:id1+1], label = '2_low(2)',
                color='blue')
        ax.plot(xx[:id2+1],Tucres[i][5][:id2+1], label = '2_high(3)',
                color='green')
        ax.hlines(1,0,L/2,color='k',linewidth=.75,linestyle='dashdot')

        ax.set_ylabel('Unity check')
```

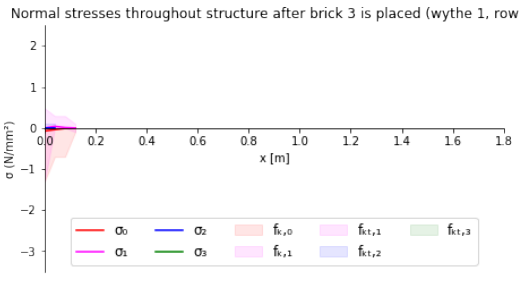
```
        ax.set_ylim(Yl[j])#, ax.set_xlim([0,L/2])

        dj,Dj = rate[j]*d,rate[j]*D
        kwargs = dict(transform=ax.transAxes, color='k', clip_on=False)
        ax.plot((-d, +d), (j-dj, j+dj), **kwargs)  # bottom-left diagonal
        ax.plot((-d, +d), (j-dj+Dj, j+dj+Dj), **kwargs)  # bottom-left diagonal

    ax1.title.set_text(
        'Unity Check throughout structure after brick %s is placed' % (i+1))
    ax2.set_xlabel('x [m]')
    ax1.legend(loc=9, prop={"size":12},ncol=2)
    if i == len(Sigres)-1:
        try:
            del textp
        except:
            print()
        annot_max(xx,UC_maxi, ax=ax1,textp=[0.015,0.9])
        xmax = xx[np.argmax(UC_maxi)]
        ax0.vlines(xmax,-3.5,2.5,linestyle=(0,(2,12)))
#         print(id0,id1,id2)
    plt.show()
```

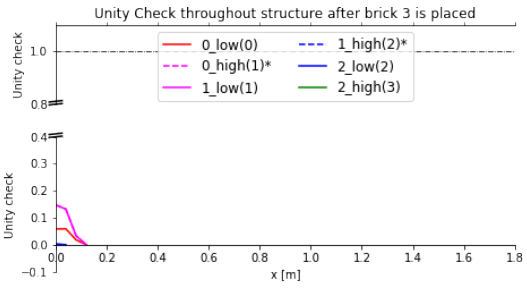


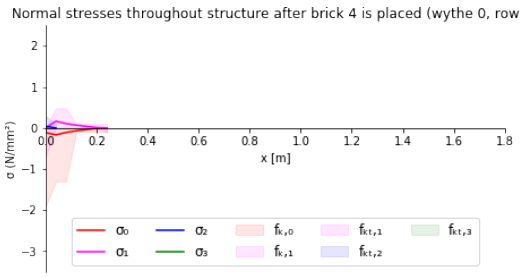


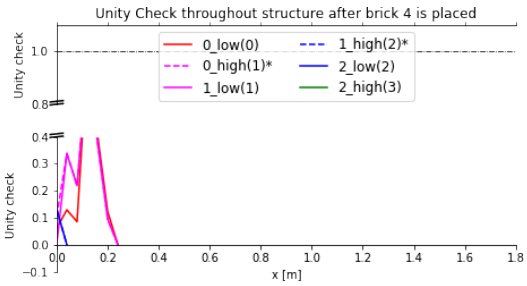


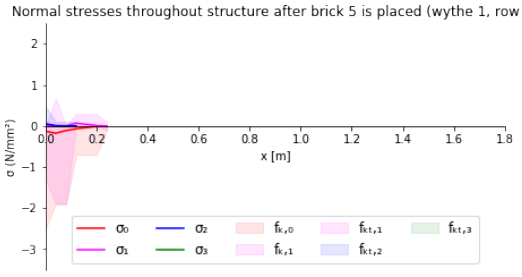


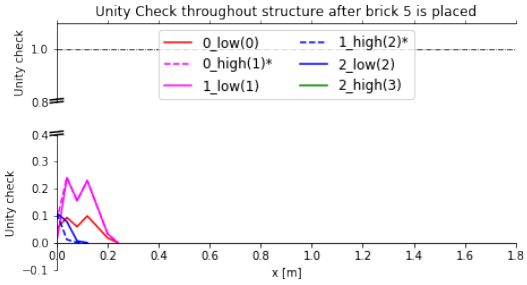


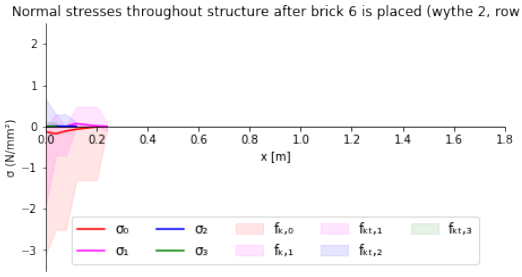


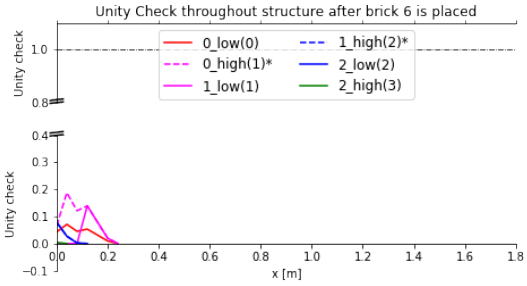

Normal stresses throughout structure after brick 4 is placed (wythe 0, row 2)
σ (N/mm²)
x [m]
σ0
σ1
σ2
σ3
fk,0
fk,1
fkt,1
fkt,2
fkt,3

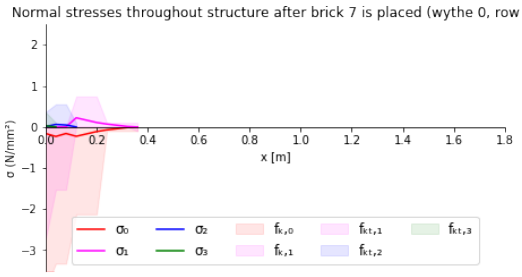

Unity Check throughout structure after brick 4 is placed
Unity check
x [m]
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

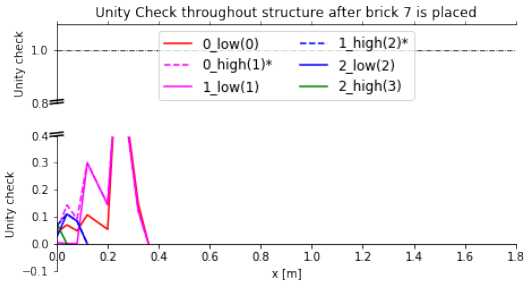

Normal stresses throughout structure after brick 5 is placed (wythe 1, row 1)
σ (N/mm²)
x [m]
σ0
σ1
σ2
σ3
fk,0
fk,1
fkt,1
fkt,2
fkt,3

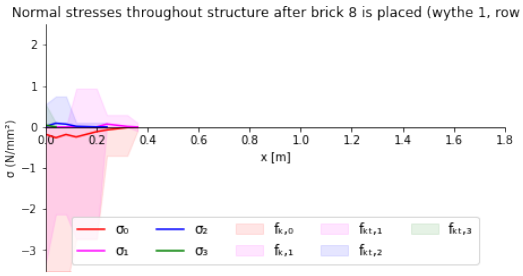

Unity Check throughout structure after brick 5 is placed
Unity check
x [m]
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

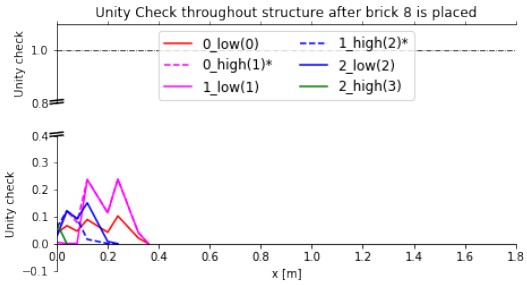

Normal stresses throughout structure after brick 6 is placed (wythe 2, row 0)
σ (N/mm²)
x [m]
σ0
σ1
σ2
σ3
fk,0
fk,1
fkt,1
fkt,2
fkt,3

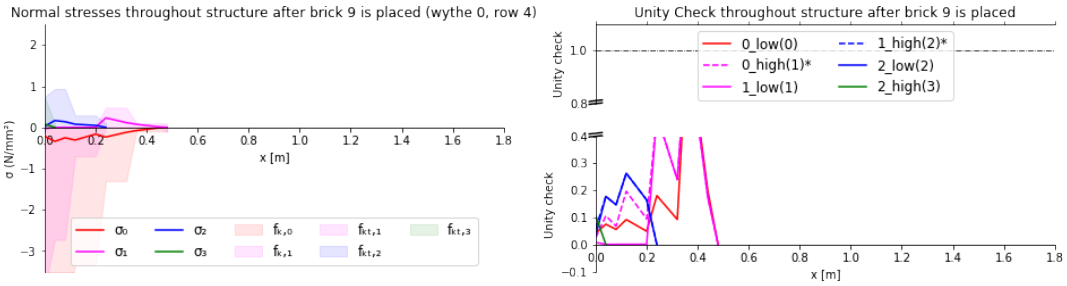

Unity Check throughout structure after brick 6 is placed
Unity check
x [m]
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

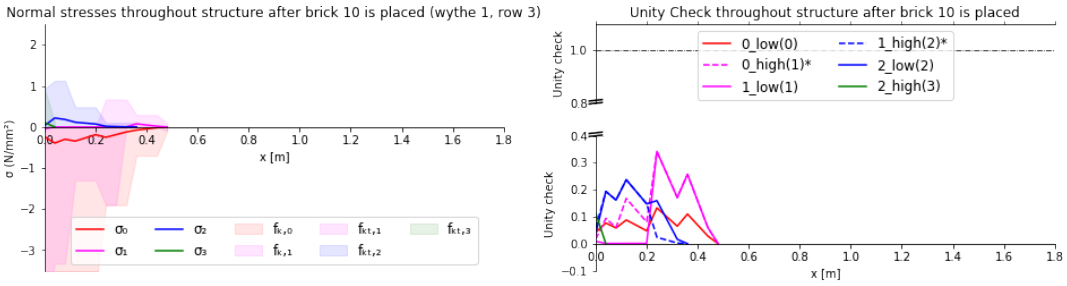

Normal stresses throughout structure after brick 7 is placed (wythe 0, row 3)
σ (N/mm²)
x [m]
σ0
σ1
σ2
σ3
fk,0
fk,1
fkt,1
fkt,2
fkt,3

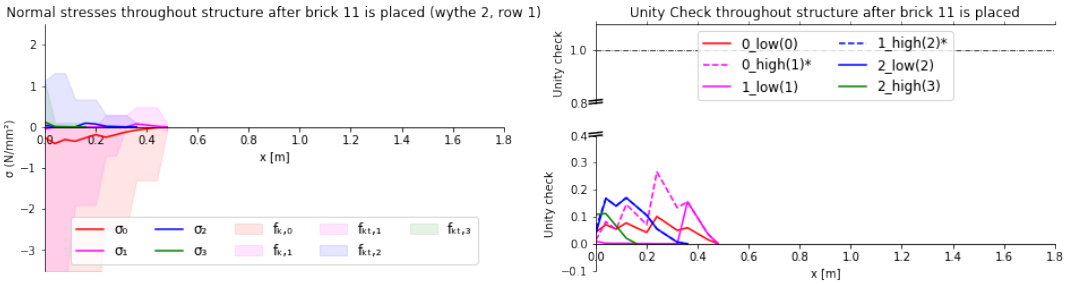

Unity Check throughout structure after brick 7 is placed
Unity check
x [m]
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

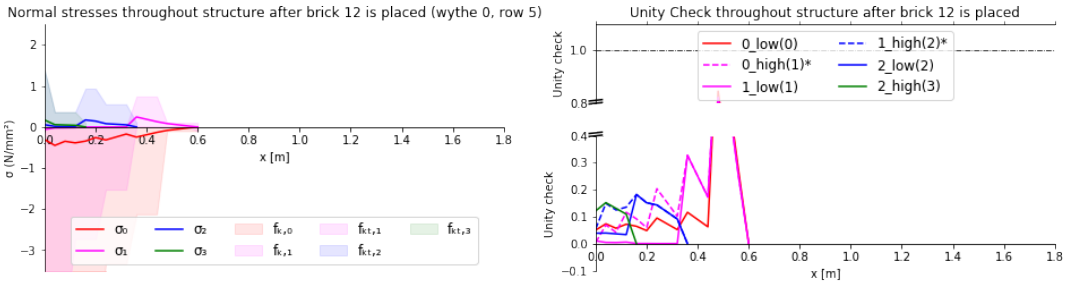

Normal stresses throughout structure after brick 8 is placed (wythe 1, row 2)
σ (N/mm²)
x [m]
σ0
σ1
σ2
σ3
fk,0
fk,1
fkt,1
fkt,2
fkt,3

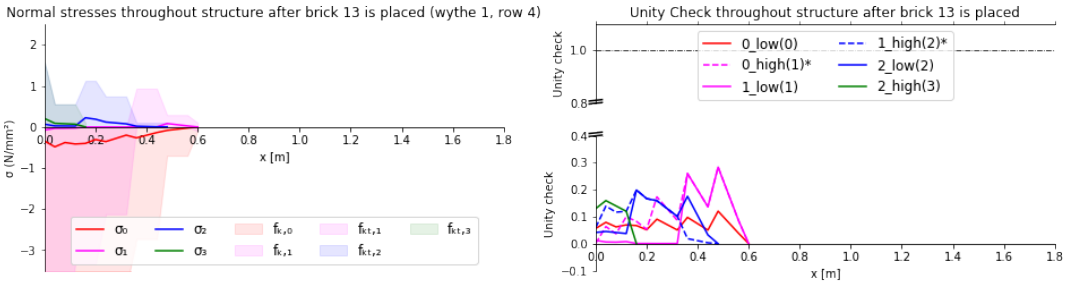

Unity Check throughout structure after brick 8 is placed
Unity check
x [m]
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

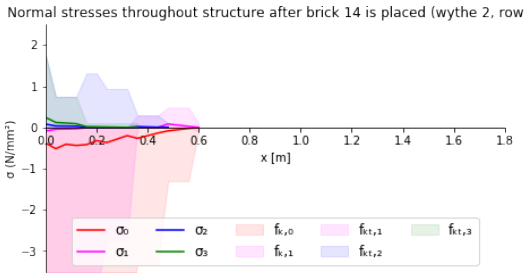

Normal stresses throughout structure after brick 9 is placed (wythe 0, row 4)
Unity Check throughout structure after brick 9 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

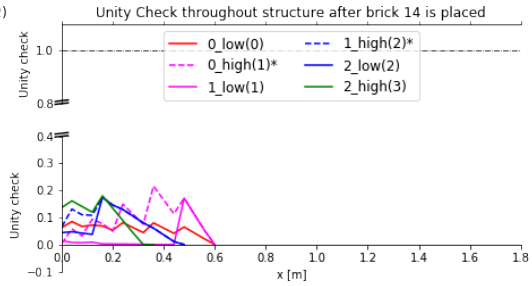

Normal stresses throughout structure after brick 10 is placed (wythe 1, row 3)
Unity Check throughout structure after brick 10 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

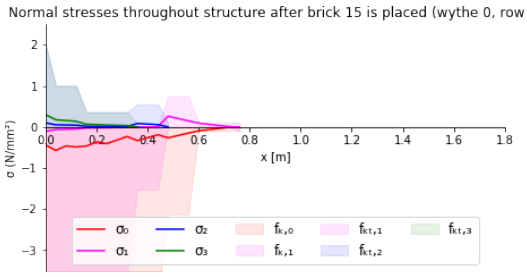

Normal stresses throughout structure after brick 11 is placed (wythe 2, row 1)
Unity Check throughout structure after brick 11 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

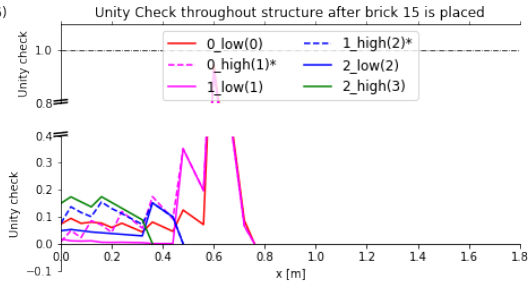

Normal stresses throughout structure after brick 12 is placed (wythe 0, row 5)
Unity Check throughout structure after brick 12 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

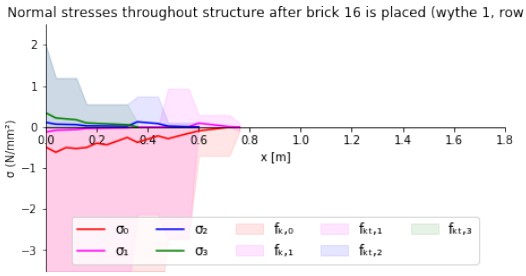

Normal stresses throughout structure after brick 13 is placed (wythe 1, row 4)
Unity Check throughout structure after brick 13 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

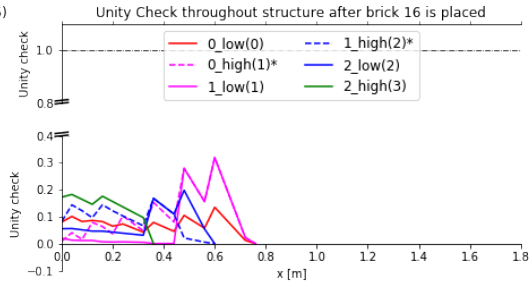

Normal stresses throughout structure after brick 14 is placed (wythe 2, row 2)
σ (N/mm²)
x [m]
σ0
σ1
σ2
σ3
fk,0
fk,1
fkt,1
fkt,2
fkt,3

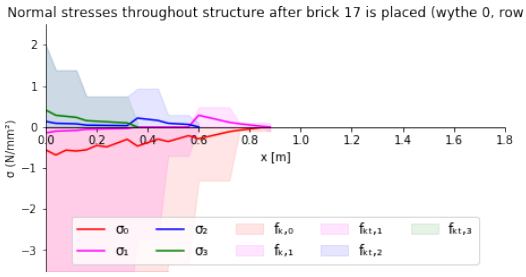

Unity Check throughout structure after brick 14 is placed
Unity check
x [m]
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

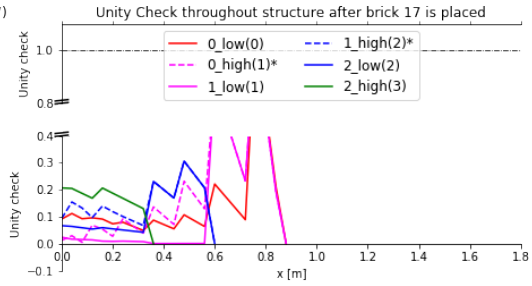

Normal stresses throughout structure after brick 15 is placed (wythe 0, row 6)
σ (N/mm²)
x [m]
σ0
σ1
σ2
σ3
fk,0
fk,1
fkt,1
fkt,2
fkt,3

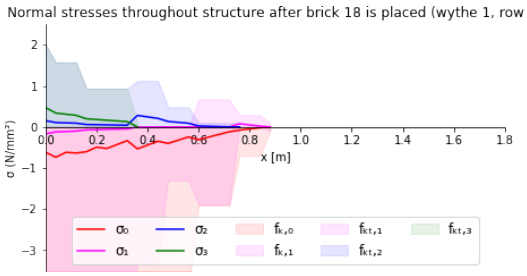

Unity Check throughout structure after brick 15 is placed
Unity check
x [m]
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

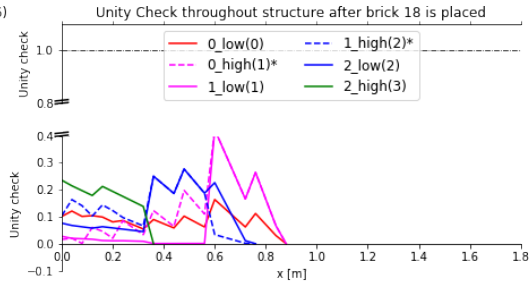

Normal stresses throughout structure after brick 16 is placed (wythe 1, row 5)
σ (N/mm²)
x [m]
σ0
σ1
σ2
σ3
fk,0
fk,1
fkt,1
fkt,2
fkt,3

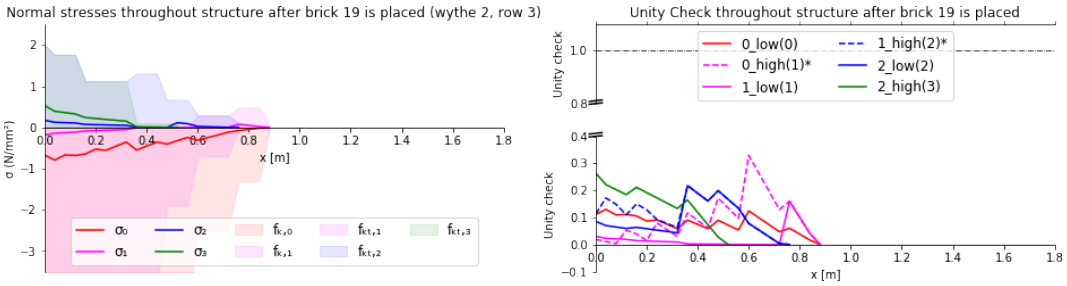

Unity Check throughout structure after brick 16 is placed
Unity check
x [m]
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

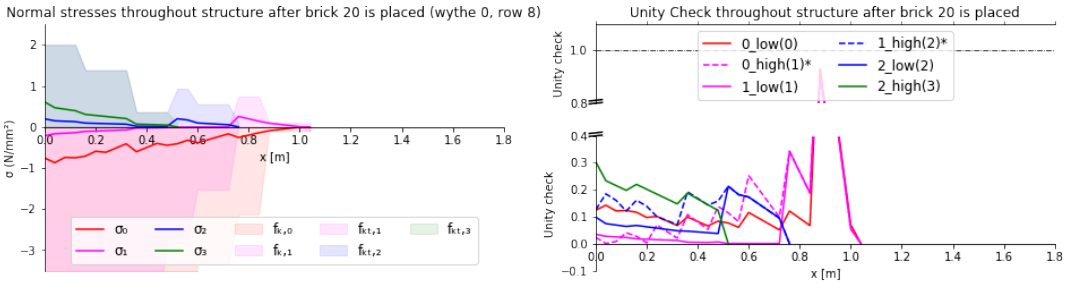

Normal stresses throughout structure after brick 17 is placed (wythe 0, row 7)
σ (N/mm²)
x [m]
σ0
σ1
σ2
σ3
fk,0
fk,1
fkt,1
fkt,2
fkt,3

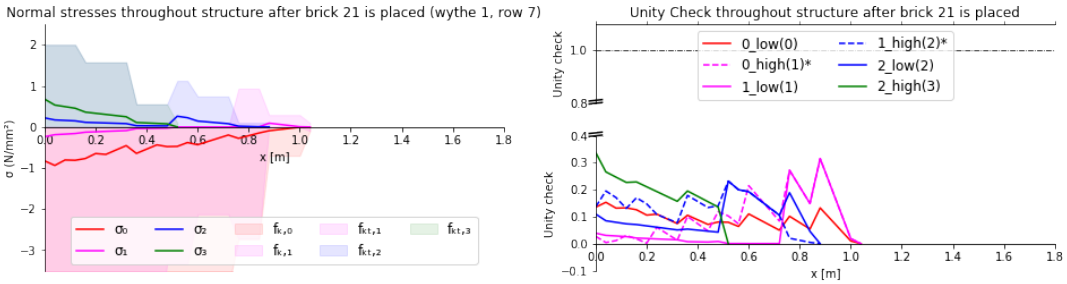

Unity Check throughout structure after brick 17 is placed
Unity check
x [m]
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

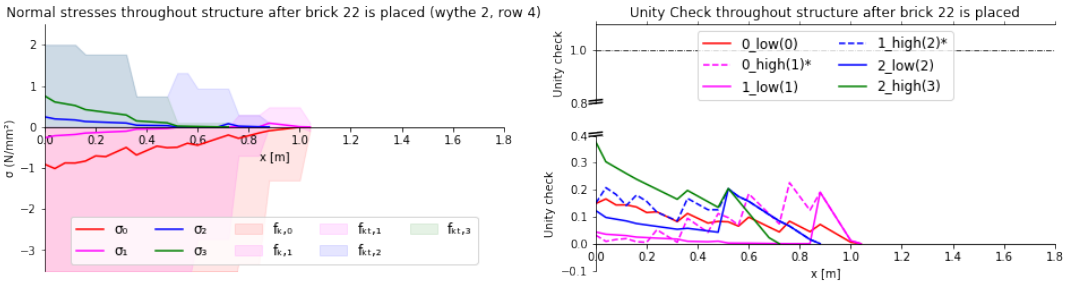

Normal stresses throughout structure after brick 18 is placed (wythe 1, row 6)
σ (N/mm²)
x [m]
σ0
σ1
σ2
σ3
fk,0
fk,1
fkt,1
fkt,2
fkt,3

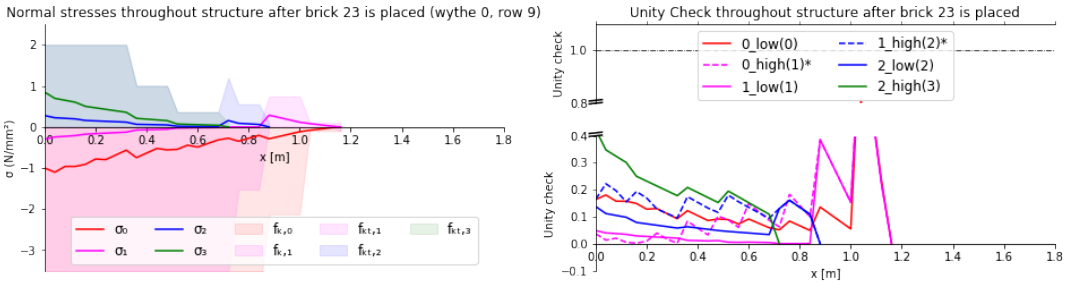

Unity Check throughout structure after brick 18 is placed
Unity check
x [m]
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

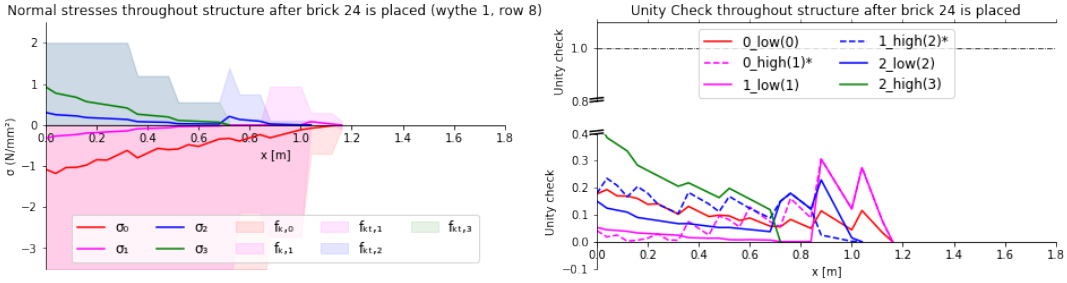

Normal stresses throughout structure after brick 19 is placed (wythe 2, row 3)
Unity Check throughout structure after brick 19 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

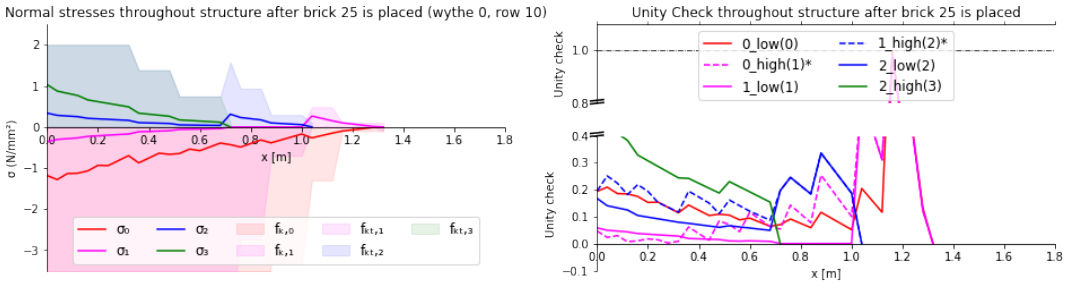

Normal stresses throughout structure after brick 20 is placed (wythe 0, row 8)
Unity Check throughout structure after brick 20 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

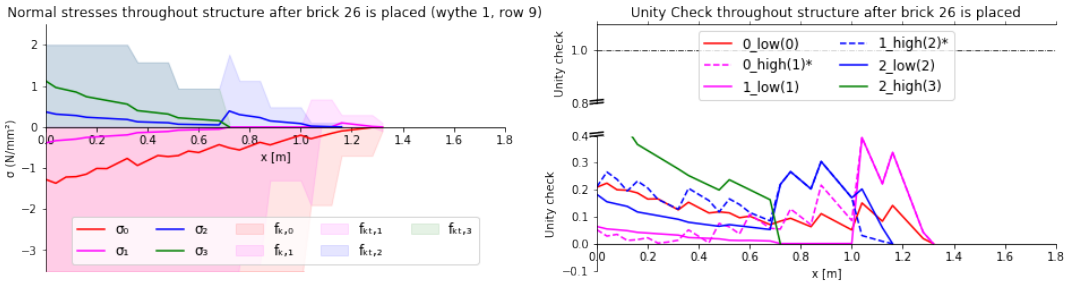

Normal stresses throughout structure after brick 21 is placed (wythe 1, row 7)
Unity Check throughout structure after brick 21 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

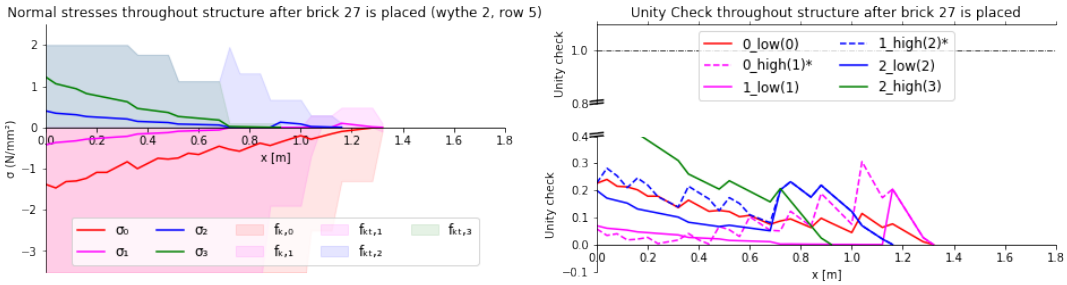

Normal stresses throughout structure after brick 22 is placed (wythe 2, row 4)
Unity Check throughout structure after brick 22 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

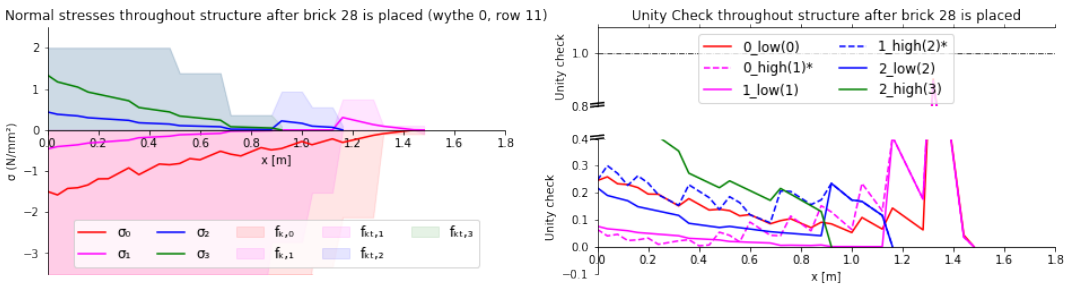

Normal stresses throughout structure after brick 23 is placed (wythe 0, row 9)
Unity Check throughout structure after brick 23 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

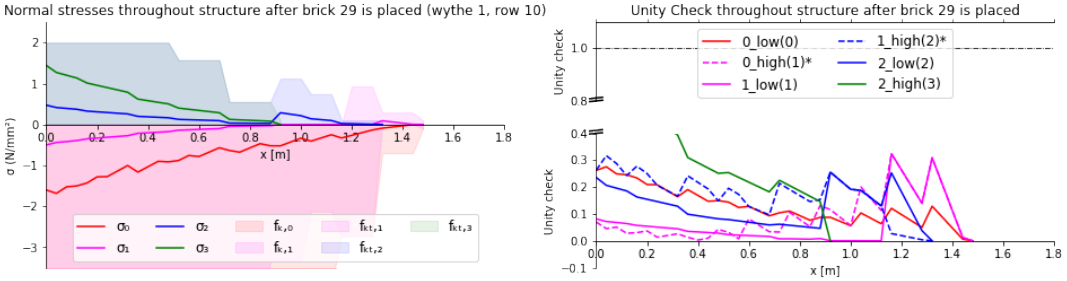

Normal stresses throughout structure after brick 24 is placed (wythe 1, row 8)
Unity Check throughout structure after brick 24 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

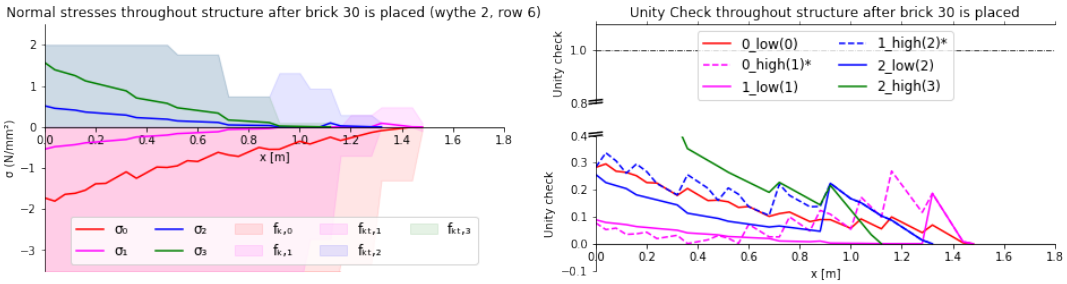

Normal stresses throughout structure after brick 25 is placed (wythe 0, row 10)
Unity Check throughout structure after brick 25 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

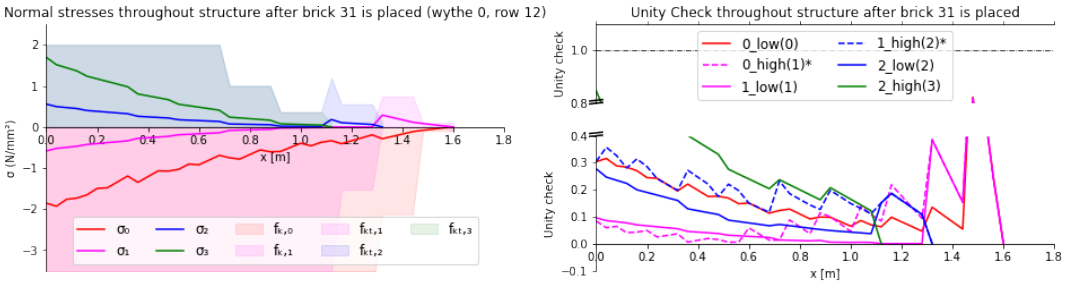

Normal stresses throughout structure after brick 26 is placed (wythe 1, row 9)
Unity Check throughout structure after brick 26 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

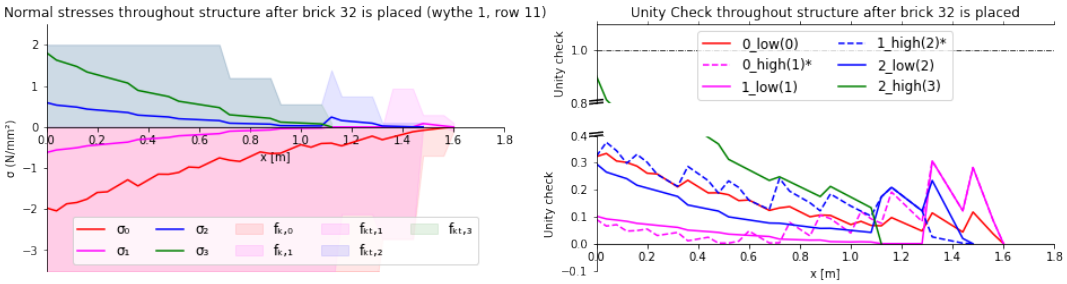

Normal stresses throughout structure after brick 27 is placed (wythe 2, row 5)
Unity Check throughout structure after brick 27 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

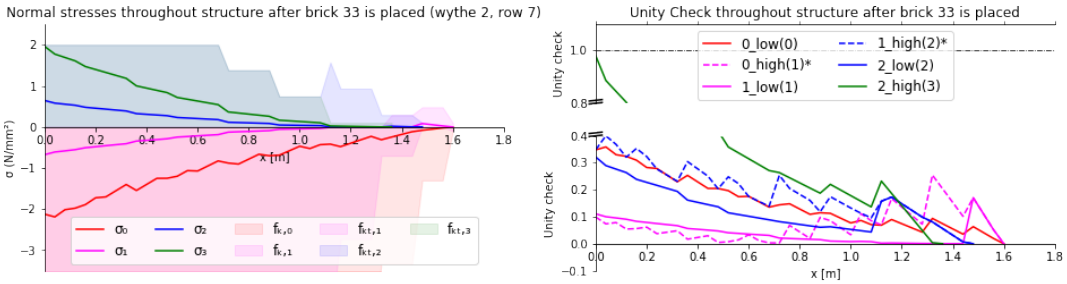

Normal stresses throughout structure after brick 28 is placed (wythe 0, row 11)
Unity Check throughout structure after brick 28 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

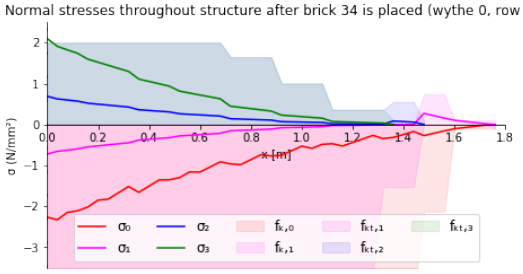

Normal stresses throughout structure after brick 29 is placed (wythe 1, row 10)
Unity Check throughout structure after brick 29 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)

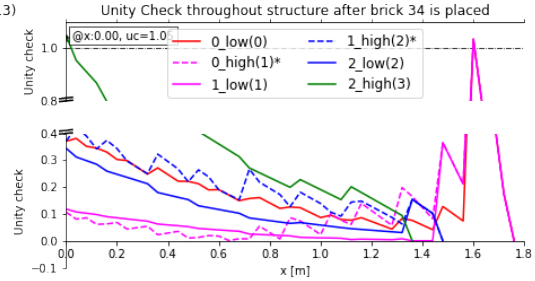

Normal stresses throughout structure after brick 30 is placed (wythe 2, row 6)
Unity Check throughout structure after brick 30 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)


Normal stresses throughout structure after brick 31 is placed (wythe 0, row 12)
Unity Check throughout structure after brick 31 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)


Normal stresses throughout structure after brick 32 is placed (wythe 1, row 11)
Unity Check throughout structure after brick 32 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)


Normal stresses throughout structure after brick 33 is placed (wythe 2, row 7)
Unity Check throughout structure after brick 33 is placed
σ (N/mm²)
x [m]
Unity check
0_low(0)
0_high(1)*
1_low(1)
1_high(2)*
2_low(2)
2_high(3)





In [10]:
```
#
get = 28
Cfan = np.array(Cstrres)[:,:,get]
Tfan = np.array(Tstrres)[:,:,get]
Sigan = np.array(Sigres)[:,:,get]

start = [min([idx if ele==wth else len(ll) for idx,ele in enumerate(ll)])
         for wth in [0,1,2]]
start.append(start[-1])

Inter = np.array([[0 if lst<start[idx] else ele
                   for idx,ele in enumerate(inter[:,get])]
                  for lst,Lst in enumerate(range(len(seq)))])

Sigani = np.array([np.zeros(len(seq)),Sigan[:,0],Sigan[:,1]+Inter[:,1],Sigan[:,1],
                   Sigan[:,2]+Inter[:,2],Sigan[:,2],Sigan[:,3],np.zeros(len(seq))])
Cfani = np.array([Cfan[:,0],Cfan[:,0],Cfan[:,1],Cfan[:,1],Cfan[:,2],Cfan[:,3]])
Tfani = np.array([Tfan[:,0],Tfan[:,0],Tfan[:,1],Tfan[:,1],Tfan[:,2],Tfan[:,3]])
Cfani[2:4,:2],Tfani[2:4,:2] = np.nan,np.nan
Cfani[4:7,:6],Tfani[4:7,:6] = np.nan,np.nan

print(Cfani)
print(Tfani)
print(Sigani)
```

```
[[-0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.7006358
  -1.30127159 -1.90190739 -2.50254319 -3.32841741 -3.92905321 -4.52968901
  -5.35556323 -5.95619902 -6.10635797 -6.10635797]
 [-0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.7006358
  -1.30127159 -1.90190739 -2.50254319 -3.32841741 -3.92905321 -4.52968901
  -5.35556323 -5.95619902 -6.10635797 -6.10635797]
 [        nan         nan -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.7006358
  -1.30127159 -1.90190739 -0.7006358  -1.52651002 -2.12714582 -2.72778161
  -3.55365583 -4.15429163 -4.75492743 -5.58080165]
 [        nan         nan -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.7006358
  -1.30127159 -1.90190739 -0.7006358  -1.52651002 -2.12714582 -2.72778161
  -3.55365583 -4.15429163 -4.75492743 -5.58080165]
 [        nan         nan         nan         nan         nan         nan
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.7006358  -1.52651002 -2.12714582 -2.72778161
  -3.55365583 -4.15429163 -4.75492743 -0.92587422]
 [        nan         nan         nan         nan         nan         nan
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
```

```
  -0.1        -0.1        -0.1        -0.1        -0.1        -0.1
  -0.1        -0.1        -0.1        -0.92587422]]
[[0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1
  0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1
  0.1     0.1     0.1     0.1     0.1     0.29    0.48    0.67    0.86
  1.12125 1.31125 1.50125 1.7625  1.9525  2.      2.     ]
 [0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1
  0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1
  0.1     0.1     0.1     0.1     0.1     0.29    0.48    0.67    0.86
  1.12125 1.31125 1.50125 1.7625  1.9525  2.      2.     ]
 [    nan     nan 0.1     0.1     0.1     0.1     0.1     0.1     0.1
  0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1
  0.1     0.1     0.1     0.1     0.1     0.29    0.48    0.67    0.29
  0.55125 0.74125 0.93125 1.1925  1.3825  1.5725  1.83375]
 [    nan     nan 0.1     0.1     0.1     0.1     0.1     0.1     0.1
  0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1
  0.1     0.1     0.1     0.1     0.1     0.29    0.48    0.67    0.29
  0.55125 0.74125 0.93125 1.1925  1.3825  1.5725  1.83375]
 [    nan     nan     nan     nan     nan     nan 0.1     0.1     0.1
  0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1
  0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.29
  0.55125 0.74125 0.93125 1.1925  1.3825  1.5725  0.36125]
 [    nan     nan     nan     nan     nan     nan 0.1     0.1     0.1
  0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1
  0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1     0.1
  0.1     0.1     0.1     0.1     0.1     0.1     0.36125]]
[[ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00 -2.37071738e-02 -2.37071738e-02
  -1.51863356e-01 -1.58455012e-01 -1.58455012e-01 -2.17113514e-01
  -2.50139704e-01 -2.50139704e-01 -3.35502123e-01 -3.95281863e-01
  -4.18935571e-01 -4.68881929e-01]
 [ 0.00000000e+00  0.00000000e+00  1.48887260e-01  1.48887260e-01
   1.48887260e-01  1.48887260e-01  1.48887260e-01  1.48887260e-01
   1.48887260e-01  1.48887260e-01  1.48887260e-01  1.48887260e-01
   1.48887260e-01  1.48887260e-01  1.48887260e-01  1.48887260e-01
   1.48887260e-01  1.48887260e-01  1.48887260e-01  1.48887260e-01
   1.48887260e-01  1.48887260e-01  1.71268984e-01  1.71268984e-01
   2.98099716e-01  1.48887260e-01  1.48887260e-01  1.48555897e-01
   1.48230705e-01  1.48230705e-01  1.47899343e-01  1.47574165e-01
   1.39541349e-01  1.22745290e-01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  2.23817241e-02  2.23817241e-02
   1.49212456e-01  2.77555756e-17  2.77555756e-17 -3.31362426e-04
  -6.56554315e-04 -6.56554315e-04 -9.87916741e-04 -1.31309461e-03
  -9.34591112e-03 -2.61419693e-02]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  2.47730000e-01  2.47730000e-01  2.47730000e-01
   2.47730000e-01  2.47730000e-01  2.47730000e-01  2.47730000e-01
   2.47730000e-01  2.47730000e-01  2.47730000e-01  2.47730000e-01
   2.47730000e-01  2.47730000e-01  2.47730000e-01  2.47730000e-01
   2.47730000e-01  2.47730000e-01  2.47730000e-01  2.47730000e-01
   2.47730000e-01  2.53671263e-01  2.53671263e-01  3.11667041e-01
   3.44042847e-01  3.44042847e-01  4.28742541e-01  4.87871925e-01
   2.47730000e-01  2.64084241e-01]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  5.94126308e-03  5.94126308e-03  6.39370409e-02
   9.63128469e-02  9.63128469e-02  1.81012541e-01  2.40141926e-01
   0.00000000e+00  1.63542416e-02]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
```

```
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   2.32089651e-02  7.27135065e-02]
 [ 0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00  0.00000000e+00  0.00000000e+00
   0.00000000e+00  0.00000000e+00]]
```

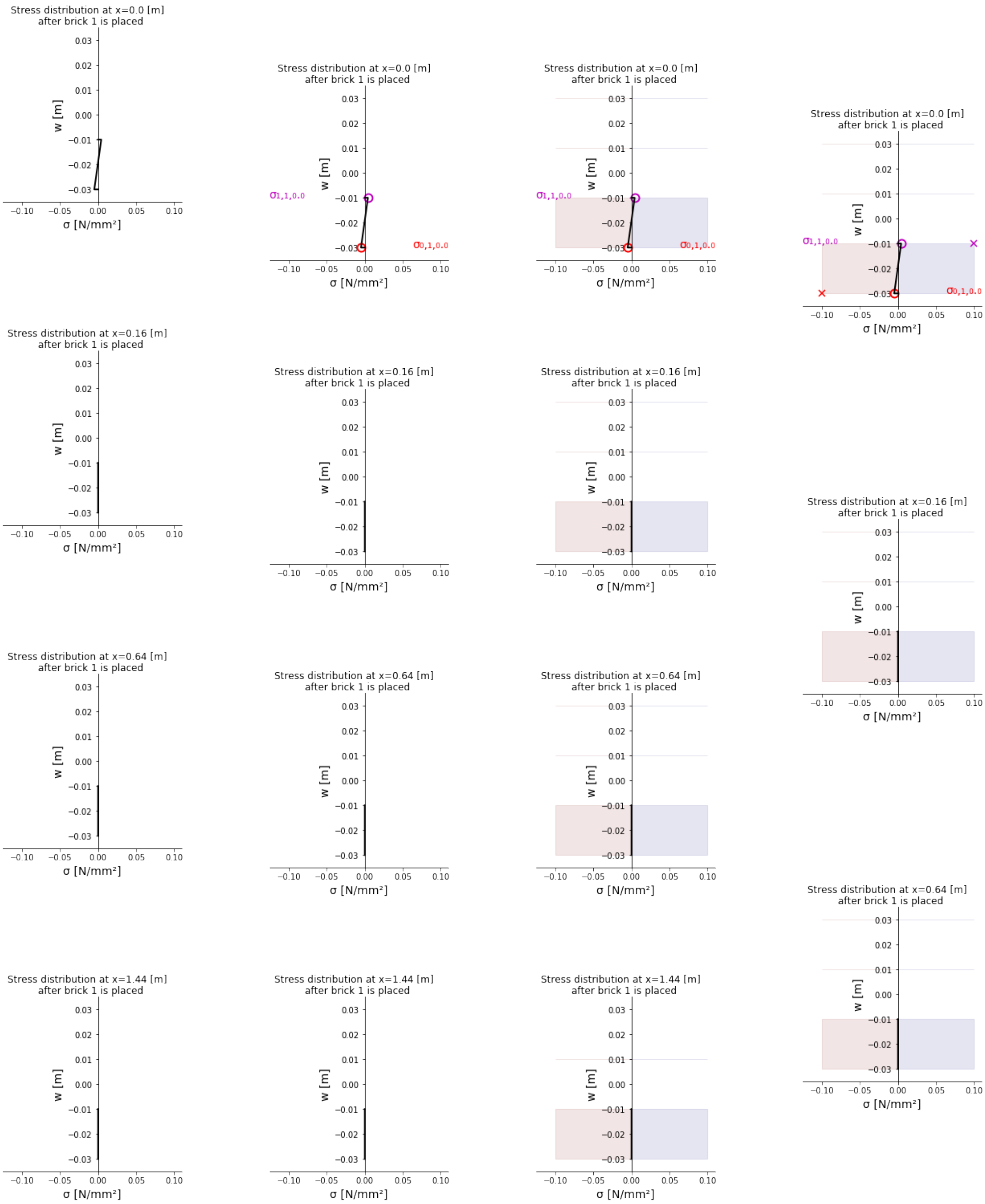
In [11]:
```
#
len(Cfani)
phasedC = Cfani
phasedT = Tfani
phasedS = Sigani
np.savetxt('C:/Users/Joris/Documents/Master Thesis quick access/PhasedC.txt',
           phasedC, delimiter = ';', fmt = '%s')
np.savetxt('C:/Users/Joris/Documents/Master Thesis quick access/PhasedT.txt',
           phasedT, delimiter = ';', fmt = '%s')
np.savetxt('C:/Users/Joris/Documents/Master Thesis quick access/PhasedS.txt',
           phasedS, delimiter = ';', fmt = '%s')
```

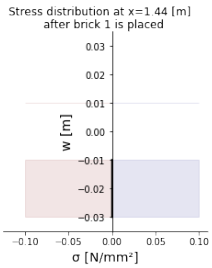
In [12]:
```
# posi stands for the x-linspace, plcd stands for bricks placed:
# posi=0 is at support, plcd=0 is 1 brick placed
xshift=4
runn = []

[runn.append([[0,0,0],[0,xshift*1**2,0],[0,xshift*2**2,0],
              [0,xshift*3**2,0]][ele]) for ele in range(4)]
[runn.append([[0,0,1],[0,xshift*1**2,1],[0,xshift*2**2,1],
              [0,xshift*3**2,1]][ele]) for ele in range(4)]
[runn.append([[0,0,1],[0,xshift*1**2,1],[0,xshift*2**2,1],
              [0,xshift*3**2,1]][ele]) for ele in range(4)]
[runn.append([[0,0,1],[0,xshift*1**2,1],[0,xshift*2**2,1],
              [0,xshift*3**2,1]][ele]) for ele in range(4)]
manu = [0,1,2,3,4,5,6,7,8,12,16,20,24,28,30]
if len(seq)>8:
    manu = np.linspace(0,8,9).tolist()
else:
    manu = np.linspace(0,len(seq)-1,len(seq)).tolist()
manu.extend([ele for ele in np.arange(0,100,4) if ele>manu[-1] and ele<len(seq)-3])
manu.extend([len(seq)-3,len(seq)-2])
manu = [int(ele) for ele in manu]
# manu = [1,2,3,4,5,6,7,8,12,16,20,24,28,30]
manul = len(manu)
for i in range(len(manu)*4):
    runn.append([manu[int(i/4)], xshift*int((i%4)**2), 1]) # plcd, posi, labelled
print(runn)
print()

for plc in range(len(runn)):
    posi = runn[plc][1]
    plcd = runn[plc][0]
    lbl = runn[plc][2]
    hh = [[-1.5*t,-1.5*t,-.5*t,-.5*t,.5*t,.5*t,1.5*t,1.5*t]
          for ele in range(len(Sigani[0]))]
    hh = [-1.5*t,-1.5*t,-.5*t,-.5*t,.5*t,.5*t,1.5*t,1.5*t]

    Cfan = np.array(Cstrres)[:,:,posi]
    Tfan = np.array(Tstrres)[:,:,posi]
    Sigan = np.array(Sigres)[:,:,posi]

    length = [sum([1 if ele==wth else 0 for idx,ele in enumerate(ll[0:plcd+1])])-1
              for wth in [0,1,2]]
    start = [y0[length[0]]<x[posi], y1[length[1]]<x[posi], y2[length[2]]<x[posi]]
    start = [True if length[idx]==-1 else ele for idx,ele in enumerate(start)]
    start.append(start[-1])

#     Inter = np.array([[0 if lst<start[idx] else ele
#                        for idx,ele in enumerate(inter[:,posi+1])]
#                       for lst,Lst in enumerate(range(len(seq)))])
    Inter = np.array([[0 if start[idx] else ele
                       for idx,ele in enumerate(inter[:,posi+1])]
                      for lst,Lst in enumerate(range(len(seq)))])

    Sigani = np.array([np.zeros(len(seq)),#0
                       Sigan[:,0],#1
                       Sigan[:,1]+Inter[:,1],#2
```

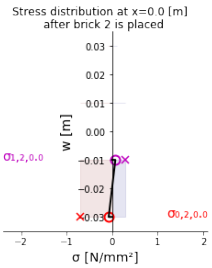
```
                    Sigan[:,1],#3
                    Sigan[:,2]+Inter[:,2],#4
                    Sigan[:,2],#5
                    Sigan[:,3],#6
                    np.zeros(len(seq))])#7
    Cfani = np.array([Cfan[:,0],Cfan[:,0],Cfan[:,1],Cfan[:,1],Cfan[:,2],Cfan[:,3]])
    Tfani = np.array([Tfan[:,0],Tfan[:,0],Tfan[:,1],Tfan[:,1],Tfan[:,2],Tfan[:,3]])
    Sigani[5] = [0 if Inter[:,2][idx]==0 else ele
                 for idx,ele in enumerate(Sigani[5])]
    Sigani[3] = [0 if Inter[:,1][idx]==0 else ele
                 for idx,ele in enumerate(Sigani[3])]

    for i in range(len(Sigani)-2-1,0,-1): # i is per interface
        i +=2
        for j in range(len(Sigani[i])): # j is per plcd
            if Sigani[i-1][j]==0 and Sigani[i-1][j-1]!=0:
                Sigani[i-1][j]+=10**-6
            if i>1 and i<7 and Sigani[i][j]==0:
                Cfani[i-1][j] = np.nan
                Tfani[i-1][j] = np.nan
            if Sigani[i-1][j]==0:# and Sigani[i-1][j-1]==0:
                Sigani[i][j] = np.nan
    if plcd==2:
        Cfani[2:4,plcd] = fm
        Tfani[2:4,plcd] = fti

    fig = plt.figure()
    fig.set_figwidth(4),fig.set_figheight(4)
    ax = fig.add_subplot(1, 1, 1)
    if plc>=4*4 or len(runn)/4==len(manu):
        ax.set_xlim(-2.4,2.1)
    else:
        ax.set_xlim(-.1257,.11)
    ax.set_ylim(-1.75*t,1.75*t)
    ax.spines['left'].set_position('zero')
    ax.spines['right'].set_color('none')
    ax.yaxis.tick_left()
    ax.spines['bottom'].set_position(('axes',0))
    ax.spines['top'].set_color('none')
    ax.xaxis.tick_bottom()
    ax.set_xlabel('\u03C3 [N/mmš]', fontsize=14)
    ax.set_ylabel('w [m]', fontsize=14)
    fig.suptitle('Stress distribution at x=%s [m] \n after brick %s is placed' % (x[posi],plcd+1),fontsize=12)

    if lbl==1:
        lyrs = np.count_nonzero([ele for ele in Sigani[:,plcd]
                                 if (not np.isnan(ele))])
        if lyrs==2:
            sig = [1,2,4,6]
        elif lyrs==4:
            sig = [1,3,4,6]
        else:
            sig = [1,3,5,6]

        clr = ['r','m','b','g']
        sig = [ele for ele in sig if (Sigani[:,plcd][ele]!=0)*(
            not np.isnan(Sigani[:,plcd][ele]))]
        for i in range(len(sig)):
            text= "\u03C3%s" % (sub('%s,%s,%s' % (i,plcd+1,x[posi]))) #text
            labpo = [-np.sign(Sigani[sig[i],plcd])/2+0.5,hh[sig[i]]/sum(
                np.abs(ax.get_ylim()))+0.5]
            ax.annotate(text, xy=(labpo[0],labpo[1]), xycoords='axes fraction',
                        color=clr[i], ha=['left','right'][int(labpo[0])],
                        fontsize=14)
            plt.plot(Sigani[sig[i],plcd],hh[sig[i]], 'o', ms=10, mfc='none',
                     mew=2,  mec=clr[i])
            if plc>=3*4 or len(runn)/4==len(manu):
                if Sigani[sig[i],plcd]<0:
                    plt.plot(Cfani[sig[i]-1,plcd],hh[sig[i]], 'x', ms=8, mfc='none',
                             mew=1.5,  mec=clr[i])
                if Sigani[sig[i],plcd]<0 and Cfani[sig[i]-1,plcd]<-2.4:
                    plt.arrow(-2.4+.4, hh[sig[i]], -.4, 0, length_includes_head=True,
                              shape='full',width=.0004, head_width = .0015,
                              head_length = .15, ec = 'none', fc = clr[i])
                if Sigani[sig[i],plcd]>=0:
                    plt.plot(Tfani[sig[i]-1,plcd],hh[sig[i]], 'x', ms=8, mfc='none',
                             mew=1.5,  mec=clr[i])

#     print ('Sig:',Sigani[:,plcd])
#     print('Cf',Cfani[:,plcd])
#     print('Tf',Tfani[:,plcd])
    ax.plot(Sigani[:,plcd],hh,color='k',lw=2)
    x_fill = np.zeros(len(hh))
    if plc>=2*4 or len(runn)/4==len(manu):
```

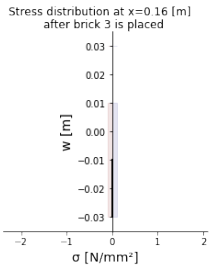
```
            ax.fill_betweenx(hh[1:7],Cfani[:,plcd],x_fill[1:7],color='maroon',alpha=.1)
            ax.fill_betweenx(hh[1:7],Tfani[:,plcd],x_fill[1:7],color='navy',alpha=.1)
#       if plc==len(runn)-2:
#           fig.set_figwidth(12),fig.set_figheight(4)
#           saveplt = ax
        plt.show();
```

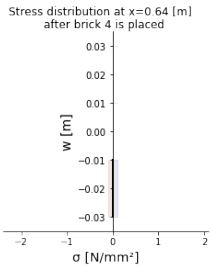
```
[[0, 0, 0], [0, 4, 0], [0, 16, 0], [0, 36, 0], [0, 0, 1], [0, 4, 1], [0, 16, 1], [0, 36, 1], [0, 0, 1], [0, 4, 1],
 [0, 16, 1], [0, 36, 1], [0, 0, 1], [0, 4, 1], [0, 16, 1], [0, 36, 1], [0, 0, 1], [0, 4, 1], [0, 16, 1], [0, 36, 1],
 [1, 0, 1], [1, 4, 1], [1, 16, 1], [1, 36, 1], [2, 0, 1], [2, 4, 1], [2, 16, 1], [2, 36, 1], [3, 0, 1], [3, 4, 1],
 [3, 16, 1], [3, 36, 1], [4, 0, 1], [4, 4, 1], [4, 16, 1], [4, 36, 1], [5, 0, 1], [5, 4, 1], [5, 16, 1], [5, 36, 1],
 [6, 0, 1], [6, 4, 1], [6, 16, 1], [6, 36, 1], [7, 0, 1], [7, 4, 1], [7, 16, 1], [7, 36, 1], [8, 0, 1], [8, 4, 1],
 [8, 16, 1], [8, 36, 1], [12, 0, 1], [12, 4, 1], [12, 16, 1], [12, 36, 1], [16, 0, 1], [16, 4, 1], [16, 16, 1],
 [16, 36, 1], [20, 0, 1], [20, 4, 1], [20, 16, 1], [20, 36, 1], [24, 0, 1], [24, 4, 1], [24, 16, 1], [24, 36, 1],
 [28, 0, 1], [28, 4, 1], [28, 16, 1], [28, 36, 1], [31, 0, 1], [31, 4, 1], [31, 16, 1], [31, 36, 1], [32, 0, 1],
 [32, 4, 1], [32, 16, 1], [32, 36, 1]]
```

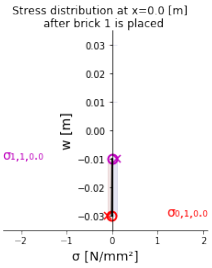


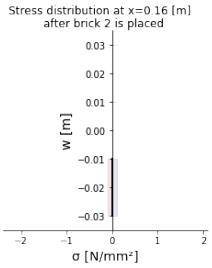

Stress distribution at x=1.44 [m]
after brick 1 is placed
w [m]
σ [N/mm²]

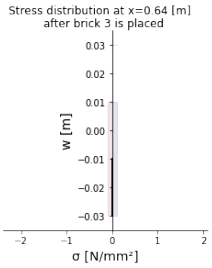

Stress distribution at x=0.0 [m]
after brick 2 is placed
σ1,2,0.0
σ0,2,0.0
w [m]
σ [N/mm²]

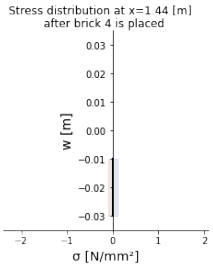

Stress distribution at x=0.16 [m]
after brick 3 is placed
w [m]
σ [N/mm²]

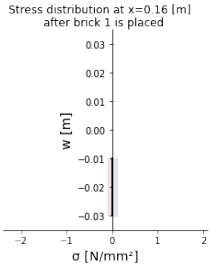

Stress distribution at x=0.64 [m]
after brick 4 is placed
w [m]
σ [N/mm²]

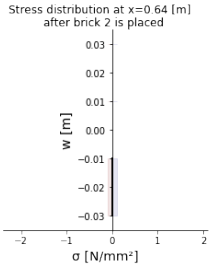

Stress distribution at x=0.0 [m]
after brick 1 is placed
σ1,1,0.0
σ0,1,0.0
w [m]
σ [N/mm²]

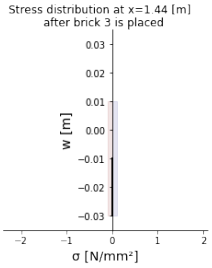

Stress distribution at x=0.16 [m]
after brick 2 is placed
w [m]
σ [N/mm²]

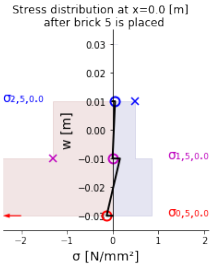

Stress distribution at x=0.64 [m]
after brick 3 is placed
w [m]
σ [N/mm²]

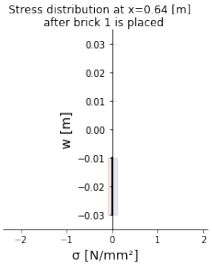

Stress distribution at x=1.44 [m]
after brick 4 is placed
w [m]
σ [N/mm²]

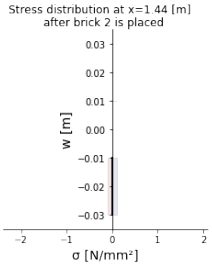

Stress distribution at x=0.16 [m]
after brick 1 is placed
w [m]
σ [N/mm²]

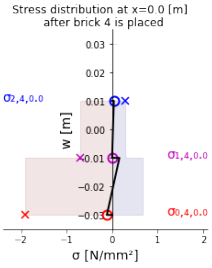

Stress distribution at x=0.64 [m]
after brick 2 is placed
w [m]
σ [N/mm²]

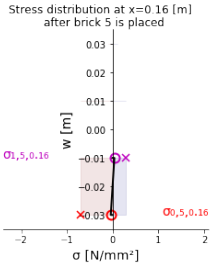

Stress distribution at x=1.44 [m]
after brick 3 is placed
w [m]
σ [N/mm²]

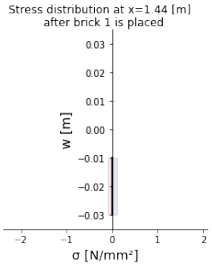

Stress distribution at x=0.0 [m]
after brick 5 is placed
σ2,5,0.0
σ1,5,0.0
σ0,5,0.0
w [m]
σ [N/mm²]

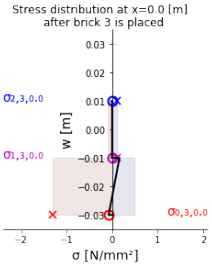

Stress distribution at x=0.64 [m]
after brick 1 is placed
w [m]
σ [N/mm²]

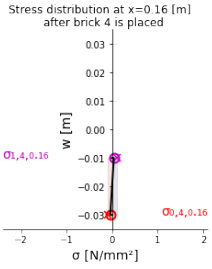

Stress distribution at x=1.44 [m]
after brick 2 is placed
w [m]
σ [N/mm²]

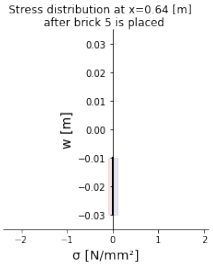

Stress distribution at x=0.0 [m]
after brick 4 is placed
σ2,4,0.0
σ1,4,0.0
σ0,4,0.0
w [m]
σ [N/mm²]

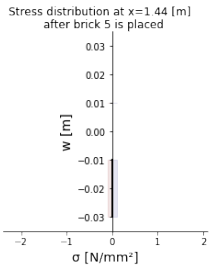

Stress distribution at x=0.16 [m]
after brick 5 is placed
σ1,5,0.16
σ0,5,0.16
w [m]
σ [N/mm²]

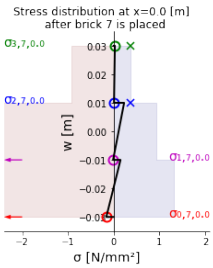

Stress distribution at x=1.44 [m]
after brick 1 is placed
w [m]
σ [N/mm²]

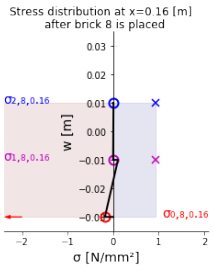

Stress distribution at x=0.0 [m]
after brick 3 is placed
σ2,3,0.0
σ1,3,0.0
σ0,3,0.0
w [m]
σ [N/mm²]

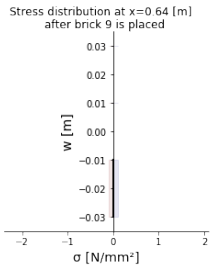

Stress distribution at x=0.16 [m]
after brick 4 is placed
σ1,4,0.16
σ0,4,0.16
w [m]
σ [N/mm²]

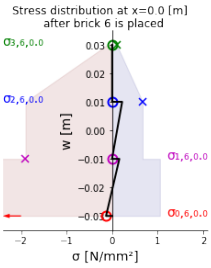

Stress distribution at x=0.64 [m]
after brick 5 is placed
w [m]
σ [N/mm²]

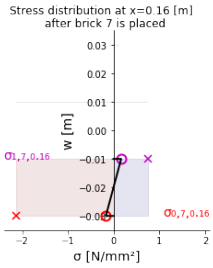

Stress distribution at x=1.44 [m]
after brick 5 is placed
w [m]
σ [N/mm²]

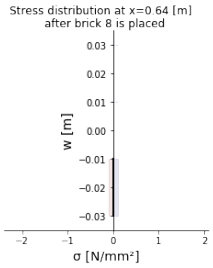

Stress distribution at x=0.0 [m]
after brick 7 is placed
σ3,7,0.0
σ2,7,0.0
σ1,7,0.0
σ0,7,0.0
w [m]
σ [N/mm²]

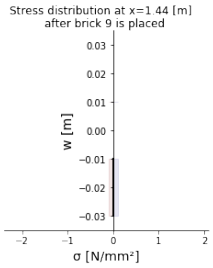

Stress distribution at x=0.16 [m]
after brick 8 is placed
σ2,8,0.16
σ1,8,0.16
σ0,8,0.16
w [m]
σ [N/mm²]

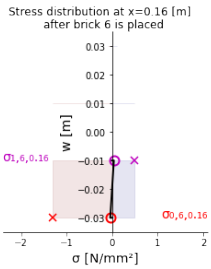

Stress distribution at x=0.64 [m]
after brick 9 is placed
w [m]
σ [N/mm²]

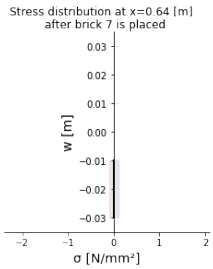

Stress distribution at x=0.0 [m]
after brick 6 is placed
σ3,6,0.0
σ2,6,0.0
σ1,6,0.0
σ0,6,0.0
w [m]
σ [N/mm²]

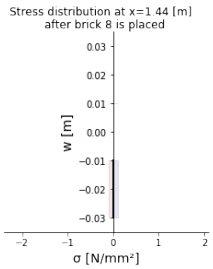

Stress distribution at x=0.16 [m]
after brick 7 is placed
σ1,7,0.16
σ0,7,0.16
w [m]
σ [N/mm²]

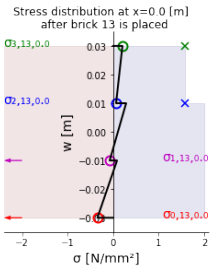

Stress distribution at x=0.64 [m]
after brick 8 is placed
w [m]
σ [N/mm²]

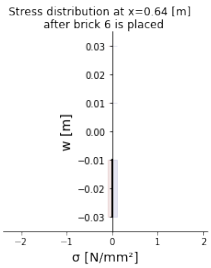

Stress distribution at x=1.44 [m]
after brick 9 is placed
w [m]
σ [N/mm²]

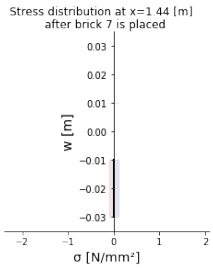

Stress distribution at x=0.16 [m]
after brick 6 is placed
σ1,6,0.16
σ0,6,0.16
w [m]
σ [N/mm²]

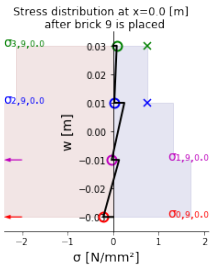

Stress distribution at x=0.64 [m]
after brick 7 is placed
w [m]
σ [N/mm²]

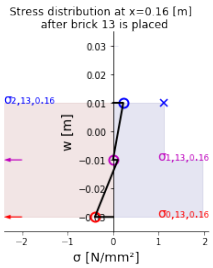

Stress distribution at x=1.44 [m]
after brick 8 is placed
w [m]
σ [N/mm²]

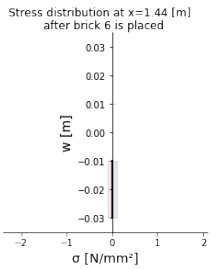

Stress distribution at x=0.0 [m]
after brick 13 is placed
σ3,13,0.0
σ2,13,0.0
σ1,13,0.0
σ0,13,0.0
w [m]
σ [N/mm²]

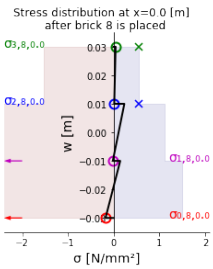

Stress distribution at x=0.64 [m]
after brick 6 is placed
w [m]
σ [N/mm²]

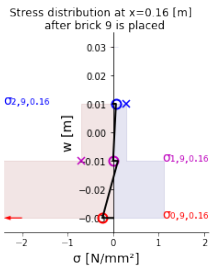

Stress distribution at x=1.44 [m]
after brick 7 is placed
w [m]
σ [N/mm²]

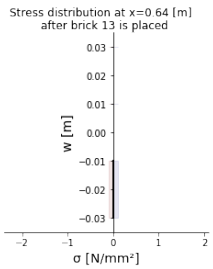

Stress distribution at x=0.0 [m]
after brick 9 is placed
σ3,9,0.0
σ2,9,0.0
σ1,9,0.0
σ0,9,0.0
w [m]
σ [N/mm²]

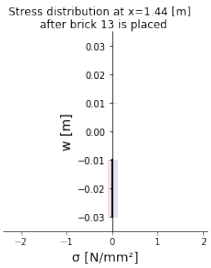

Stress distribution at x=0.16 [m]
after brick 13 is placed
σ2,13,0.16
σ1,13,0.16
σ0,13,0.16
w [m]
σ [N/mm²]

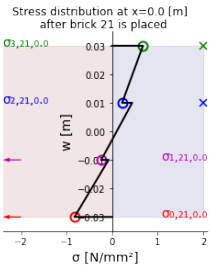

Stress distribution at x=1.44 [m]
after brick 6 is placed
w [m]
σ [N/mm²]

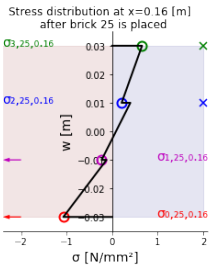

Stress distribution at x=0.0 [m]
after brick 8 is placed
σ3,8,0.0
σ2,8,0.0
σ1,8,0.0
σ0,8,0.0
w [m]
σ [N/mm²]

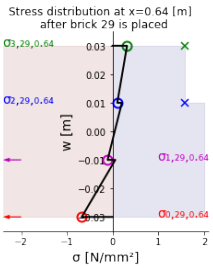

Stress distribution at x=0.16 [m]
after brick 9 is placed
σ2,9,0.16
σ1,9,0.16
σ0,9,0.16
w [m]
σ [N/mm²]

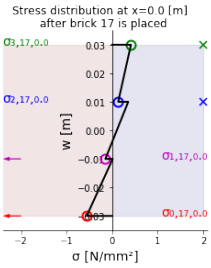

Stress distribution at x=0.64 [m]
after brick 13 is placed
w [m]
σ [N/mm²]

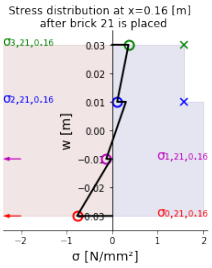

Stress distribution at x=1.44 [m]
after brick 13 is placed
w [m]
σ [N/mm²]

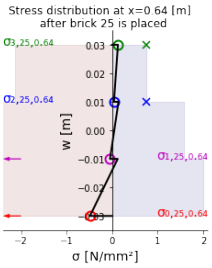

Stress distribution at x=0.0 [m]
after brick 21 is placed
σ3,21,0.0
σ2,21,0.0
σ1,21,0.0
σ0,21,0.0
w [m]
σ [N/mm²]

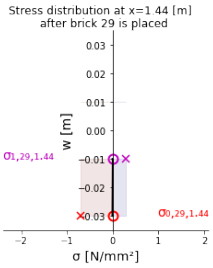

Stress distribution at x=0.16 [m]
after brick 25 is placed
σ3,25,0.16
σ2,25,0.16
σ1,25,0.16
σ0,25,0.16
w [m]
σ [N/mm²]

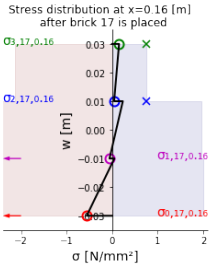

Stress distribution at x=0.64 [m]
after brick 29 is placed
σ3,29,0.64
σ2,29,0.64
σ1,29,0.64
σ0,29,0.64
w [m]
σ [N/mm²]

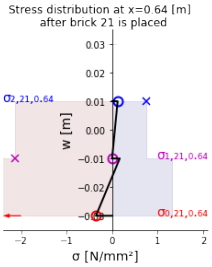

Stress distribution at x=0.0 [m]
after brick 17 is placed
σ3,17,0.0
σ2,17,0.0
σ1,17,0.0
σ0,17,0.0
w [m]
σ [N/mm²]

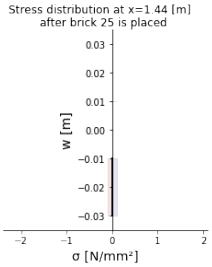

Stress distribution at x=0.16 [m]
after brick 21 is placed
σ3,21,0.16
σ2,21,0.16
σ1,21,0.16
σ0,21,0.16
w [m]
σ [N/mm²]

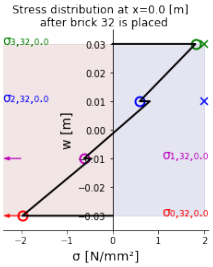

Stress distribution at x=0.64 [m]
after brick 25 is placed
σ3,25,0.64
σ2,25,0.64
σ1,25,0.64
σ0,25,0.64
w [m]
σ [N/mm²]

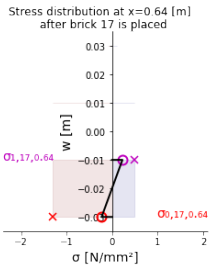

Stress distribution at x=1.44 [m]
after brick 29 is placed
σ1,29,1.44
σ0,29,1.44
w [m]
σ [N/mm²]

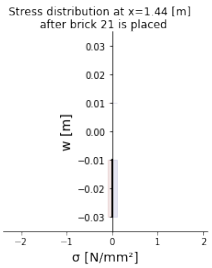

Stress distribution at x=0.16 [m]
after brick 17 is placed
σ3,17,0.16
σ2,17,0.16
σ1,17,0.16
σ0,17,0.16
w [m]
σ [N/mm²]

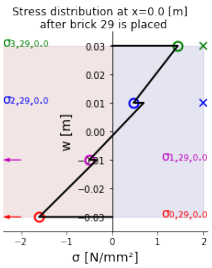

Stress distribution at x=0.64 [m]
after brick 21 is placed
σ2,21,0.64
σ1,21,0.64
σ0,21,0.64
w [m]
σ [N/mm²]

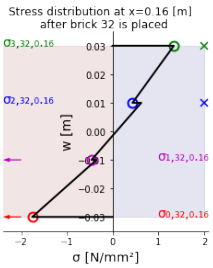

Stress distribution at x=1.44 [m]
after brick 25 is placed
w [m]
σ [N/mm²]

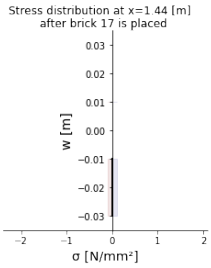

Stress distribution at x=0.0 [m]
after brick 32 is placed
σ3,32,0.0
σ2,32,0.0
σ1,32,0.0
σ0,32,0.0
w [m]
σ [N/mm²]

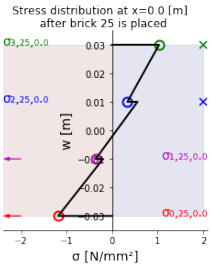

Stress distribution at x=0.64 [m]
after brick 17 is placed
σ1,17,0.64
σ0,17,0.64
w [m]
σ [N/mm²]

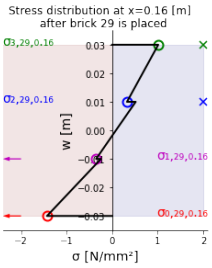

Stress distribution at x=1.44 [m]
after brick 21 is placed
w [m]
σ [N/mm²]

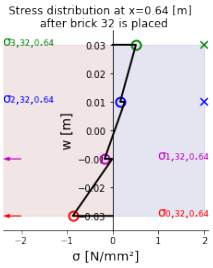

Stress distribution at x=0.0 [m]
after brick 29 is placed
σ3,29,0.0
σ2,29,0.0
σ1,29,0.0
σ0,29,0.0
w [m]
σ [N/mm²]

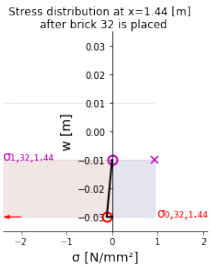

Stress distribution at x=0.16 [m]
after brick 32 is placed
σ3,32,0.16
σ2,32,0.16
σ1,32,0.16
σ0,32,0.16
w [m]
σ [N/mm²]

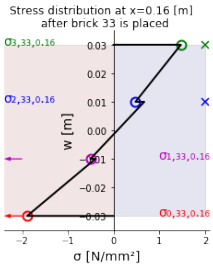

Stress distribution at x=1.44 [m]
after brick 17 is placed
w [m]
σ [N/mm²]

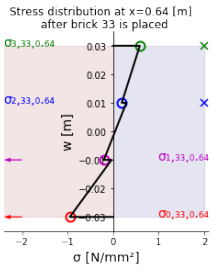

Stress distribution at x=0.0 [m]
after brick 25 is placed
σ3,25,0.0
σ2,25,0.0
σ1,25,0.0
σ0,25,0.0
w [m]
σ [N/mm²]

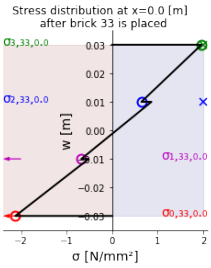

Stress distribution at x=0.16 [m]
after brick 29 is placed
σ3,29,0.16
σ2,29,0.16
σ1,29,0.16
σ0,29,0.16
w [m]
σ [N/mm²]

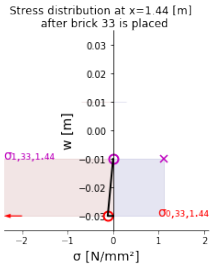

Stress distribution at x=0.64 [m]
after brick 32 is placed
σ3,32,0.64
σ2,32,0.64
σ1,32,0.64
σ0,32,0.64
w [m]
σ [N/mm²]





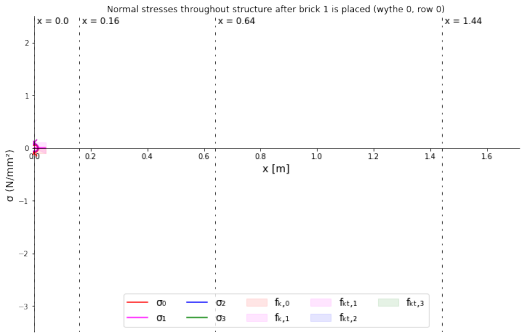


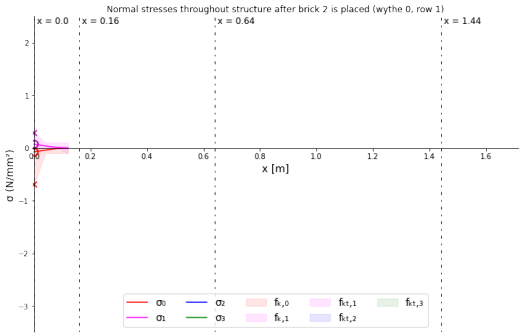


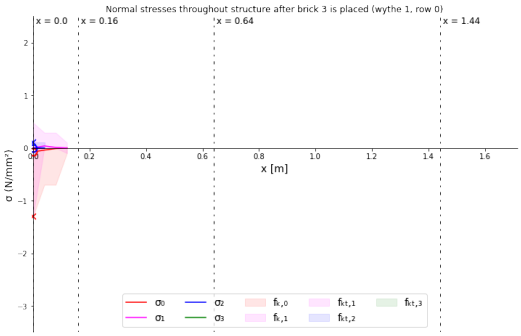


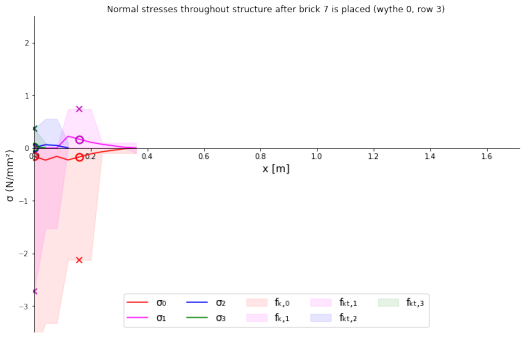
In [13]:
```
#
n0,n1,n2 = -2,-2,-2
id0,id1,id2 = 0,0,0

C0res=np.array([[[0 if Sigres[brk][lay][idx]>0 else ele
                  for idx,ele in enumerate(Cstrres[brk][lay])]
                 for lay,Lay in enumerate(Cstrres[brk])]
                for brk,Brk in enumerate(Cstrres)])
T0res=np.array([[[0 if Sigres[brk][lay][idx]<0 else ele
                  for idx,ele in enumerate(Tstrres[brk][lay])]
                 for lay,Lay in enumerate(Tstrres[brk])]
                for brk,Brk in enumerate(Tstrres)])

for i in range(len(manu)):
    posi = np.array(runn)[:4,1]
    i,add=manu[i],i
    n0,n1,n2 = [sum([1 if ele==wth else 0 for idx,ele in enumerate(ll[0:i+1])])-1
                for wth in [0,1,2]]
    id0,id1 = i0.index(n0+1) if n0>-1 else -1, i1.index(n1+1) if n1>-1 else -1
    id2 = i2.index(n2+1) if n2>-1 else -1

    fig = plt.figure()
    fig.set_figwidth(12),fig.set_figheight(8)
    ax = fig.add_subplot(1, 1, 1)
#
    ax.spines['left'].set_position('zero')
    ax.spines['right'].set_color('none')
    ax.yaxis.tick_left()

    ax.spines['bottom'].set_position('zero')

    ax.spines['top'].set_color('none')
    ax.xaxis.tick_bottom()

    xx = x

    ax.plot(xx[:id0+1],Sigres[i][0][:id0+1], label = '\u03C3%s' % (sub('0')),
            color='red')
    ax.plot(xx[:id0+1],Sigres[i][1][:id0+1], label = '\u03C3%s' % (sub('1')),
            color='magenta')
    ax.plot(xx[:id1+1],Sigres[i][2][:id1+1], label = '\u03C3%s' % (sub('2')),
            color='blue')
    ax.plot(xx[:id2+1],Sigres[i][3][:id2+1], label = '\u03C3%s' % (sub('3')),
            color='green')

    alp = .1
    ax.fill_between(xx[:id0+1],C0res[i][0][:id0+1], color='red',alpha=alp,
                    label = 'f%s' % (sub('k,0')))
    ax.fill_between(xx[:id0+1],C0res[i][1][:id0+1], color='magenta',alpha=alp,
                    label = 'f%s' % (sub('k,1')))
    ax.fill_between(xx[:id0+1],T0res[i][1][:id0+1], color='magenta',alpha=alp,
                    label = 'f%s' % (sub('kt,1')))
    ax.fill_between(xx[:id1+1],Tstrres[i][2][:id1+1], color='blue',alpha=alp,
                    label = 'f%s' % (sub('kt,2')))
```

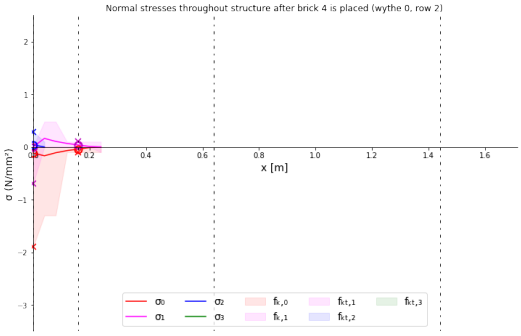
```
        ax.fill_between(xx[:id2+1],Tstrres[i][3][:id2+1], color='green',alpha=alp,
                        label = 'f%s' % (sub('kt,3')))

        ax.set_xlabel('x [m]', fontsize=14)
        ax.set_ylabel('\u03C3 (N/mmš)', fontsize=14)
        ax.title.set_text('Normal stresses throughout structure after brick %s is placed (wythe %s, row %s)
        ' % (i+1,ll[i],nn[i]))

        ax.legend(loc=8, fontsize=14,ncol=5)
        ax.set_ylim([-3.5,2.5]), ax.set_xlim([-0.005,1.715])

        hd_wd = .008*sum(np.abs(ax.get_xlim()))
        hd_ln = 8*hd_wd
        for ar in range(len(posi)):

            if np.sign(Sigres[i][0][posi[ar]]) != 0:
                ax.plot(xx[posi[ar]], Sigres[i][0][posi[ar]], 'o', ms=10, mfc='none',
                        mew=2,  mec='r')
                ax.plot(xx[posi[ar]], COres[i][0][posi[ar]], 'x', ms=8, mfc='none',
                        mew=1.5,  mec='r')
    # #
            if np.sign(Sigres[i][1][posi[ar]]) != 0:
                ax.plot(xx[posi[ar]],Sigres[i][1][posi[ar]], 'o', ms=10, mfc='none',
                        mew=2,  mec='m')
                if Sigres[i][1][posi[ar]]<0:
                          ax.plot(xx[posi[ar]], COres[i][1][posi[ar]], 'x', ms=8,
                                  mfc='none', mew=1.5,  mec='m')
                if Sigres[i][1][posi[ar]]>=0:
                          ax.plot(xx[posi[ar]], TOres[i][1][posi[ar]], 'x', ms=8,
                                  mfc='none', mew=1.5,  mec='m')
    # #
            if np.sign(Sigres[i][2][posi[ar]]) != 0:
                ax.plot(xx[posi[ar]],Sigres[i][2][posi[ar]], 'o', ms=10, mfc='none',
                        mew=2,  mec='b')
                ax.plot(xx[posi[ar]], Tstrres[i][2][posi[ar]], 'x', ms=8, mfc='none',
                        mew=1.5,  mec='b')
    # #
            if np.sign(Sigres[i][3][posi[ar]]) != 0:
                ax.plot(xx[posi[ar]],Sigres[i][3][posi[ar]], 'o', ms=10, mfc='none',
                        mew=2,  mec='g')
                ax.plot(xx[posi[ar]], Tstrres[i][3][posi[ar]], 'x', ms=8, mfc='none',
                        mew=1.5,  mec='g')

            if i<manu[5] or i==manu[-1]:
                          ax.plot([xx[posi[ar]],xx[posi[ar]]],ax.get_ylim(),
                                  ls=(0,(2,7,4,7)),c='k',lw=1)
            if i<manu[3] or i==manu[-1]:
                          ax.annotate(" x = %s" % (x[posi[ar]]),
                                      xy=(xx[posi[ar]],ax.get_ylim()[1]),  color='k',
                                      fontsize=12, va='top')

        plt.show()
```

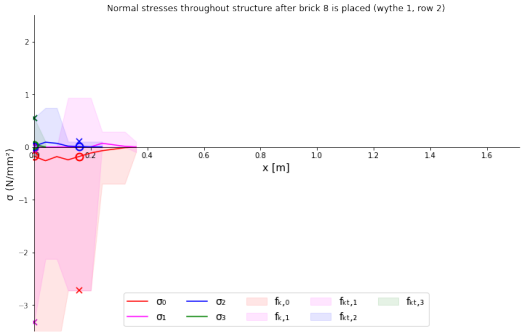


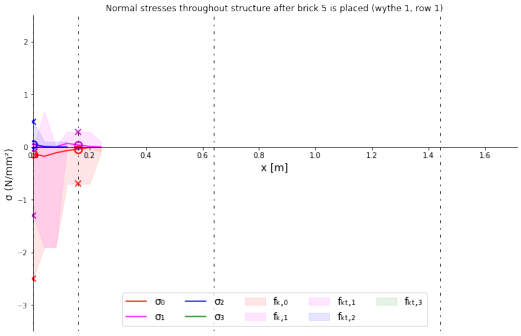


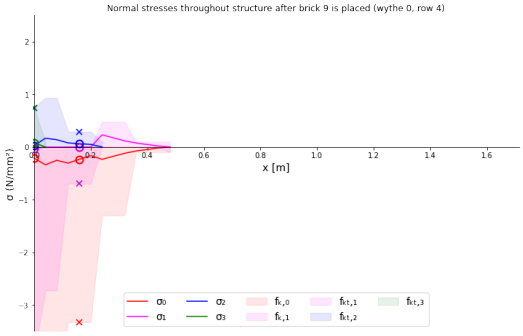

Normal stresses throughout structure after brick 3 is placed (wythe 1, row 0)
x = 0.0
x = 0.16
x = 0.64
x = 1.44
x [m]
σ (N/mm²)
σ0
σ1
σ2
σ3
fk,0
fk,1
fkt,1
fkt,2
fkt,3

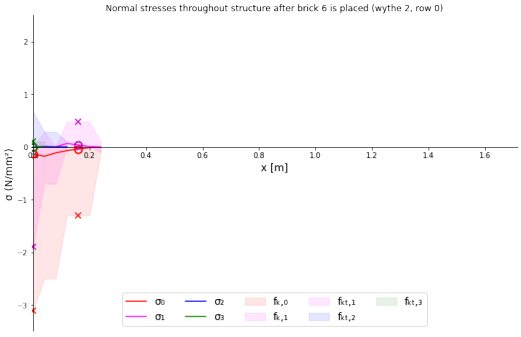

Normal stresses throughout structure after brick 7 is placed (wythe 0, row 3)
x [m]
σ (N/mm²)
σ0
σ1
σ2
σ3
fk,0
fk,1
fkt,1
fkt,2
fkt,3

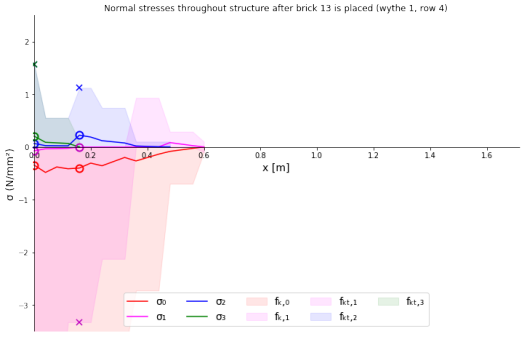

Normal stresses throughout structure after brick 4 is placed (wythe 0, row 2)
x [m]
σ (N/mm²)
σ0
σ1
σ2
σ3
fk,0
fk,1
fkt,1
fkt,2
fkt,3

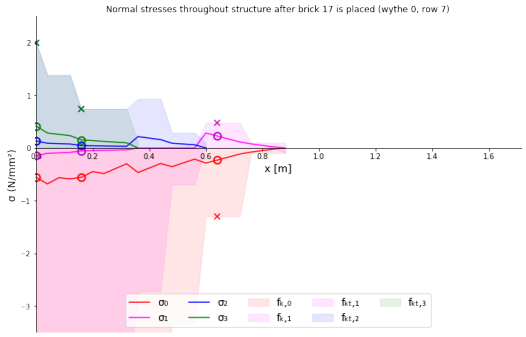

Normal stresses throughout structure after brick 8 is placed (wythe 1, row 2)
x [m]
σ (N/mm²)
σ0
σ1
σ2
σ3
fk,0
fk,1
fkt,1
fkt,2
fkt,3

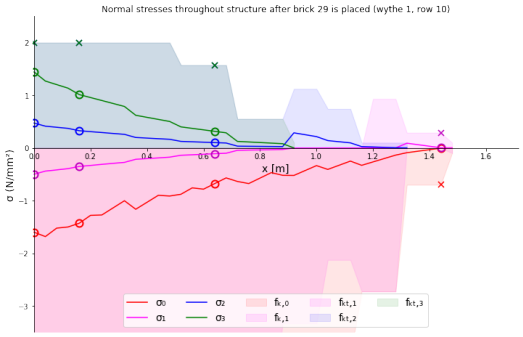

Normal stresses throughout structure after brick 5 is placed (wythe 1, row 1)
x [m]
σ (N/mm²)
σ0
σ1
σ2
σ3
fk,0
fk,1
fkt,1
fkt,2
fkt,3

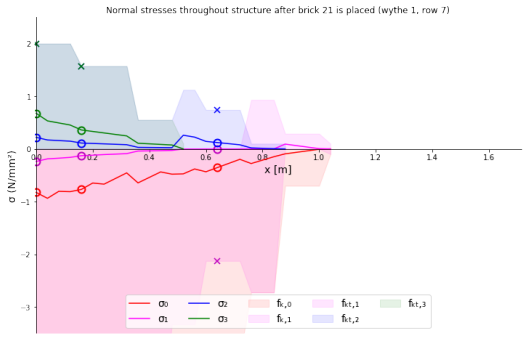

Normal stresses throughout structure after brick 9 is placed (wythe 0, row 4)
x [m]
σ (N/mm²)
σ0
σ1
σ2
σ3
fk,0
fk,1
fkt,1
fkt,2
fkt,3

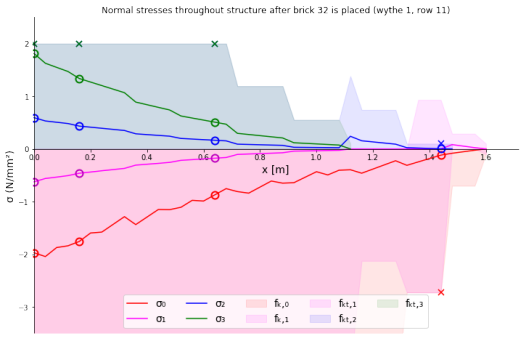

Normal stresses throughout structure after brick 6 is placed (wythe 2, row 0)
x [m]
σ (N/mm²)
σ0
σ1
σ2
σ3
fk,0
fk,1
fkt,1
fkt,2
fkt,3

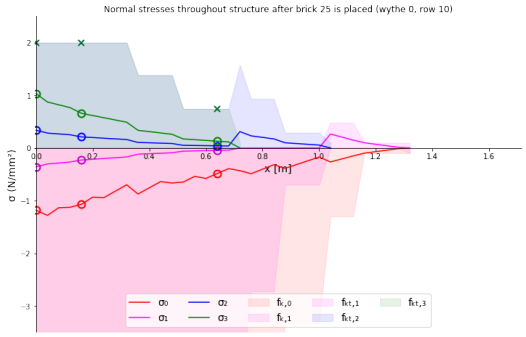

Normal stresses throughout structure after brick 13 is placed (wythe 1, row 4)
x [m]
σ (N/mm²)
σ0
σ1
σ2
σ3
fk,0
fk,1
fkt,1
fkt,2
fkt,3

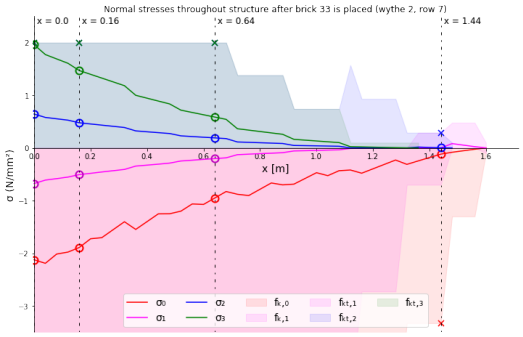










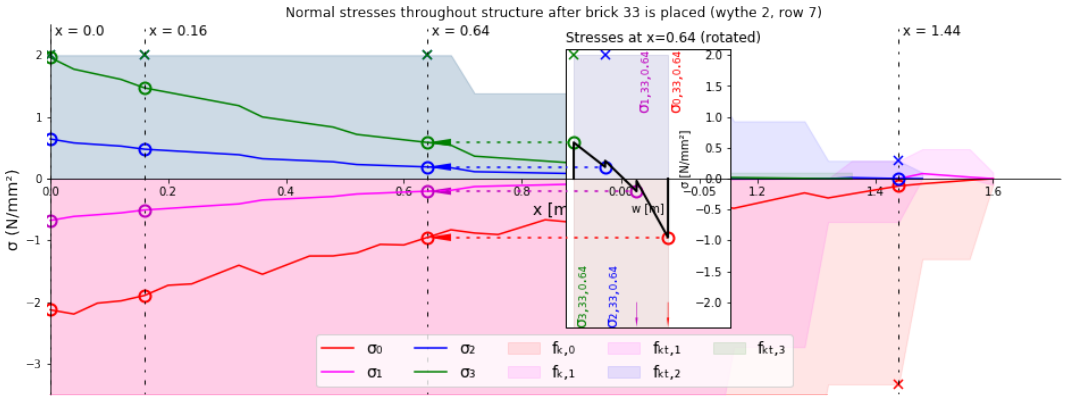



```
In [14]: #
         from matplotlib.transforms import Affine2D
         import mpl_toolkits.axisartist.floating_axes as floating_axes
         posi = np.array(runn)[:4,1]
         fig  = plt.figure(figsize=(16,6))
         ax1 = fig.add_subplot(1,1,1)

         if True:
             i = manu[-1]
             n0,n1,n2 = [sum([1 if ele==wth else 0 for idx,ele in enumerate(ll[0:i+1])])-1
                         for wth in [0,1,2]]
             id0,id1 = i0.index(n0+1) if n0>-1 else -1, i1.index(n1+1) if n1>-1 else -1
             id2 = i2.index(n2+1) if n2>-1 else -1
             ax1.spines['left'].set_position('zero')
             ax1.spines['right'].set_color('none')
             ax1.yaxis.tick_left()

             ax1.spines['bottom'].set_position('zero')

             ax1.spines['top'].set_color('none')
             ax1.xaxis.tick_bottom()

             xx = x
```

```
        ax1.plot(xx[:id0+1],Sigres[i][0][:id0+1], label = '\u03C3%s' % (sub('0')),
                 color='red')
        ax1.plot(xx[:id0+1],Sigres[i][1][:id0+1], label = '\u03C3%s' % (sub('1')),
                 color='magenta')
        ax1.plot(xx[:id1+1],Sigres[i][2][:id1+1], label = '\u03C3%s' % (sub('2')),
                 color='blue')
        ax1.plot(xx[:id2+1],Sigres[i][3][:id2+1], label = '\u03C3%s' % (sub('3')),
                 color='green')

        alp = .1
        ax1.fill_between(xx[:id0+1],C0res[i][0][:id0+1], color='red',alpha=alp,
                         label = 'f%s' % (sub('k,0')))
        ax1.fill_between(xx[:id0+1],C0res[i][1][:id0+1], color='magenta',alpha=alp,
                         label = 'f%s' % (sub('k,1')))
        ax1.fill_between(xx[:id0+1],T0res[i][1][:id0+1], color='magenta',alpha=alp,
                         label = 'f%s' % (sub('kt,1')))
        ax1.fill_between(xx[:id1+1],Tstrres[i][2][:id1+1], color='blue',alpha=alp,
                         label = 'f%s' % (sub('kt,2')))
        ax1.fill_between(xx[:id2+1],Tstrres[i][3][:id2+1], color='green',alpha=alp,
                         label = 'f%s' % (sub('kt,3')))

        ax1.set_xlabel('x [m]', fontsize=14)
        ax1.set_ylabel('\u03C3 (N/mmš)', fontsize=14)
        ax1.title.set_text('Normal stresses throughout structure after brick %s is placed (wythe %s, row %s)
        ' % (i+1,ll[i],nn[i]))

        ax1.legend(loc=8, fontsize=14,ncol=5)
        ax1.set_ylim([-3.5,2.5]), ax1.set_xlim([-0.005,1.715])

        hd_wd = .008*sum(np.abs(ax1.get_xlim()))
        hd_ln = 8*hd_wd
        for ar in range(len(posi)):
            if np.sign(Sigres[i][0][posi[ar]]) != 0:
                ax1.plot(xx[posi[ar]], Sigres[i][0][posi[ar]], 'o', ms=10, mfc='none',
                         mew=2,  mec='r')
                ax1.plot(xx[posi[ar]], C0res[i][0][posi[ar]], 'x', ms=8, mfc='none',
                         mew=1.5,  mec='r')

            if np.sign(Sigres[i][1][posi[ar]]) != 0:
                ax1.plot(xx[posi[ar]],Sigres[i][1][posi[ar]], 'o', ms=10, mfc='none',
                         mew=2,  mec='m')
                if Sigres[i][1][posi[ar]]<0:
                          ax1.plot(xx[posi[ar]], C0res[i][1][posi[ar]], 'x', ms=8,
                                   mfc='none', mew=1.5,  mec='m')
                if Sigres[i][1][posi[ar]]>=0:
                          ax1.plot(xx[posi[ar]], T0res[i][1][posi[ar]], 'x', ms=8,
                                   mfc='none', mew=1.5,  mec='m')

            if np.sign(Sigres[i][2][posi[ar]]) != 0:
                ax1.plot(xx[posi[ar]],Sigres[i][2][posi[ar]], 'o', ms=10, mfc='none',
                         mew=2,  mec='b')
                ax1.plot(xx[posi[ar]], Tstrres[i][2][posi[ar]], 'x', ms=8, mfc='none',
                         mew=1.5,  mec='b')

            if np.sign(Sigres[i][3][posi[ar]]) != 0:
                ax1.plot(xx[posi[ar]],Sigres[i][3][posi[ar]], 'o', ms=10, mfc='none',
                         mew=2,  mec='g')
                ax1.plot(xx[posi[ar]], Tstrres[i][3][posi[ar]], 'x', ms=8, mfc='none',
                         mew=1.5,  mec='g')

            if i<manu[5] or i==manu[-1]:
                          ax1.plot([xx[posi[ar]],xx[posi[ar]]],ax1.get_ylim(),
                                   ls=(0,(2,7,4,7)),c='k',lw=1)
            if i<manu[3] or i==manu[-1]:
                          ax1.annotate(" x = %s" % (xx[posi[ar]]),
                                       xy=(xx[posi[ar]],ax1.get_ylim()[1]),
                                       color='k', fontsize=12, va='top')
    xstart = 6
    subwidth = xstart + 7

    tr_rot = Affine2D().scale(.1,100).rotate_deg(90)

    ax2 = fig.add_axes([xx[posi[2]+xstart] / sum(np.abs(ax1.get_xlim())) *.775+.125,
           (-2.4-ax1.get_ylim()[0]) / sum(np.abs(ax1.get_ylim())) *.755+.125,
           (xx[posi[2]+subwidth]-xx[posi[2]+xstart]) / sum(np.abs(ax1.get_xlim())) *.775,
           (2.1--2.4) / sum(np.abs(ax1.get_ylim())) *.755 ])
    if True:
        plc = len(runn)-2
        posi = runn[plc][1]
        plcd = runn[plc][0]
        lbl = runn[plc][2]
        hh = [[-1.5*t,-1.5*t,-.5*t,-.5*t,.5*t,.5*t,1.5*t,1.5*t]
              for ele in range(len(Sigani[0]))]
```

```
    hh = [-1.5*t,-1.5*t,-.5*t,-.5*t,.5*t,.5*t,1.5*t,1.5*t]

    Cfan = np.array(Cstrres)[:,:,posi]
    Tfan = np.array(Tstrres)[:,:,posi]
    Sigan = np.array(Sigres)[:,:,posi]

    length = [sum([1 if ele==wth else 0 for idx,ele in enumerate(ll[0:plcd+1])])-1
              for wth in [0,1,2]]
    start = [y0[length[0]]<xx[posi], y1[length[1]]<xx[posi], y2[length[2]]<xx[posi]]
    start = [True if length[idx]==-1 else ele for idx,ele in enumerate(start)]
    start.append(start[-1])

    Inter = np.array([[0 if start[idx] else ele
                       for idx,ele in enumerate(inter[:,posi+1])]
                      for lst,Lst in enumerate(range(len(seq)))])

    Sigani = np.array([np.zeros(len(seq)),#0
                       Sigan[:,0],#1
                       Sigan[:,1]+Inter[:,1],#2
                       Sigan[:,1],#3
                       Sigan[:,2]+Inter[:,2],#4
                       Sigan[:,2],#5
                       Sigan[:,3],#6
                       np.zeros(len(seq))])#7
    Cfani = np.array([Cfan[:,0],Cfan[:,0],Cfan[:,1],Cfan[:,1],Cfan[:,2],Cfan[:,3]])
    Tfani = np.array([Tfan[:,0],Tfan[:,0],Tfan[:,1],Tfan[:,1],Tfan[:,2],Tfan[:,3]])

    Sigani[5] = [0 if Inter[:,2][idx]==0 else ele for idx,ele
                 in enumerate(Sigani[5])]
    Sigani[3] = [0 if Inter[:,1][idx]==0 else ele for idx,ele
                 in enumerate(Sigani[3])]

    for i in range(len(Sigani)-2): # i is per interface
        i +=2
        for j in range(len(Sigani[i])): # j is per plcd
            if i>1 and i<7 and Sigani[i][j]==0:
                Cfani[i-1][j] = np.nan
                Tfani[i-1][j] = np.nan
            if Sigani[i-1][j]==0:
                Sigani[i][j] = np.nan

    ax2.set_ylim(-2.4,2.1)
    ax2.set_xlim(1.75*t,-1.75*t-.5*(3.5*t))
    ax2.spines['bottom'].set_position('zero')
#     ax2.spines['top'].set_color('none')
    ax2.yaxis.tick_left()
    ax2.spines['left'].set_position(('axes',1))
#     ax2.spines['right'].set_color('none')
    ax2.patch.set_edgecolor('k'),ax2.patch.set_linewidth('1')
    ax2.xaxis.tick_bottom()
    ax2.set_ylabel('\u03C3 [N/mmš]', fontsize=10,ha='left')
    ax2.set_xlabel('w [m]', fontsize=10)
    ax2.set_title('Stresses at x=%s (rotated)' % (x[posi]),loc='left')

    if lbl==1:
        lyrs = np.count_nonzero([ele for ele in Sigani[:,plcd]
                                 if (not np.isnan(ele))])
        if lyrs==2:
            sig = [1,2,4,6]
        elif lyrs==4:
            sig = [1,3,4,6]
        else:
            sig = [1,3,5,6]

        clr = ['r','m','b','g']
        sig = [ele for ele in sig if (Sigani[:,plcd][ele]!=0)*(
            not np.isnan(Sigani[:,plcd][ele]))]
        for i in range(len(sig)):
            text= "\u03C3%s" % (sub('%s,%s,%s' % (i,plcd+1,xx[posi]))) #text
            labpo = [-np.sign(Sigani[sig[i],plcd])/2+0.5,
                     -(hh[sig[i]]-ax2.get_xlim()[0])/sum(np.abs(ax2.get_xlim()))]
            ax2.annotate(text, xy=(labpo[1],labpo[0]),
                         xycoords='axes fraction', color=clr[i], fontsize=14,
                         va=['bottom','top'][int(labpo[0])],
                         rotation=90)
            ax2.plot(hh[sig[i]],Sigani[sig[i],plcd], 'o', ms=10, mfc='none',
                     mew=2,  mec=clr[i])
            if plc>=3*4:
                if Sigani[sig[i],plcd]<0:
                       ax2.plot(hh[sig[i]],Cfani[sig[i]-1,plcd], 'x', ms=8,
                                mfc='none', mew=1.5,  mec=clr[i])
                if Sigani[sig[i],plcd]<0 and Cfani[sig[i]-1,plcd]<-2.4:
                       ax2.arrow(hh[sig[i]], -2.4+.4, 0, -.4,
                                 length_includes_head=True, shape='full',
```

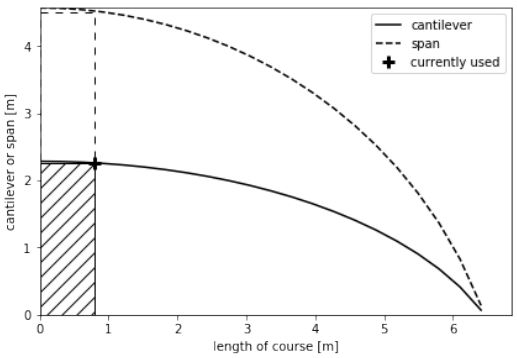
```
                                    width=.0004, head_width = .0015,
                                    head_length = .15, ec = 'none', fc = clr[i])
                        if Sigani[sig[i],plcd]>=0:
                               ax2.plot(hh[sig[i]],Tfani[sig[i]-1,plcd], 'x', ms=8,
                                        mfc='none', mew=1.5,  mec=clr[i])
                    ax2.plot([ax2.get_xlim()[0],hh[sig[i]]],
                             [Sigani[sig[i],plcd],Sigani[sig[i],plcd]],
                             c=clr[i],ls=(1,(2,4)))
                    ax1.plot([xx[posi],xx[posi+xstart]],
                             [Sigani[sig[i],plcd],Sigani[sig[i],plcd]],
                             c=clr[i],ls=(1,(2,4)))
                    ax1.arrow(xx[posi+1],
                              Sigani[sig[i],plcd],
                               -xx[posi+1]+xx[posi],
                               0,
                               length_includes_head=True,
                               shape='full',
                               head_width = .12,
                               head_length = xx[posi+1]-xx[posi],
                               ec = 'none',
                               fc = clr[i])
                ax2.plot(hh,Sigani[:,plcd],color='k',lw=2)
                x_fill = np.zeros(len(hh))

                ax2.fill_between(hh[1:7],Cfani[:,plcd],color='maroon',alpha=.1)
                ax2.fill_between(hh[1:7],Tfani[:,plcd],color='navy',alpha=.1)

        plt.show()
```

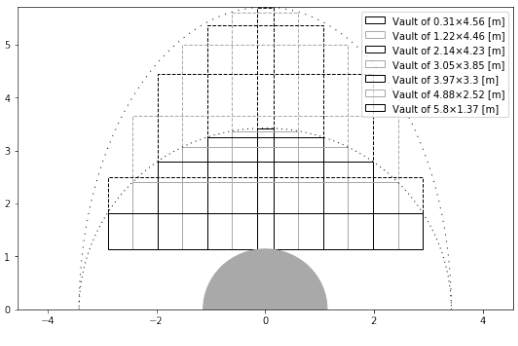


In [15]:
```
def annot_max1(x,y, ax=None):
    xmax = x[np.argmax(y)]
    ymax = y.max()
    text= "bricks placed={:}, uc={:.2f}".format(int(xmax), ymax)
    if not ax:
        ax=plt.gca()
    bbox_props = dict(boxstyle="square,pad=0.3", fc="w", ec="k", lw=0.72)
    arrowprops=dict(arrowstyle="->",connectionstyle="angle,angleA=0,angleB=60")
    kw = dict(xycoords='data',textcoords="axes fraction",
              arrowprops=arrowprops, bbox=bbox_props, ha="right", va="top")
    ax.annotate(text, xy=(xmax, ymax), xytext=(0.82,0.96), **kw)
```

In [16]:
```
# Maximum dimensions of vault
%matplotlib inline
%matplotlib inline
from matplotlib.patches import Rectangle
from matplotlib.patches import Circle
from matplotlib.patches import Ellipse

Ro = 3123 #mm
Ri = 1092.7
Ro = 3425
Ri = 1140
Ro,Ri = Ro/1000,Ri/1000
# a2+b2=c2

plt.rcParams["figure.figsize"] = (2*(Ro),2*(Ro-Ri)) # (w, h)
fig, ax = plt.subplots()

course = np.linspace(0,Ro+1,30)
course = [ele for ele in course if abs(ele)<=Ro]
```

```
cant = [np.sqrt(Ro**2 - ele**2)-Ri for ele in course]
course = [ele for idx, ele in enumerate(course) if cant[idx]>=0]
cant = [ele for ele in cant if ele>=0]
span = [2*ele for ele in cant]
cours = [2*ele for ele in course]

plt.plot(cours,cant,color='k',label='cantilever')
plt.plot(cours,span,linestyle='dashed',color='k',label='span')
width = .8
height= 4.5/2
plt.plot(width,height,'+',color='k',ms=10,label='currently used',mew=3)
ax.add_patch(Rectangle((0,0),width,height,fill=None,hatch='//'))
ax.add_patch(Rectangle((0,height),width,height,fill=None, linestyle=(0,(6,6))))

plt.ylim([0,2*(Ro-Ri)]), plt.xlim([0,2*(Ro)])

plt.xlabel('length of course [m]')
plt.ylabel('cantilever or span [m]')
plt.legend(prop={"size":10})
plt.show()

    #Other graph
x_0 = [-ele for idx,ele in enumerate(course) if idx % 3 ==1]
x_1 = [ele for idx,ele in enumerate(course) if idx % 3 ==1]
y_0 = [Ri for idx,ele in enumerate(cant) if idx % 3 ==1]
y_1 = [ele+Ri for idx,ele in enumerate(cant) if idx % 3 ==1]
y_2 = [ele+Ri for idx,ele in enumerate(span) if idx % 3 ==1]
d_x = [x_1[idx]-x_0[idx] for idx,ele in enumerate(x_0)]
d_y = [y_1[idx]-y_0[idx] for idx,ele in enumerate(x_0)]

plt.rcParams["figure.figsize"] = (2*(Ro+Ri),2*Ro-Ri) # (w, h)
fig, ax = plt.subplots()
lne = ['dashed','solid']

for i in range(len(x_0)):
    g = float((i % 2)/3*2)
    ax.add_patch(Rectangle((x_0[i], y_1[i]), (x_1[i]-x_0[i]), (y_1[i]-y_0[i]), fill=None,color=str(g),
    linestyle = 'dashed'))
    ax.add_patch(Rectangle((x_0[i], y_0[i]), (x_1[i]-x_0[i]), (y_1[i]-y_0[i]), fill=None,color=str(g),
                           label = 'Vault of %s×%s [m]' % (np.round(d_x[i],2),np.round(2*d_y[i],2))))

ax.add_patch(Circle((0,0), Ro,fill=None,linestyle=(0,(1,4))))
ax.add_patch(Ellipse((0,0), width=2*Ro,height=(4*Ro-2*Ri),fill=None,linestyle=(0,(1,6))))
ax.add_patch(Circle((0,0), Ri,color='darkgray'))

plt.ylim([0,2*Ro-Ri]), plt.xlim([-Ro-Ri,Ro+Ri])
plt.legend(prop={"size":10})
plt.show()
```

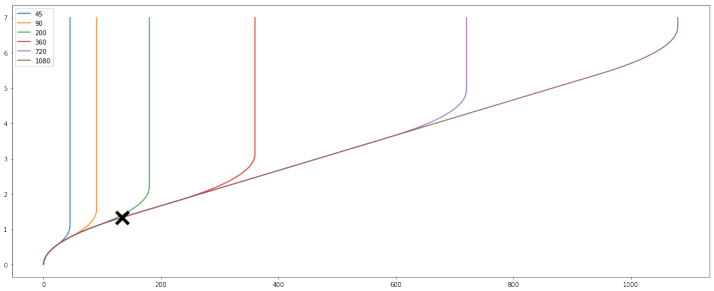





```
In [17]: from scipy.misc import derivative as diff
         H1 = (8/3)
         xq = np.linspace(0,7,101)
         def func(xq):
             afcd = (
                 (H1>(2*4/3))*(
                     (xq<(4/3))*(xq>=0) *
                     (.5*150*xq**2)
                     + (xq>=(4/3))*(xq<=(H1-4/3))  *
                     (200*(xq-4/3)+0.5*150*(4/3)**2)
                     + (xq>(H1-4/3))*(xq<=H1) *
                     (150*H1*xq -.5*150*xq**2 - .5*150*H1**2 +200*H1 - 800/3)
                     + (xq>H1) *
```

```
                    (200*(H1-4/3))
                )
                + (H1<=(2*4/3))*(
                    (xq<=(H1/2))*(xq>=0) *
                    (.5*150*xq**2)
                    + (xq>(H1/2))*(xq<=H1) *
                    (150*H1*xq-.5*150*xq**2-150/4*H1**2)
                    + (xq>H1) *
                    (150/4*H1**2)
                )
            )
            return afcd

        # def dif(xq):
        #     if H1>(2*4/3):
        #         afcd2 = (xq<(4/3)) * (150*xq) + (xq>=(4/3))*(xq<=(H1-4/3)) * (200) + (xq>(H1-4/3))*(xq<=H1) *
        #                 (200 - 150*(xq-H1+4/3)) + (xq>H1) * 0
        #     return afcd2
        # print(func(xq))
        print('4/3',func(4/3))
        # print('d4/3',diff(func,4/3))
        T = [(np.sqrt(30)/5),(2*np.sqrt(15)/5),(2*np.sqrt(30)/5),(47/15),(74/15),(101/15)]
        Deg = [45,90,200,360,720,1080]
        plt.figure(figsize=(20,8))
        for i in range(len(T)):
            H1 = T[i]
            print('H1',func(H1),H1)
            plt.plot(func(xq),xq,label=Deg[i]);
        plt.plot((400/3),(4/3),"x",ms=20,mec='k',mew=5)
        plt.legend();
        # plt.plot(dif(xq),xq);
```

```
4/3 133.33333333333331
H1 45.00000000000002 1.0954451150103321
H1 89.99999999999996 1.5491933384829668
H1 180.00000000000009 2.1908902300206643
H1 359.99999999999994 3.1333333333333333
H1 720.0 4.933333333333334
H1 1080.0 6.733333333333333
```

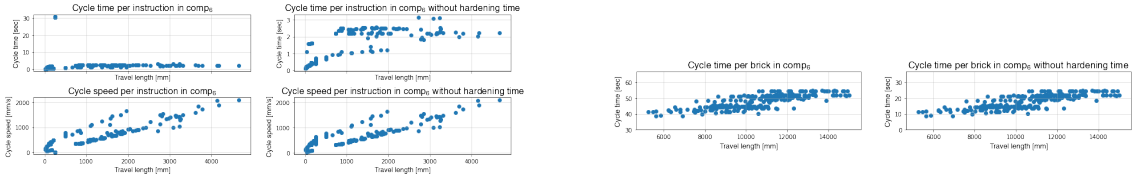


In [18]:
```
        #
        travel = [15696,10667,2534,168,1451,16061,41464,18238,1451,1415,11049,252,1451,1666,1451...
        # travel[0] = 46624
        times = [1414,2191,30425,123,408,2261,2014,1223,417,351,2191,176,422,1616,434,2191,30455,428,411,2495,2004,11...
        speed = [1110,487,8,137,355,710,2058,1491,348,403,504,143,344,103,335,479,8,491,353,671,1841,1649,8,300,1463,...
        times = [ele/1000 for ele in times]
        travel = [ele/10 for ele in travel]
        print(len(travel),len(times),len(speed))

        comper = "6"

        ### Per Instruction
        plt.rcParams["figure.figsize"] = (15,4.5) # (w, h)
        params = {'mathtext.default': 'regular' }
        plt.rcParams.update(params)
        fig,((ax0,ax1),(ax2,ax3)) = plt.subplots(2,2,sharex=True)
        # fig = plt.figure()
        # ax0 = fig.add_subplot(2,2,1)
        # ax1 = fig.add_subplot(2,2,2,sharex=ax0)
        # ax2 = fig.add_subplot(2,2,3,sharex=ax0)
        # ax3 = fig.add_subplot(2,2,1)

        ax0.set_title("Cycle time per instruction in $comp_%s$" % (comper),fontsize=16,fontname="Helvetica")
        ax0.set_xlabel("Travel length [mm]",fontsize=12,fontname="Helvetica")
        ax0.set_ylabel("Cycle time [sec]",fontsize=12,fontname="Helvetica")
```

```
ax0.grid(which='major',lw=.5,alpha=0.75)
ax0.scatter(travel,times)
# ax0.set_xlim(xlims)
# xlims = ax0.set_xlim()

timesmin = [ele-30 if ele>30 else ele for ele in times]

ax1.set_title("Cycle time per instruction in $comp_%s$ without hardening time" % (comper),fontsize=16,
fontname="Helvetica")
ax1.set_xlabel("Travel length [mm]",fontsize=12,fontname="Helvetica")
ax1.set_ylabel("Cycle time [sec]",fontsize=12,fontname="Helvetica")
ax1.grid(which='major',lw=.5,alpha=0.75)
ax1.scatter(travel,timesmin)

ax2.set_title("Cycle speed per instruction in $comp_%s$" % (comper),fontsize=16,fontname="Helvetica")
ax2.set_xlabel("Travel length [mm]",fontsize=12,fontname="Helvetica")
ax2.set_ylabel("Cycle speed [mm/s]",fontsize=12,fontname="Helvetica")
ax2.grid(which='major',lw=.5,alpha=0.75)
ax2.scatter(travel,speed)

speedmin = [travel[ind]/timesmin[ind] for ind,ele in enumerate(travel)]

ax3.set_title("Cycle speed per instruction in $comp_%s$ without hardening time" % (comper),fontsize=16,
fontname="Helvetica")
ax3.set_xlabel("Travel length [mm]",fontsize=12,fontname="Helvetica")
ax3.set_ylabel("Cycle speed [mm/s]",fontsize=12,fontname="Helvetica")
ax3.grid(which='major',lw=.5,alpha=0.75)
ax3.scatter(travel,speedmin)
# sns.regplot(np.array(travel), np.array(speedmin),ci=99.9,ax=ax3)
# b3,a3 = np.polynomial.polynomial.polyfit(travel,speedmin,deg=1)
# b30,b31 = b3+300, b3-300
# upper,downer = [a3*ele+b30 for ele in travel], [a3*ele+b31 for ele in travel]
# upper,downer = sum([speedmin[ind]<upper[ind] for ind,ele in enumerate(upper)]),sum([speedmin[ind]>downer[ind]
#         for ind,ele in enumerate(downer)])
# print(upper/len(travel),downer/len(travel))
# ax3.plot(travel, [a3*ele+b30 for ele in travel], ls='dotted', c='#1f77b4',alpha=0.5)
# ax3.plot(travel, [a3*ele+b31 for ele in travel], ls='dotted', c='#1f77b4',alpha=0.5)
plt.subplots_adjust(hspace=0.45)

#### Per brick
# travel0 = [57764,46934,50007,53566,57722,62457,67763,73616,67952,58728,53866,57012,61007,65541,69943,75678,7...
# times0 = [41464,42016,41820,41676,41237,41058,40624,40580,40924,41282,41357,41466,41753,42197,41661,42614,43...
# bri_ins0 = [10,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,10,18,24,24,24,24,24,24,24,24,24,24,14,14,...
# travel1 = [49619,55060,56783,58894,60672,63523,66880,70619,65125,54613,60847,62892,65320,68098,70926,74409,7...
# times1 = [41518,41798,41738,41627,41269,41078,40574,40468,40764,41103,41327,41436,41682,42068,41663,42570,43...
# bri_ins1 = [10,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,10,18,24,24,24,24,24,24,24,24,24,24,14,14,...
# travel2 = [63831,46128,48767,57125,57624,56944,61778,73027,67610,59040,61402,65765,74517,69024,69950,75236,7...
# times2 = [38311,41047,41050,42782,42726,40463,40484,39937,40172,41376,41448,41527,43287,43079,40835,40847,41...
# bri_ins2 = [10,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,14,10,18,24,24,24,24,24,24,24,24,24,24,14,14,...
# travel3 = [56839,46613,49330,53002,57231,61936,67074,72606,67296,59995,54976,57935,61646,65899,70641,75864,7...
# times3 = [39926,40849,40734,40590,40388,40143,39844,39603,40011,43540,43538,43477,43348,43241,42876,42265,42...
# travel4 = [61686,53530,56147,59740,63913,68582,73702,79232,74154,64961,59545,62460,66142,70382,75123,80356,7...
# times4 = [310782,312422,312318,312184,311990,311752,311460,311224,311641,312881,312852,312785,312651,312540,...
# travel5 = [70294,57708,60291,63802,67954,72704,71767,70305,66683,69576,73154,71924,78787,73961,82426,80715,8...
# times5 = [40077,41184,41074,40974,40868,40663,43463,43625,44122,43979,43842,38581,43330,46239,47497,46004,47...
travel6 = [75098,100549,101178,102773,104949,107657,110855,114504,108526,106988,97732,98774,100932,103650,106...
times6 = [41062,43799,43772,43759,43754,43755,43763,43775,43742,44686,44664,44658,44664,44679,44700,44743,447...
# travel7 = [58029,47434,49753,53464,57756,62556,67827,73524,67778,59267,54663,57543,61249,65415,70187,75490,7...
# times7 = [40663,41980,40941,40819,40639,40423,40134,39917,40349,42833,42724,42672,42560,41843,41521,40863,41...
times = [ele/1000 for ele in times6]
travel = [ele/10 for ele in travel6]
# bri_ins = bri_ins1
speed = [travel[ind]/times[ind] for ind,ele in enumerate(travel)]

# fig,((ax0,ax1),(ax2,ax3)) = plt.subplots(2,2,sharex="row")
plt.rcParams["figure.figsize"] = (15,1.8) # (w, h)
fig,(ax0,ax1) = plt.subplots(1,2,sharex="row")

ax0.set_title("Cycle time per brick in $comp_%s$" % (comper),fontsize=16,fontname="Helvetica")
ax0.set_xlabel("Travel length [mm]",fontsize=12,fontname="Helvetica")
ax0.set_ylabel("Cycle time [sec]",fontsize=12,fontname="Helvetica")
ax0.grid(which='major',lw=.5,alpha=0.75)
ax0.scatter(travel,times)
ax0.set_ylim(30,max(times)+12)

timesmin = [ele-30 if ele>30 else ele for ele in times]
# colo10 = ['#79a3c7' for ind,ele in enumerate(bri_ins) if bri_ins[ind]==10]
# colo12 = ['#6394bd' for ind,ele in enumerate(bri_ins) if bri_ins[ind]==12]
# colo14 = ['#4d85b4' for ind,ele in enumerate(bri_ins) if bri_ins[ind]==14]
# colo16 = ['#3776ab' for ind,ele in enumerate(bri_ins) if bri_ins[ind]==16]
# colo18 = ['#306898' for ind,ele in enumerate(bri_ins) if bri_ins[ind]==18]
# colo20 = ['#2a5b85' for ind,ele in enumerate(bri_ins) if bri_ins[ind]==20]
# colo22 = ['#244e72' for ind,ele in enumerate(bri_ins) if bri_ins[ind]==22]
# colo24 = ['#1e415f' for ind,ele in enumerate(bri_ins) if bri_ins[ind]==24]
```

```
        # colo26 = ['#18344c' for ind,ele in enumerate(bri_ins) if bri_ins[ind]==26]
        # colo28 = ['#122739' for ind,ele in enumerate(bri_ins) if bri_ins[ind]==28]
        # time10 = [ele for ind,ele in enumerate(timesmin) if bri_ins[ind]==10]
        # time12 = [ele for ind,ele in enumerate(timesmin) if bri_ins[ind]==12]
        # time14 = [ele for ind,ele in enumerate(timesmin) if bri_ins[ind]==14]
        # time16 = [ele for ind,ele in enumerate(timesmin) if bri_ins[ind]==16]
        # time18 = [ele for ind,ele in enumerate(timesmin) if bri_ins[ind]==18]
        # time20 = [ele for ind,ele in enumerate(timesmin) if bri_ins[ind]==20]
        # time22 = [ele for ind,ele in enumerate(timesmin) if bri_ins[ind]==22]
        # time24 = [ele for ind,ele in enumerate(timesmin) if bri_ins[ind]==24]
        # time26 = [ele for ind,ele in enumerate(timesmin) if bri_ins[ind]==26]
        # time28 = [ele for ind,ele in enumerate(timesmin) if bri_ins[ind]==28]
        # dist10 = [ele for ind,ele in enumerate(travel) if bri_ins[ind]==10]
        # dist12 = [ele for ind,ele in enumerate(travel) if bri_ins[ind]==12]
        # dist14 = [ele for ind,ele in enumerate(travel) if bri_ins[ind]==14]
        # dist16 = [ele for ind,ele in enumerate(travel) if bri_ins[ind]==16]
        # dist18 = [ele for ind,ele in enumerate(travel) if bri_ins[ind]==18]
        # dist20 = [ele for ind,ele in enumerate(travel) if bri_ins[ind]==20]
        # dist22 = [ele for ind,ele in enumerate(travel) if bri_ins[ind]==22]
        # dist24 = [ele for ind,ele in enumerate(travel) if bri_ins[ind]==24]
        # dist26 = [ele for ind,ele in enumerate(travel) if bri_ins[ind]==26]
        # dist28 = [ele for ind,ele in enumerate(travel) if bri_ins[ind]==28]

        ax1.set_title("Cycle time per brick in $comp_%s$ without hardening time" % (comper),fontsize=16,
        fontname="Helvetica")
        ax1.set_xlabel("Travel length [mm]",fontsize=12,fontname="Helvetica")
        ax1.set_ylabel("Cycle time [sec]",fontsize=12,fontname="Helvetica")
        ax1.grid(which='major',lw=.5,alpha=0.75)
        # ax1.scatter(dist10,time10, color=colo10,label='10')
        # # ax1.scatter(dist12,time12, color=colo12,label='12')
        # ax1.scatter(dist14,time14, color=colo14,label='14')
        # ax1.scatter(dist16,time16, color=colo16,label='16')
        # ax1.scatter(dist18,time18, color=colo18,label='18')
        # ax1.scatter(dist20,time20, color=colo20,label='20')
        # ax1.scatter(dist22,time22, color=colo22,label='22')
        # ax1.scatter(dist24,time24, color=colo24,label='24')
        # # ax1.scatter(dist26,time26, color=colo26,label='26')
        # ax1.scatter(dist28,time28, color=colo28,label='28')
        ax1.scatter(travel,timesmin)
        # ax1.legend(ncol=4, framealpha=0.4, handletextpad=0.2, columnspacing=1.2)
        # ax1.legend(['8 instructions','12 instructions','14 instructions','16 instructions','18 instructions',
        #         '20 instructions','22 instructions','24 instructions','26 instructions','28 instructions'])
        ax1.set_ylim(0,max(timesmin)+12)

        # ## Per Station
        # stat = ['Pallet','Adhesive','Vault']
        # stattim = [ 1220,5626,10470]
        # statdis = [1390628,1061459,1400550]

        # stattimmin = [stattim[0],stattim[1],stattim[2]-30*len(timesmin)]
        # stattimplu = [ele - stattimmin[ind] for ind,ele in enumerate(stattim)]

        # ax2.set_title("Total time for all instructions per station in $comp_%s$" % (comper),fontsize=16,
        #         fontname="Helvetica")
        # ax2.set_xlabel("Station",fontsize=12,fontname="Helvetica")
        # ax2.set_ylabel("Time [sec]",fontsize=12,fontname="Helvetica")
        # ax2.bar(stat,stattimmin,width=0.6,label="Without hardening time")
        # ax2.bar(stat,stattimplu,width=0.6,bottom=stattimmin, color="cornflowerblue",label="Hardening time")
        # ax2.legend()
        # ax3.set_title("Total distance for all instructions per station in $comp_%s$" % (comper),fontsize=16,
        #         fontname="Helvetica")
        # ax3.set_xlabel("Station",fontsize=12,fontname="Helvetica")
        # ax3.set_ylabel("Distance [mm]",fontsize=12,fontname="Helvetica")
        # ax3.bar(stat,statdis,width=0.6)

        plt.subplots_adjust(hspace=0.5)
        plt.show();
```

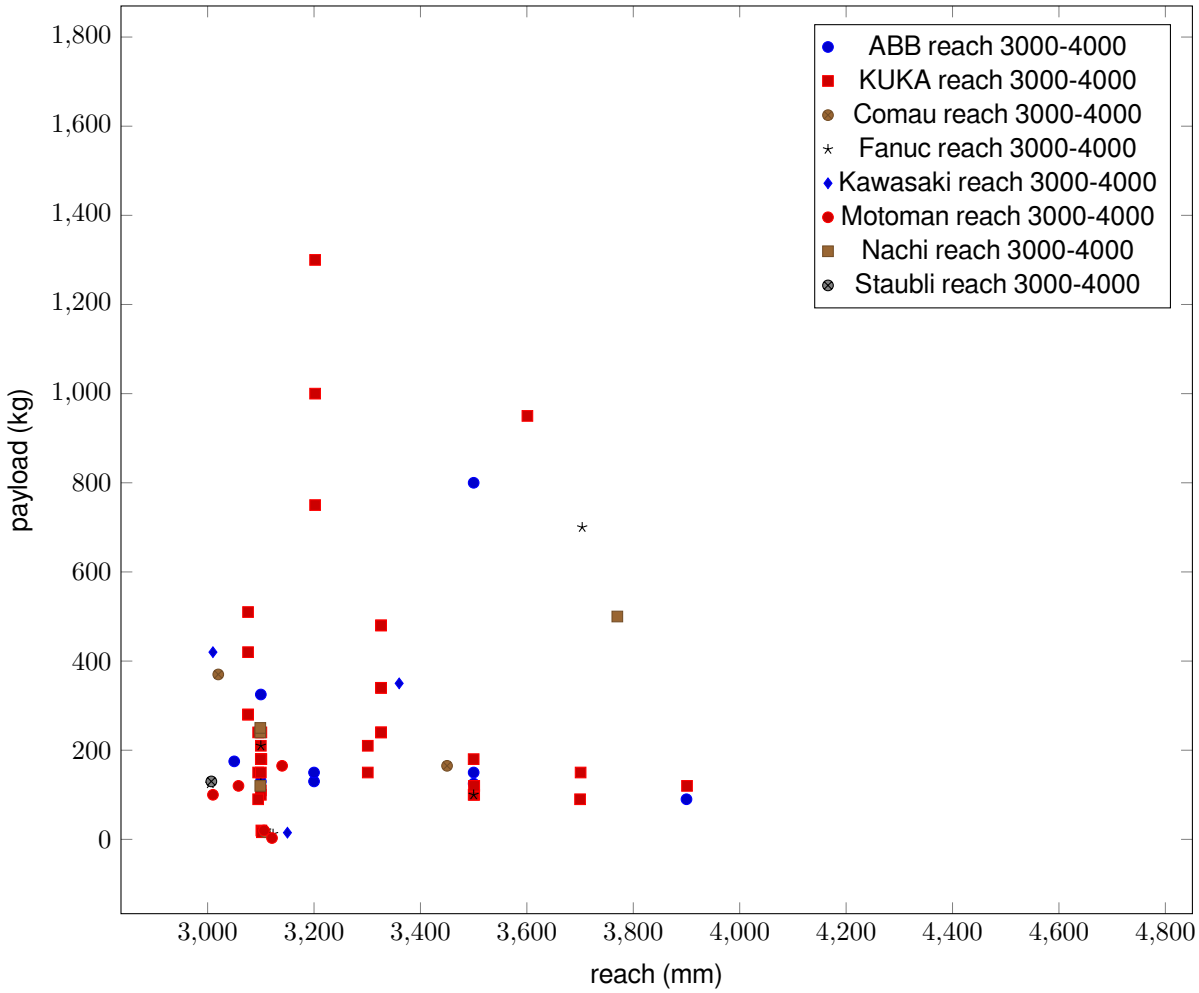
```
548 548 548
```

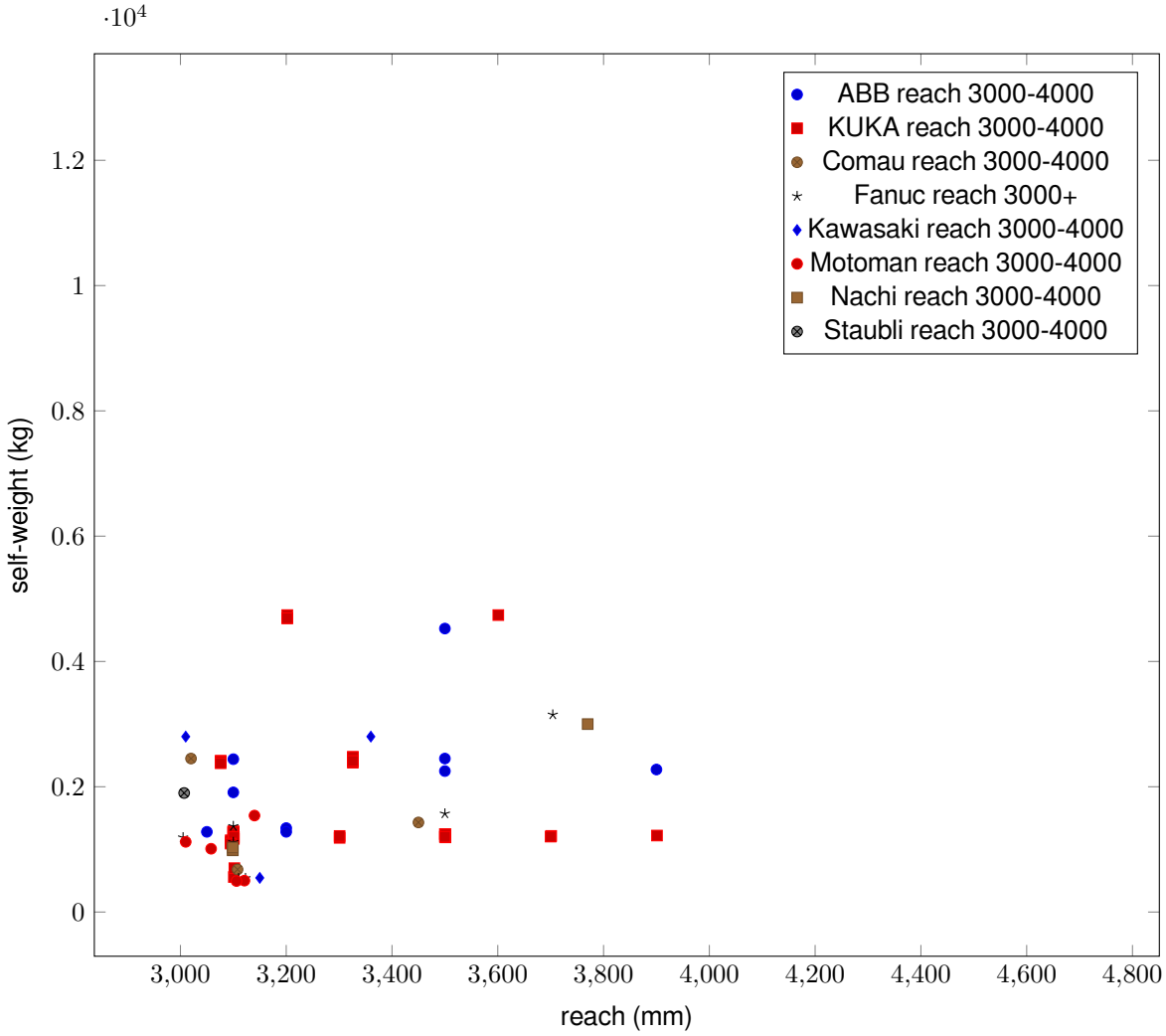


E

Robotic arms from the RoboDK library

This chapter provides a broader view of the robots that could have been selected and the overall programming from the works done in RoboDK.




$\cdot 10^4$
1.2
1
0.8
0.6
0.4
0.2
0
self-weight (kg)
3,000
3,200
3,400
3,600
3,800
4,000
4,200
4,400
4,600
4,800
reach (mm)
ABB reach 3000-4000
KUKA reach 3000-4000
Comau reach 3000-4000
Fanuc reach 3000+
Kawasaki reach 3000-4000
Motoman reach 3000-4000
Nachi reach 3000-4000
Staubli reach 3000-4000

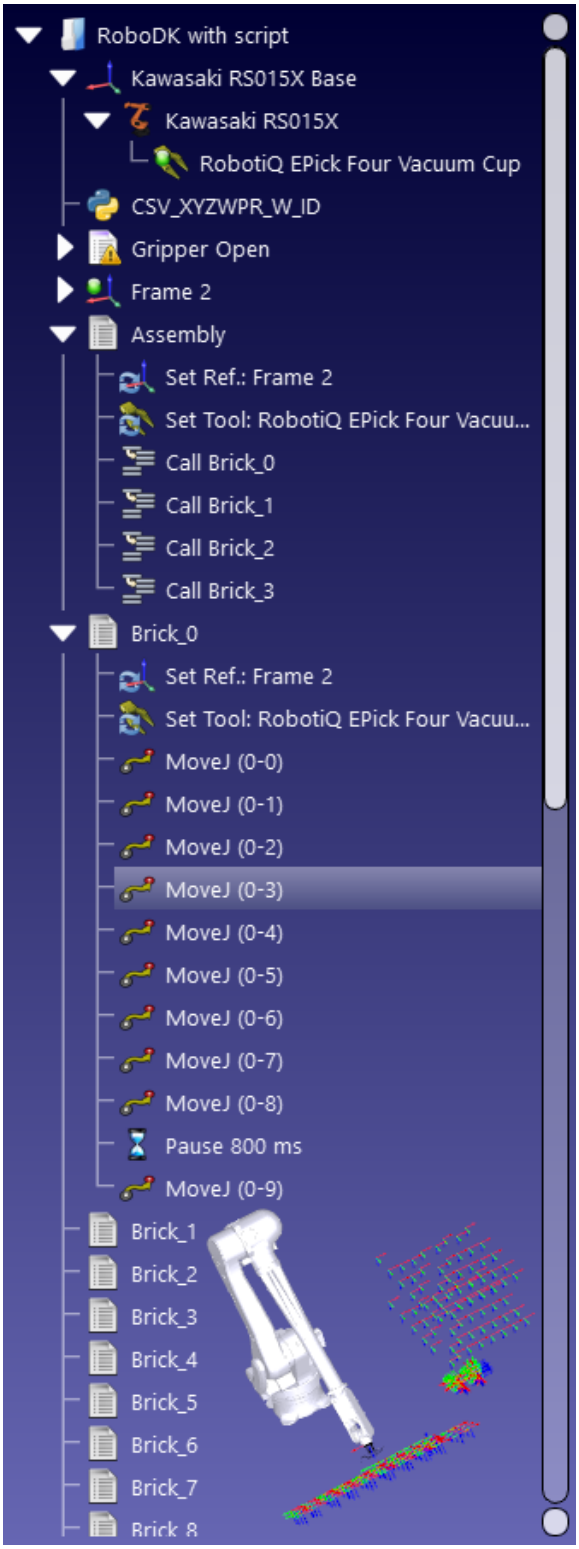


Figure E.1: Work process in RoboDK (as GUI).

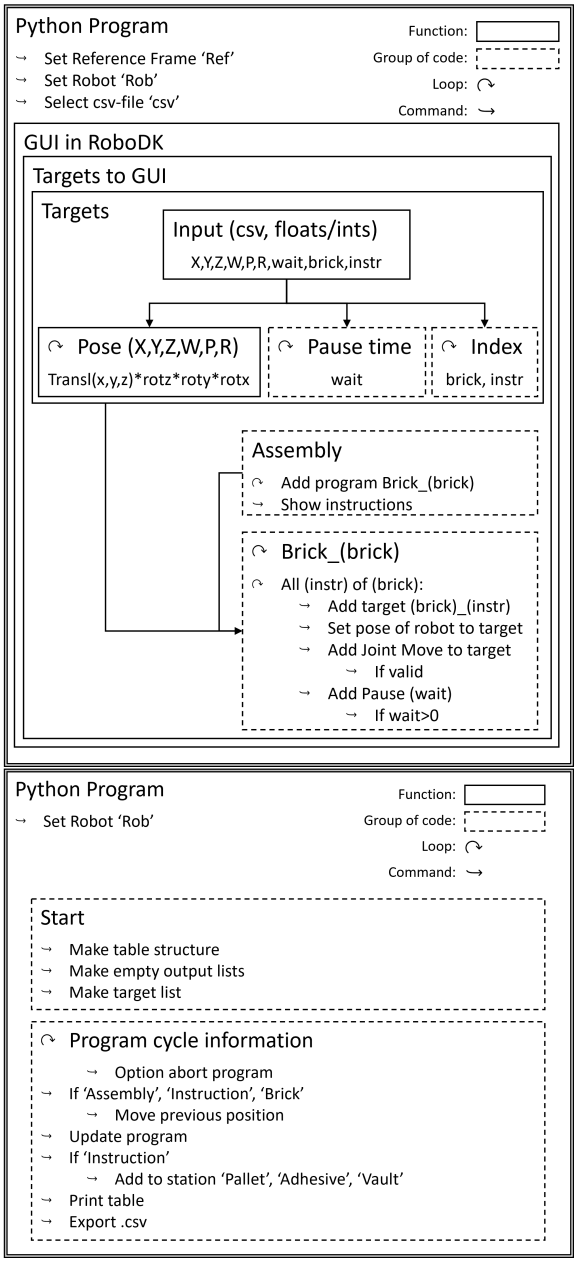


Figure E.2: Workflows in Python.

In [1]:
```
print("This is python file CSV_XYZWPR_W_ID_2.py")
```

```
This is python file CSV_XYZWPR_W_ID_2.py
```

In [2]:
```
# This macro can load CSV files from Denso programs in RoboDK.
# Supported types of files are:
#  1-Tool data : Tool.csv
#  2-Work object data: Work.csv
#  3-Target data: P_Var.csv
# This macro can also filter a given targets file

# Type help("robolink") or help("robodk") for more information
# Press F5 to run the script
# Visit: http://www.robodk.com/doc/PythonAPI/
# For RoboDK API documentation

from robolink import *    # API to communicate with RoboDK
from robodk import *      # basic matrix operations

# Start communication with RoboDK
RDK = Robolink()

# Ask the user to select the robot (ignores the popup if only
ROBOT = RDK.ItemUserPick('Select a robot', ITEM_TYPE_ROBOT)

# Check if the user selected a robot
if not ROBOT.Valid():
    quit()

# Automatically retrieve active reference and tool

#Remove old stuff
##fram = RDK.Item('Targets', ITEM_TYPE_FRAME)# Get the first program called program_name
##if fram.Valid():
##        fram.Delete()
##fram = RDK.Item('Objects', ITEM_TYPE_FRAME)# Get the first program called program_name
##if fram.Valid():
##        fram.Delete()

FRAME = ROBOT.getLink(ITEM_TYPE_FRAME)
##print(FRAME)
##FRAME = Rhino.RDK.AddFrame('Targets','Rhino')
##FRAMO = Rhino.RDK.AddFrame('Objects','Rhino')
TOOL = ROBOT.getLink(ITEM_TYPE_TOOL)

##FRAME = RDK.ItemUserPick('Select a reference frame', ITEM_TYPE_FRAME)
##TOOL = RDK.ItemUserPick('Select a tool', ITEM_TYPE_TOOL)


if not FRAME.Valid() or not TOOL.Valid():# or not FRAMO.Valid():
    raise Exception("Select appropriate FRAME and TOOL references")

#       103 7101
redoit = mbox("Is this a continuation? \n Type No if this is first iteration \n Check row 49",entry='No')
if redoit == 'No':
    start=0
else:
    start = int(redoit)

# Function to convert XYZWPR to a pose
# Important! Specify the order of rotation
def xyzwpr_to_pose(xyzwpr):
    x,y,z,rx,ry,rz = xyzwpr
    return transl(x,y,z)*rotz(rz*pi/180)*roty(ry*pi/180)*rotx(rx*pi/180)
    #return transl(x,y,z)*rotx(rx*pi/180)*roty(ry*pi/180)*rotz(rz*pi/180)
    #return KUKA_2_Pose(xyzwpr)

# csv_file = 'C:/Users/Albert/Desktop/Var_P.csv'
csv_file = getOpenFile(RDK.getParam(r'C:/Users/Joris/Documents/Master Thesis quick access/RoboDK'))

# Specify file codec
codec = 'utf-8' #'ISO-8859-1'

# Load P_Var.CSV data as a list of poses, including links to reference and tool frames
def load_targets(strfile):
    csvdata = LoadList(strfile, ',', codec) # X, Y, Z, W, P, R, wait, brick, instruction, # instr. per brick
    poses = []
    waits = []
    idxs = []
    for i in range(0, len(csvdata)):
        x,y,z,rx,ry,rz = csvdata[i][0:6]
        poses.append(xyzwpr_to_pose([x,y,z,rx,ry,rz]))
        waits.append(csvdata[i][6])
        #idxs.append(csvdata[i][6])
        idxs.append(csvdata[i][7:10])
```

```
    return poses, idxs, waits

# Load and display Targets from P_Var.CSV in RoboDK
def load_targets_GUI(strfile,start):
    #input
    poses, idxs, waits = load_targets(strfile) # Values from csv file

    program_name = 'Assembly' # Name program GH_RDK
    program_name = program_name.replace('-','_').replace(' ','_')
    print(program_name)
    #Remove old stuff
    program = RDK.Item(program_name, ITEM_TYPE_PROGRAM)# Get the first program called program_name
    if start==0:
        if program.Valid():
            program.Delete()

        #Add & set
        program = RDK.AddProgram(program_name, ROBOT)   # Add program and relate to used robot (optional)
        #program.setFrame(FRAME)    #Obsolete. reference frame of robot
    program.setPoseFrame(FRAME) # Sets reference frame of robot
    program.setPoseTool(TOOL)   # Set the robot tool pose (TCP) with respect to the robot flange
        #program.setTool(TOOL)  #Obsolete. robot tool pose to robot flange
    program.ShowInstructions(show=False)
    RDK.setSimulationSpeed(1)

    ls,js,la,ja = 100,200,150,150 #mm/s,deg/s,mm/sš,deg/sš
    program.setSpeed(speed_linear=ls,speed_joints=js,accel_linear=la,accel_joints=ja)
    js1,js2,js3,js4,js5,js6 = 180,180,200,410,360,610
##    program.setSpeedJoints(js1,js2,js3,js4,js5,js6)

    #instructions
    #1
    program.MoveJ(ROBOT.JointsHome())

    #2-...
    proglist,statlist,instruclist = [],[],[]

##    for i in range(4):
##        stat_name = ['Pallet','Adhesive','Vault','Inbetween'][i] # Name program GH_RDK
##        stat_name = stat_name.replace('-','_').replace(' ','_')
##        #Remove old stuff
##        stat = RDK.Item(stat_name, ITEM_TYPE_PROGRAM)# Get the first program called program_name
##        if stat.Valid():
##                stat.Delete()
##        #Add & set
##        stat = RDK.AddProgram(stat_name, ROBOT)   # Add program and relate to used robot (optional)
##        stat.setPoseFrame(FRAME) # Sets reference frame of robot
##        stat.setPoseTool(TOOL)   # Set the robot tool pose (TCP) with respect to the robot flange
##        statlist.append(stat)
    if start==0:
        redo=int(idxs[-1][0])+1
    else:
        redo=0
    for i in range(redo): #Create each program call, len() won't work, -1 is last brick, 0 is brick number
        if (i+1)/10 == int((i+1)/10):
            print(i+1,int(idxs[-1][0])+1)


##        program.RunInstruction('Brick_%s' % (i))    #Assemble all program calls
        #Remove old stuff
##        brick_name = 'Bricks_P_%i' % (i)
##        bric = RDK.Item(brick_name, ITEM_TYPE_OBJECT)# Get the first program called program_name
##        if bric.Valid():
##                bric.Delete()
##        RDK.AddFile(r'C:/Users/Joris/Documents/Master Thesis quick access/RoboDK/Objects/Bricks_P_%i.stp' % (i))

        prog_name = 'Brick_%s' % (i) # Name program GH_RDK
        prog_name = prog_name.replace('-','_').replace(' ','_')
        #Remove old stuff
        prog = RDK.Item(prog_name, ITEM_TYPE_PROGRAM)# Get the first program called program_name
        if prog.Valid():
                prog.Delete()
        #Add & set
        prog = RDK.AddProgram(prog_name, ROBOT)   # Add program and relate to used robot (optional)
        prog.setPoseFrame(FRAME) # Sets reference frame of robot
        prog.setPoseTool(TOOL)   # Set the robot tool pose (TCP) with respect to the robot flange
        prog.ShowInstructions(show=False)
        prog.setSpeed(speed_linear=ls,speed_joints=js,accel_linear=la,accel_joints=ja)
##        prog.setSpeedJoints(js1,js2,js3,js4,js5,js6)
        proglist.append(prog)

    print('Brick_i done')

    for j in range(start,len(idxs)):
        bri, instr = int(idxs[j][0]), int(idxs[j][1])

##        proglist[bri].RunInstruction('Instruction_%s_%s' % (bri,instr))    #Assemble all program calls
```

```
        if int(j/11)==j/11:
            print(j,len(idxs)-1)
            breakable = mbox("Are you sure you want to break the program?",b1 = 'Yes!', b2 = 'No!',t=.5)
            if breakable == True:
                print('Write down:' +j)
                break
            instruc_name = 'Instruction_%s_%s' % (bri,instr) # Name program GH_RDK
            instruc_name = instruc_name.replace('-','_').replace(' ','_')
            #Remove old stuff
            instruc = RDK.Item(instruc_name, ITEM_TYPE_PROGRAM)# Get the first program called program_name
            if instruc.Valid():
                    instruc.Delete()
            #Add & set
            instruc = RDK.AddProgram(instruc_name, ROBOT)   # Add program and relate to used robot (optional)
            instruc.setPoseFrame(FRAME) # Sets reference frame of robot
            instruc.setPoseTool(TOOL)   # Set the robot tool pose (TCP) with respect to the robot flange
            instruc.ShowInstructions(show=False)
            instruc.setSpeed(speed_linear=ls,speed_joints=js,accel_linear=la,accel_joints=ja)
##        instruc.setSpeedJoints(js1,js2,js3,js4,js5,js6)

        for pose, idx, wait in zip(poses, idxs, waits):
##                bri, instr = int(idxs[i][0]), int(idxs[i][1])
            if idx[0:2]==[bri,instr]:
                name = '%i-%i' % (bri,instr)    # Sets name of target to (GH_RDK-0)
                #Remove old stuff
                target = RDK.Item(name, ITEM_TYPE_TARGET)   # Get the first target named 'name'
                if target.Valid():
                    target.Delete()
                #Add target
                target = RDK.AddTarget(name, FRAME, ROBOT)  # Add target to reference frame and to robot
                target.setPose(pose) # Pose relative to robot reference frame

                try:
    ##                    if idx[1]==3 or idx[1]==idx[2]-3:
    ##                        program.MoveJ(target)
    ##                    else:
    ##                        program.MoveL(target)

                    instruc.MoveJ(target)
                    if wait>0:
                        instruc.Pause(wait)
                except:
                    if int(j/11)==j/11:
                        print('Warning: %s can not be reached. It will not be added to the program' % name)
                try:
                    proglist[bri].MoveJ(target)
                    if wait>0:
                        proglist[bri].Pause(wait)
                except:
                    print('Warning: %s can not be reached. It will not be added to the program' % name)
                try:
                    program.MoveJ(target)
                    if wait>0:
                        program.Pause(wait)

    ##                    if idx[1]==1:
    ##                        AttachClosest(keyword='Bricks_P_%i' % (idx[0]),tolerance_mm=15)
    ##                    if idx[1]==idx[2]-2:
    ##                        DetachAll(parent=FRAMO)

                except:
                    print('Warning: %s can not be reached. It will not be added to the program' % name)

        if int(j/11)==j/11:
            del instruc
##        statlist[].RunInstruction('Instruction_%s_%s' % (bri,instr))    #Assemble all program calls

    program.MoveJ(ROBOT.JointsHome())
    program.InstructionListJoints(flags=4,save_to_file=
    r"C:\Users\Joris\Documents\Master Thesis quick access\RoboDK\jointlist.csv")
    #


def load_targets_move(strfile):
    poses, idxs = load_targets(strfile)

    ROBOT.setFrame(FRAME)
    ROBOT.setTool(TOOL)

    ROBOT.MoveJ(ROBOT.JointsHome())

    for pose, idx in zip(poses, idxs):
        try:
            ROBOT.MoveJ(pose)
        except:
            RDK.ShowMessage('Target %i can not be reached' % idx, False)
    ROBOT.MoveJ(ROBOT.JointsHome())
```

```
        # Force just moving the robot after double clicking
        #load_targets_move(csv_file)
        #quit()

        # Recommended mode of operation:
        # 1-Double click the python file creates a program in RoboDK station
        # 2-Generate program generates the program directly

        MAKE_GUI_PROGRAM = False

        ROBOT.setFrame(FRAME)
        ROBOT.setTool(TOOL)


        if RDK.RunMode() == RUNMODE_SIMULATE:
            MAKE_GUI_PROGRAM = True
            # MAKE_GUI_PROGRAM = mbox('Do you want to create a new program? If not, the robot will just move along
             #         the tagets', 'Yes', 'No')
        else:
            # if we run in program generation mode just move the robot
            MAKE_GUI_PROGRAM = False



        if MAKE_GUI_PROGRAM:
            RDK.Render(False) # Faster if we turn render off
            load_targets_GUI(csv_file,start)
        else:
            load_targets_move(csv_file)

        RDK.ShowMessage('Program has run')


        ---------------------------------------------------------------------------

        ModuleNotFoundError                       Traceback (most recent call last)

        <ipython-input-2-01b1dc9e8f0f> in <module>()
         11 # For RoboDK API documentation
         12
    ---> 13 from robolink import *    # API to communicate with RoboDK
         14 from robodk import *      # basic matrix operations
         15


        ModuleNotFoundError: No module named 'robolink'


In [3]: print("This is python file Cycle Time Asembly, Brick & Instructions.py")

This is python file Cycle Time Asembly, Brick & Instructions.py


In [4]: # This example shows how to quickly calculate the cycle time of all programs in the RoboDK station
        #
        # Important notes and tips for accurate cycle time calculation:
        # https://robodk.com/doc/en/General.html#CycleTime

        # Start the RoboDK API
        from robolink import *    # RoboDK API
        from robodk import *
        import numpy as np
        RDK = Robolink()
        ROBOT = RDK.ItemUserPick('Select a robot', ITEM_TYPE_ROBOT)

        Comp = 5

        check_comp = mbox("Have you updated Comp? \n Please check \n Really awful if you hadn't",b1 = 'Yes!',
        b2 = 'No, thanks!')
        if check_comp == False:
            print('Change comp!')
            quit()

        export = [["Program name","Have all targets been reached?","Travel length [mm]","Cycle Time [s]",
        "Cycle Speed [mm/s]"]]
        writeline = "Program name\tHave all targets been reached?\tTravel length\tCycle Time\tCycle Speed"
        msg_html = "<table border=1><tr><td>"+writeline.replace('\t','</td><td>')+"</td></tr>"
        Pallt,Adhet,Vault = [],[],[]
        Pallx,Adhex,Vaulx = [],[],[]
        exprt = [['Pallt','Pallx','Adhet','Adhex','Vault','Vaulx']]
        i=0
        tot = len(RDK.ItemList(ITEM_TYPE_PROGRAM))
        targ = RDK.ItemList(ITEM_TYPE_TARGET)
        targ = [[int(ta) for ta in ele.Name().split('-')] for ele in targ]
```

```
# Ask the user to select a program
#program = RDK.ItemUserPick('Select a program', ITEM_TYPE_PROGRAM)
for program in RDK.ItemList(ITEM_TYPE_PROGRAM):
    if int(i/5)==i/5 or i==1:
        breakable = mbox("Are you sure you want to break the program?",b1 = 'Yes!', b2 = 'No!',t=.5)
        if breakable == True:
            print('Note down:' +i)
            break
        i+=1
        print(i,tot)
    else:
        i+=1
    if program == RDK.Item('Gripper Open'):
        continue
    elif program == RDK.Item('Assembly') or program == RDK.Item('Brick_0')
    or program == RDK.Item('Instruction_0_0'):
        ROBOT.MoveJ(ROBOT.JointsHome())
        target = program
        ROBOT.WaitFinished()
    elif program.Name().split('_')[0]=='Instruction' and int(program.Name().split('_')[2])!=0:
        name = '%i-%i' % (int(program.Name().split('_')[1]),int(program.Name().split('_')[2])-1)
        target = RDK.Item(name, ITEM_TYPE_TARGET)
        ROBOT.MoveJ(target)
        ROBOT.WaitFinished()
    else:
        name = int(program.Name().split('_')[1])-1
        name = '%i-%i' % (int(program.Name().split('_')[1])-1,max([int(ele[1]) for ele in targ
        if int(ele[0])==name]))
        target = RDK.Item(name, ITEM_TYPE_TARGET)
        ROBOT.MoveJ(target)
        ROBOT.WaitFinished()

##    if program == RDK.Item('Gripper Open'):
##        continue
##    elif program == RDK.Item('Assembly') or program == RDK.Item('Brick_0')
##    or program == RDK.Item('Instruction_0_0'):
##        ROBOT.MoveJ(ROBOT.JointsHome())
##        ROBOT.WaitFinished()
##    else:
##        name = '%i-%i' % (int(program.split('_')[1]),int(program.split('_')[2])-1)
##        target = RDK.Item(name, ITEM_TYPE_TARGET)
##        ROBOT.MoveJ(target)
##
####        start_inst = previ.Instruction(previ.InstructionCount()-1)[0]
####        if 'Instruction' in start_inst:
####            RDK.RunCode(start_inst)
####        else:
####            RDK.RunCode(previ.Name())
##        ROBOT.WaitFinished()
####    target = RDK.ItemUserPick(itemtype_or_list=ITEM_TYPE_TARGET)
####    ROBOT.MoveJ(target)
##    previ = program
##    # Retrieve the robot linked to the selected program
##    #robot = program.getLink(ITEM_TYPE_ROBOT)
##
##    # Output the linear speed, joint speed and time (separated by tabs)

    result = program.Update()
    instructions, time, travel, ok, error = result
    speed = travel/time
    if program in [RDK.Item('Brick_%s' %(i)) for i in range(3)]:
        speed = travel/(time-30)
    if program == RDK.Item('Assembly'):
        speed = travel/(time-3*30)
    if ok==1:
        ok = 'Yes'
    elif ok==0:
        ok = 'No'
    if 'Instruction' in program.Name():
        nm,br,ins = program.Name().split('_')
        sum_ins = len([ele for ele in RDK.ItemList(ITEM_TYPE_PROGRAM,True) if 'Instruction_%s' % (br) in ele])
        if int(ins)<3:
            Pallt = (time)
            Pallx = (travel)
            exprt.append([Pallt,Pallx,0,0,0,0])
        elif int(ins)>sum_ins-4:
            Vault = (time)
            Vaulx = (travel)
            exprt.append([0,0,0,0,Vault,Vaulx])
        elif int(ins)>2 and int(ins)<sum_ins-3:
            Adhet = (time)
            Adhex = (travel)
            exprt.append([0,0,Adhet,Adhex,0,0])

    # Print the information
    newline = "%s\t %s \t %.1f mm \t %.1f s \t %.1f mm/s" % (program.Name(), str(ok), travel, time, speed)
```

```
        export.append([program.Name(),str(ok),travel, time, speed])

        msg_html = msg_html + '<tr><td>' + newline.replace('\t','</td><td>') + '</td></tr>'


    try:
        export = [[str(ele).replace('.',',') for ele in export[idx] ] for idx,Ele in enumerate(export)]
        exprt = [[str(ele).replace('.',',') for ele in exprt[idx] ] for idx,Ele in enumerate(exprt)]
        np.savetxt(r"C:\Users\Joris\Documents\Master Thesis quick access\RoboDK\CycleTime.csv",
        export,delimiter ="; ",fmt ='% s')
        np.savetxt(r"C:\Users\Joris\Documents\Master Thesis quick access\RoboDK\CycleTime2.csv",
        exprt,delimiter ="; ",fmt ='% s')
        np.savetxt(r"C:\Users\Joris\Documents\Master Thesis quick access\RoboDK\CycleTime_%s.csv" % (Comp),
        export,delimiter ="; ",fmt ='% s')
        np.savetxt(r"C:\Users\Joris\Documents\Master Thesis quick access\RoboDK\CycleTime2_%s.csv" % (Comp),
        exprt,delimiter ="; ",fmt ='% s')
        RDK.ShowMessage('CSV created/updated')
    except:
        pass
    msg_html = msg_html + '</table>'

    program_name = 'Assembly' # Name program GH_RDK
    program_name = program_name.replace('-','_').replace(' ','_')
    program = RDK.Item(program_name, ITEM_TYPE_PROGRAM)
    program.InstructionListJoints(flags=4,save_to_file=r"C:\Users\Joris\Documents\Master Thesis quick access
    \RoboDK\jointlist.csv")

    RDK.ShowMessage(msg_html)


    ---------------------------------------------------------------------------

    ModuleNotFoundError                       Traceback (most recent call last)

    <ipython-input-4-d47a08201e1a> in <module>()
      5
      6 # Start the RoboDK API
----> 7 from robolink import *    # RoboDK API
      8 from robodk import *
      9 import numpy as np


    ModuleNotFoundError: No module named 'robolink'
```