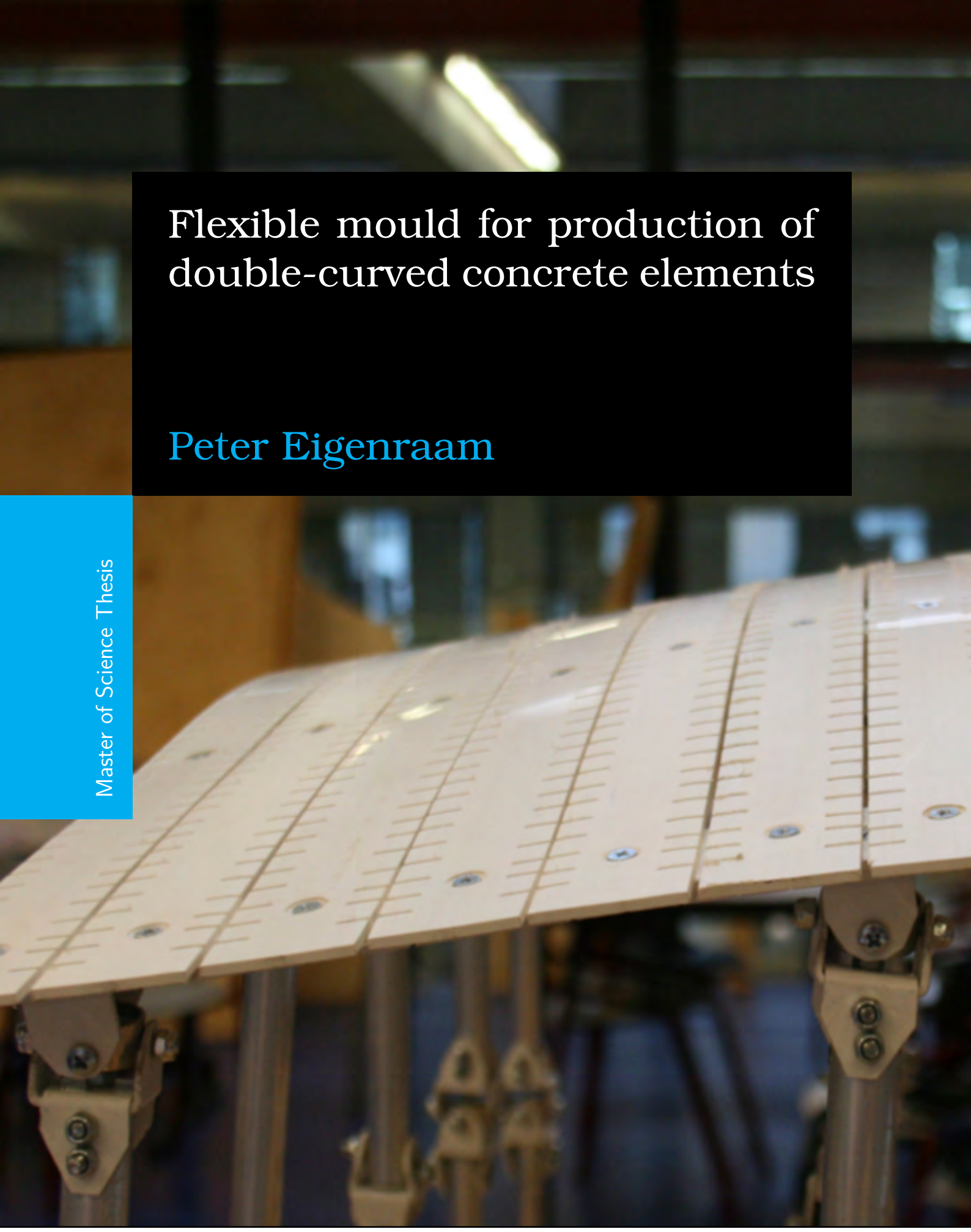
Flexible mould for production of double-curved concrete elements

Peter Eigenraam

Master of Science Thesis



TU Delft Delft University of Technology

Flexible mould for production of double-curved concrete elements

Master of Science Thesis

For the partial fulfillment of the requirements for the degree of Master of Science in Building Engineering at Delft University of Technology

Student:
Peter Eigenraam

Committee members:
Ir. A. Borgart
Dr.ir. J.L. Coenders
Prof.dr.ir. J. Rots (Chairman)
Ir. H.R. Schipper

June 28, 2013

Faculty of Civil Engineering and Geosciences (CE) · Delft University of Technology

Summary

Freeform shapes in concrete

Within architectural design a large variety of complex shaped buildings can be found. This is partly thanks to geometry-defining software which make modeling of almost any shape possible and gives designers a tool to design an infinite variety of objects. Freeform shapes play an increasingly important role in contemporary architecture and pose big challenges to fabrication of these structures. Production of double curved concrete elements is one of the responses. Prefabricated double curved concrete elements currently find their main application in façades. Two examples of double curved concrete elements can be found are shown in the Figures 1 and 2. They are the façades of the Pierre Budin daycare in Paris (France) and the Heydar Aliyev Cultural Center in Baku (Azerbaijan). Current production methods for double curved concrete elements often involve Computer Numerical Controlled milling machines which produce mould from polystyrene blocks. In most favorable situations these moulds can reused in limited series making the production very expensive.

Figure 1: Pierre Budin daycare in Paris (France) by ECDM [Ductal, 2013].

Figure 2: Heydar Aliyev Cultural Center in Baku (Azerbaijan) by Zaha Hadid Architects [Akhundov, 2008].

Research goal and question

A flexible mould for the production of double curved concrete elements could offer an cheaper alternative for current production methods. The main advantages of a flexible mould is that it can be reused multiple times and each time for a unique shape. The idea of the flexible mould for double curved element dates back to the 1960's when Renzo Piano experimented with a flexible mould for plastic elements. Since then several alternative moulds have been designed and tested. At the Technical University in Delft research and development of a flexible mould was started by Dr. Ir. Karel Vollers and later with Ir. Daan Rietbergen. Currently, June 2013, Ir. Roel Schipper is rounding up his PhD research on manufacturing double-curved concrete elements at the Faculty of Civil Engineering and Geosciences. During this research a new version of the flexible mould was developed together with Ir. Bas Janssen which was based on earlier results by Vollers and Rietbergen. This mould was the starting point of this research and can be seen in Figure 6. Since the mould was built to fulfill research requirements within a limited budget it was expected that the mould would have shortcomings that could cause inaccurate geometry of elements. Also the behavior of the mould during deformation is difficult to understand due to its three-dimensional character. This research was set up aiming to gain more insight in this behavior and to find aspects that influence accuracy of the mould. Therefore the following research question was formulated; Which aspects significantly influence the geometry of the mould surface?

The flexible mould

The concept of a flexible mould is very simple: a flexible surface or membrane is forced into a required shape. This can be any shape and is also referred to as freeform. The concrete which was poured on top of the mould takes the shape of the mould after deformation. However, the mixture of the concrete should not be too liquid since the concrete could flow out of the mould. Before production support heights are determined and set. Then the concrete is casted while the mould is kept in horizontal position by a temporary structure until the concrete has sufficiently stiffened. Then the mould surface can be deformed by lowering a temporary support structure as can be seen in Figure 3. When al supports have reached the set height the required shape has been obtain and the concrete can be left to harden. This can be seen in Figure 4.

At the start of this research a series of test with the flexible mould were performed to gain more insight in the performance of the flexible mould. And also a more theoretical approach was used when looking into deformation of general surfaces. The relation between the change of Gaussian curvature and strain of a surface provided a way to explain displacement, rotations and, very important, in-plane shear deformation of the mould surface, which is illustrated in Figure 5. After this it was concluded that the accuracy of the mould could not be ensured because horizontal displacements and rotation of the mould could not be controlled sufficiently and cause deviation of the intended shape.

Figure 3: The flexible mould during production of a double curved concrete element.

Figure 4: A deformed mould in which the concrete is left to harden.

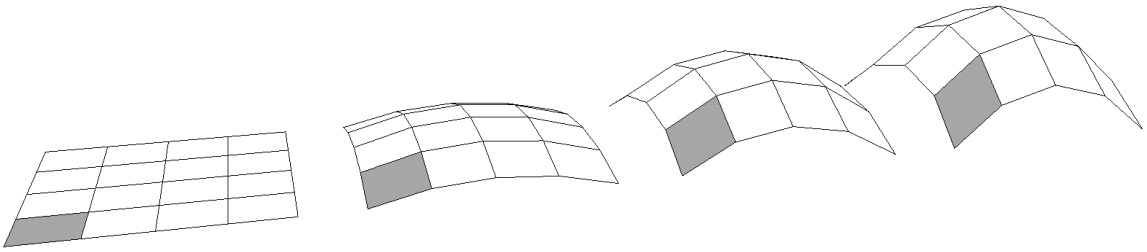


Figure 5: Shear deformation due to a increase of Gaussian curvature.

The following aspect could be improved to provide more control over the geometry;

- An alternative method for applying additional loading of the surface
- A mould surface with low resistance to shear deformation
- Connections between two layer of strips that allows in-plane rotation
- Connections between strips and supports that allow horizontal and rotational translations
- Compensation method for deviations caused by horizontal displacements and rotation the mould surface

Design for new prototype mould

It was found that the accuracy of the previous version of the mould could not be ensured due to shortcomings in its buildup. These shortcomings have been identified. Based on these findings and results from the research on deformation of surfaces a design for a new prototype has been made and build, which can be seen in Figure 7, in order to research the findings in more detail. Therefore the following solutions have been introduced, which can also be seen in the Figures 8 to 11:

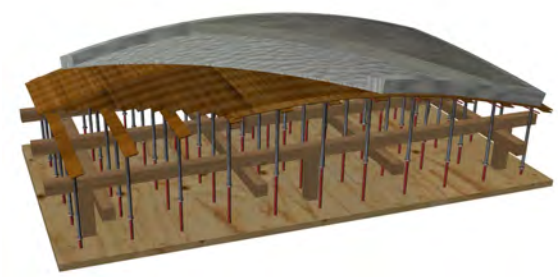


Figure 6: Version of the flexible mould developed by Ir. Roel Schipper and Ir. Bas Janssen.

Figure 7: New prototype of the flexible mould as was introduced in this research.

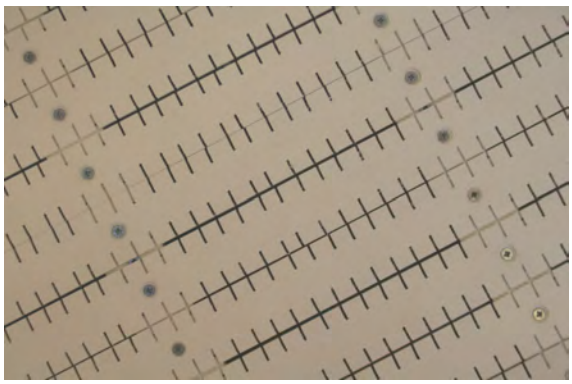


Figure 8: Plastic strips with notches. Notches reduce the resistance to deformation of the strips. Most importantly in their strong direction.

Figure 9: Universal joint connections allow rotation of the mould surface and are capable to transfer a force.

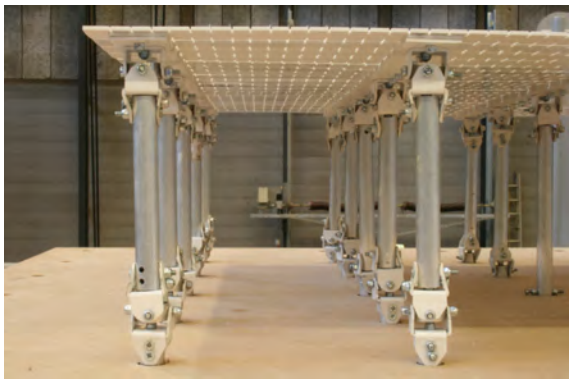


Figure 10: Three-dimensional pendulum supports allow horizontal displacements and are capable to transfer a force.

Figure 11: Weight applied for additional loading such that no extra loading of the mould surface is required.

- Plastic strips with notches. Strips have been bolted together, but not fully tightened. Self locking nuts have been used. Notches where made to reduce the resistance to deformation of the strips. Most importantly in their strong direction.
- Universal joints as connections that allow rotation while being capable of transferring a force
- Three-dimensional pendulum supports that allow horizontal displacement while being capable of transferring a force
- Weights as alternative way to apply additional loading to the surface

Method for setup of the flexible mould

Deformation of the mould surface results in horizontal displacements and rotations of the mould surface. These cause a deviation of the intended shape. A method has been developed compensates that aims to compensate for these deviations by calculating and adjusting the support heights. Therefore the method approximates the horizontal displacements and rotations. The compensation can be calculated by kinematic calculations which use the result of structural calculations.

During the structural calculation an aspect was noticed that is of importance for the accuracy of the mould surface. Calculation where performed for shapes with continuous curvature. From the used boundary condition at the end supports of the strips it followed that there was no bending moment. From theory of mechanics follows that the curvature will also be zero. But the required shape has continuous curvature. A deviation of the shape was the result. In Figure 12 the intended and calculated shape are shown when five support are used. Here can be seen that near the ends of the strip there is a deviation while in the middle it cannot be seen. This can be explained by the presence of moments. Figure 13 shows the difference between the intended shape and the calculated shape. At the point of the supports the deviation becomes zero. A clear distinction can be made in the deviation of the near ends and the middle of the strips. The effect can also be seen when the number of support increases. This could be solved by creating means to introduce moments at the end supports. However this would make the buildup of the mould unnecessary complicated. It is recommended not to use the end parts of the strips and thereby the edges of the mould surface.

Tests and performance of the new prototype

To research the behavior of the mould the introduced ideas for the new prototype were tested with four different shapes. The geometry of these shapes was determined using a 3D scanner to gain more insight in the accuracy of the flexible mould as can be seen in Figure 14.

Two type of measurement have been performed. Horizontal displacement and the overall accuracy. Unfortunately the accuracy of the measurements for the horizontal displacement were not sufficiently accurate to validate the expected displacements. More accurate measurements are required. However the accuracy of the total mould surface was determined more accurately an were compared to the intended geometry. This is shown in Figure 15.

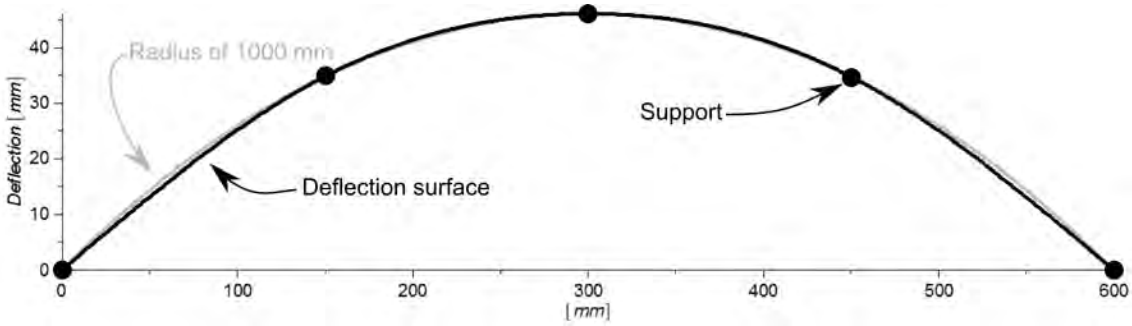


Figure 12: Intended and calculated shape of a strip on five supports. Conditions: R = 1000 mm, q = 0 kN/m, EI = 192550 Nmm^2.

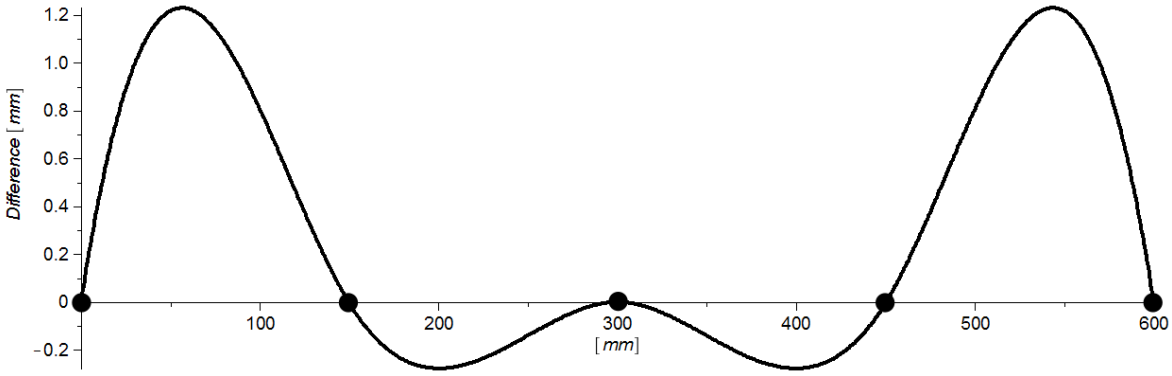


Figure 13: Difference between intended and calculated shape.

Figure 14: Scanning of tested shape with the flexible mould.

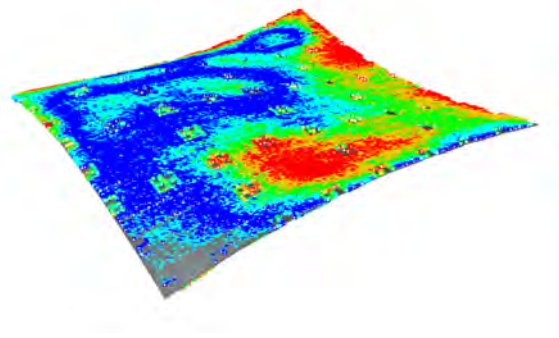


Figure 15: Point deviation of the scanned points compared the intended surface.

The mean deviation of the four tested shaped were respectively 0.91, 1.25, 1.22 and 1.63 millimeter. Locally three of the four test showed a maximum point deviation between 4 and 5 millimeter. Only the last test, which was free of form, showed a maximum point deviation between 4 and 7 millimeter.

If only the middle of the surface is considered the mean deviation reduces to respectively 0.78, 0.96, 0.86 and 1.11 millimeter. Locally all tests showed a maximum point deviation between 3 and 4 millimeter.

Conclusions

This research was set out to find aspect that influence the accuracy of the flexible mould for production of double curved concrete elements. The following aspects have been found to have significant influence on the geometry of the mould surface:

- Level of change of Gaussian curvature following directly from the required shape
- Stiffness of the mould surface. And especially the resistance to in plane shear deformation
- Loading of the surface
- Direction and size of displacements and rotations of the mould surface
- Type of material used for the strips
- Buildup and connection details of the mould

In order to research the behavior of the flexible mould in detail a new prototype has been designed, build and tested. The maximum measured deviation of tested shaped compared to intended shapes is 7 millimeter if the full surface was taken into account. When the area between the last two supports would not be used the maximum deviation reduced to 4 millimeter.

Table of Contents

Preface

This thesis was submitted in partial fulfillment of the degree of Master of Science in Building Engineering at the Faculty of Civil Engineering of the Technical University of Delft. This thesis contributed to research and development of a flexible mould for the production of double curved concrete elements.

In August 2012 I started this research. Now, in June 2013, I can look back and know that I have worked on this topic with great pleasure. The part of this thesis I liked the most was that I could combine theory and practice. I went through theory of structural mechanics and have been casting concrete in the laboratory. At one moment I was wearing the 'Engineering shoes' and the next 'practical boots'. Figure 16 nicely illustrates this when I was preparing for a series of castings.

Figure 16: 'Engineering shoes' and 'practical boots'. I liked working alternating on theoretic and practical topics during this research.

Peter Eigenraam

Acknowledgments

I would like to express my appreciation to all those who provided me the possibility to complete this report. In particular I would like to thank my daily supervisor Roel Schipper with who I worked together with much pleasure. Thank you for being inspirational and critical at the same time. I also would like to thank the other committee members Andrew Borgart, Jeroen Coenders and Jan Rots for providing me with the chance to present my work and giving me feedback during the course of my research.

With the completion of this thesis I also finish a period of studies as a student. Throughout this period my parents, brothers, sister, grandfathers and grandmothers have always supported and inspired me. Words cannot express how thankful I am for their care and support. So I finish with just saying thanks.

Delft, University of Technology
June 28, 2013

Peter Eigenraam

Peter Eigenraam

”It’s only fun when you are trying to get it in your grasp. Once you catch it throw it back into the water and then catch it again”

— *John Mayer*

Chapter 1

Introduction

Within architectural design a large variety of complex shaped buildings can be found. This type of architecture is also referred to as freeform design in those cases that shapes are applied that cannot easily be defined mathematically by the used of primitives such as e.g. spheres, cones or cylinders. Geometry-defining software makes modeling of almost any shape possible, giving designers a tool to design an infinite variety of objects. Freeform shapes play an increasingly important role in contemporary architecture and pose big challenges to fabrication of these structures. Research on and development on the flexible mould for production of double curved concrete elements is one way of facing these challenges. The flexible mould can be used to make uniquely shape concrete elements using the same mould multiple times.

1-1 Freeform design

The use of freeform design in society is strongly related to the development of graphical software since the 1960s. Fields of design, media and architecture adopted this software around the 1990s. Some say that computer graphics have become the language of contemporary design and architecture [Manovich, 2008]. A research area has emerged around freeform design in architectural geometry which is situated at the border between applied geometry and architecture [cec]. Architects, but also engineers have become more and more skilled in designing freeform structures resulting in esthetically and structurally interesting buildings. Early examples of modern concrete structures with complex shape can be found around the 1940's. Important contributions come from famous engineers like; Pier Luigi Nervi, Felix Candela, Ove Arup and Heinz Isler. The Figures 1-1 to 1-3 show some of the buildings that were designed by these engineers. More recent examples come from architects like Frank Gehry, in Figure 1-4, Zaha Hadid , Coop Himmelb(l)au, UNStudio and many more.

Figure 1-1: Airplane hangar in Orvito (Italy) by Pier Luigu Nervi (1938). Early example of freeform structure [Anengineersaspect, 2009].

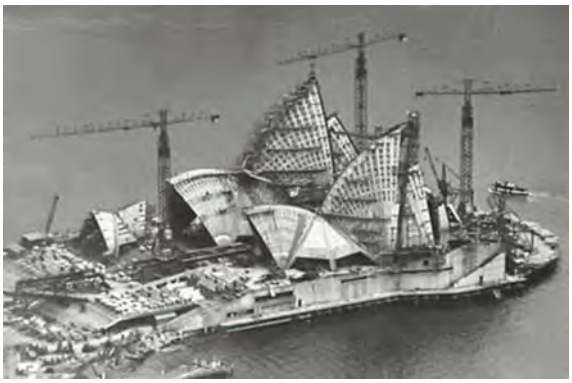


Figure 1-2: Opera House in Sidney (Australia) engineered by Arup (1973) [Lasplash, n.d.].

Figure 1-3: Tennis hall in Heimberg (Switserland) by Heinz Isler (1978) [Shadesofgreendesign, 2011].

Figure 1-4: Walt Disney Concert Hall in Los Angeles by Frank Gehry (2003). A more recent example where the complexity of the geometry has increased [Lauren and James, 2012].

1-2 Freeform shapes in concrete

Freeform structures have been designed and realized using all kinds of materials. Concrete has often been used for shell structures. When designed well these structures can become both esthetically pleasing and structurally efficient. Like normal concrete structures, freeform structures need a formwork in which the concrete can be cast. The cost of the formwork make up for a substantial part of the total budget of the concrete structure because the formwork is often difficult to make and requires a lot of manual labor and expertise. For freeform shapes highly skilled woodworkers are needed and the complexity of the shape in combination with the required accuracy make that labor costs will increase quickly for freeform concrete structures. Traditional wooden formwork cannot always be used. This makes production of the formwork very expensive. A recent example of a freeform roof is shown in Figure 1-5. It is a residential building in Amsterdam (The Netherlands) called "Het Funen" . The roof has been made using traditional formwork as is shown in Figure 1-6.

Figure 1-5: Residential building "Het Funen" by NLArchitects [NLArchitects, 2012].

Figure 1-6: Roof under construction. Scaffolding is partly visible [NLArchitects, 2012].

Figure 1-7: CNC (Computer numeric controlled) mill for foam or polystyrene blocks used by Nedcam [Nedcam, 2012a].

Figure 1-8: Boat hull milled out of polystyrene blocks [Nedcam, 2012a].

An alternative is to make formwork out of foam or polystyrene blocks. Shapes can be milled according to the required shape using CNC (Computer Numeric Controlled) mills. This technique is also used for construction of boat hulls. In the Figures 1-7 and 1-8 two CNC mills are shown which are used by Nedcam. An example of a project where milled foam blocks where used as formwork is the Spencer Dock Bridge in Dublin (Ireland) designed by Future Systems. This project is shown in Figure 1-9. The formwork was designed and manufactured by Nedcam. EPS foam blocks were milled into shape and coated with several layers of hot spray polyurea for a high quality surface, after which the blocks were placed on a scaffolding structure as in Figure 1-10. After a required hardening period of five weeks the scaffolding and foam formwork was removed to reveal a smooth and curved concrete surface [Nedcam, 2012b].

1-3 Application of prefabricated double curved concrete elements

Prefabricated double curved concrete elements currently find their main application in façades. Two examples of double curved concrete elements can be found are shown in the Figures 1-

Figure 1-9: Spencer Dock Bridge (2009) in Dublin (Ireland) by Future Systems [Infomatique, 2010].

Figure 1-10: CNC milled formwork designed by Nedcam. Foam blocks are placed on a scaffolding structure [Nedcam, 2012b].

11 and 1-12. They are the façades of the Pierre Budin daycare in Paris (France) and the Heydar Aliyev Cultural Center in Baku (Azerbaijan). The latter façade consists of concrete elements that are fixed on a steel space frame. The elements are cast into customized moulds. The molds are partly from CNC-cut 2D ribs and partly from 3D-milled styrofoam blocks [Dispenza, 2011]. A disadvantage of this method is that the formwork can be used only once or, at best, only for small series of elements since the geometry of the elements is different for most individual panels. A flexible mould as used in this research does not have disadvantage because its shape can be changed for every individual element. This is the main reason why a flexible mould would be an interesting production method for double curved concrete elements. However, until now, the flexible mould method is still in the stage of research and development, not having been applied yet in commercial application in building industry. This Master's thesis research aims to contribute to the development of such a method.

The workflow in the production of double curved concrete elements comprises several steps which makes the process rather complex: first, freeform shapes need to be divided into smaller pieces that can be prefabricated. This requires the use of penalization-algorithms. Secondly, a mould shape needs to be installed or produced for each individual element, likely using a CNC-method. The shape of each element needs to be produced within sufficient accuracy to enable an esthetically acceptable fitting during the assembly stage. Thirdly, the elements need to be cast, to be cured, to harden for a certain time, and finally to be transported to the building site for final assembly.

1-4 Research framework and problem description

This research project focuses on the development of a flexible mould for double curved concrete elements that can be reused and each time for a unique shape. The research addresses a number of issues that have been found in earlier researches on the flexible mould at Technical University Delft, and aims to contribute to solving or improving these issues. Although interest at TU Delft for complex building shapes already stems from the last decades of the 20th century, more intensive research was initiated around 2004, at that time also referred

Figure 1-11: Pierre Budin daycare in Paris (France) by ECDM [Ductal, 2013].

Figure 1-12: Heydar Aliyev Cultural Center in Baku (Azerbaijan) by Zaha Hadid Architects [Akhundov, 2008].

to as Blobs. The following students have contributed to the development of a flexible mould through their Master's or PhD thesis; Hans Jansen [2004], Marlies Quack [2001], Michelle van Roosbroeck [2006], Eline den Hartog [2008], Daan Rietbergen, Koen Huyge and Arnoud Schoofs [2009] and Bas Janssen Janssen [2011]. Currently and parallel to this research implementation of textile reinforcement has been researched by Marijn Kok [2013]. A wide variety of topics is researched which will be discussed in the next chapter. Since 2001 and still research is performed by Dr. Karel Vollers at the Faculty of Architecture. For specific application of a flexible mould for concrete elements Roel Schipper performs research within a PhD at the Faculty of Civil Engineering and Geosciences. So the latter has focused specifically on application of the flexible mould in combination with concrete. Topics that have been research are; freeform architecture, applications of elements, techniques of manufacturing, mechanic behavior and laboratory experiments [Schipper and Janssen, 2011].

The existing model of the flexible mould has some shortcomings. So far the mould has been improved incrementally. However it has been build to fit research requirements within reasonable budget. Unavoidable the mould has shortcomings which cannot be neglected in further development. The accuracy of the mould is unknown. An aspects that influence the accuracy have not yet been researched in detail. There are theoretical and practical issues that need to be overcome. The geometry of the mould surface after deformation is difficult to predict due its 3-dimensional character. Predicting the shape is required to gain insight in the factors that have influence on the shape. When elements are made tolerances need to be fulfilled. The mould buildup determines the tolerances, possibilities and limitations in shape and provide control to set the shape.

In Figure 1-13 the mould devise is shown. This is the starting point of this research. A multi layered surface on which concrete can be cast is supported by multiple supports. These can be set to curtain height and influences the shape of the surface. A new shape requires resetting of the support heights. Initially the devise is fixed in a horizontal position. At this stage the concrete can be cast. When the devise is released the weight of the concrete acts on the surface which will therefore deform according to the set support heights. A more elaborate description of the devise will be given in the next chapter. For practical reason in further text the device is referred to as 'the mould'. Which could be confused with the top part containing the actual concrete.

Figure 1-13: Flexible mould used for production of double curved concrete element at the Delft Technical University. A flexible surface on multiple adjustable supports [Schipper, 2013].

Figure 1-14: Test for a double curved element in the laboratory. The mould surface consists of separate strips [Schipper, 2013].

Within an earlier Master's thesis research contributions have been made by Bas Janssen [2011]. An important result of this research was the application of strips instead of a solid plate. This can be seen in Figure 1-14. More research on the following topics is needed to come to a fully functional commercial application of the mould:

- Mould buildup and production process.
- Geometry of the elements and production tolerances.
- Concrete composition and reinforcement.
- Application of elements and detailing of connections.

This thesis will focus on the mould and the geometry of the elements. The mould buildup and the geometry of the surface are directly related and therefore they are treated together.

1-5 Research goal and questions

The goal of this research is to understand the behavior of the mould during deformation. This understanding could help improving the mould. because the surface of the mould should match a reference geometry as close as necessary. Therefore more insight is required in the aspects that influence the geometry and accuracy of the mould. Accuracy of the mould can be researched in term of tolerances and deviation compared to a reference geometry. Predicting the shape of the surface presents geometric and structural challenges. All these aspects are formulated in the following research questions;

Main question:

- *Which aspects significantly influence the geometry of the mould surface?*

Secondary questions:

- *How do aspects influence the geometry?*
- *Which changes on the mould buildup will improve control over the geometry?*
- *How can the deformations and shape of the surface including the edges be predicted?*
- *Within which tolerances can the surface be set?*

1-6 Research strategy and thesis outline

Within this research four parts can be distinguished in order to achieve the research goal and to answer the questions.

Part 1 Observation and assessment of existing mould: to become familiar with the mould buildup and the production process several castings of elements have been done in the laboratory using the existing mould. An assessment has been done on the relation between change of Gaussian curvature and strain of a surface. Studies on other relevant and related literature has been done and will be referred to throughout the chapters. Chapter 2 describes observations of the mould buildup, the production process, shortcomings of the existing flexible mould and the assessment of the relation between Gaussian curvature and strain of a surface.

Part 2 Design of a new flexible mould prototype: a new prototype has been designed and build. Modifications and improvements of the mould are introduced based on the results of the first stage. Part 2 and 3 have been parallel in process. Introduced improvements and methods have been developed simultaneous. Chapter 3 describes technical details and the process of building the new mould prototype.

Part 3 Development of method for setup of the new flexible mould: a method has been developed to obtain the required input needed for the production process. The result of this method are the support heights that are needed for a curtain required shape. Several factors and their influence on the geometry of the mould surface are described. A distinction can be made between mechanical effects (deformation and support reactions) and kinematics effects (displacement). The method includes steps that can also be used to determine the location of the edges for freeform shape. Chapter 4 describes the developed method.

Part 4 Testing performance and accuracy of new prototype: the new mould prototype has been tested and the results of the developed method has been compared to measurements. Geometry of shapes is obtained using 3D laser scanner resulting in a pointcloud. This was used to determine displacements of specific points on the mould surface and the to determine the accuracy of the mould . To process the geometric data a method has been developed. Chapter 5 describes the test setup, scans, methods and results.

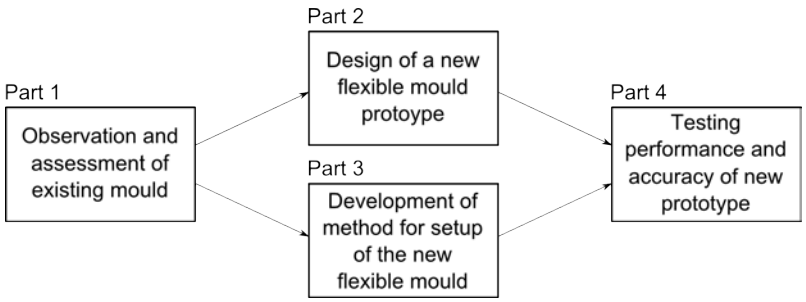


Figure 1-15: Research strategy

Chapter 2

Description of existing flexible mould

Double curved concrete elements are often fabricated using CNC-milled moulds. Moulds can be reused in limited series which leads to high production costs. There is an alternative method which makes use of a flexible mould. The mould can be reused multiple times and each time for a new unique shape. The architect Renzo Piano was one of the first to describe such a mould already in the 1960's. Figure 2-1 shows a sketch of his idea. Vertical adjustable pistons support a flexible surface. Since then the concept has been developed further. Different types of moulds where made for glass, plastic and concrete elements.

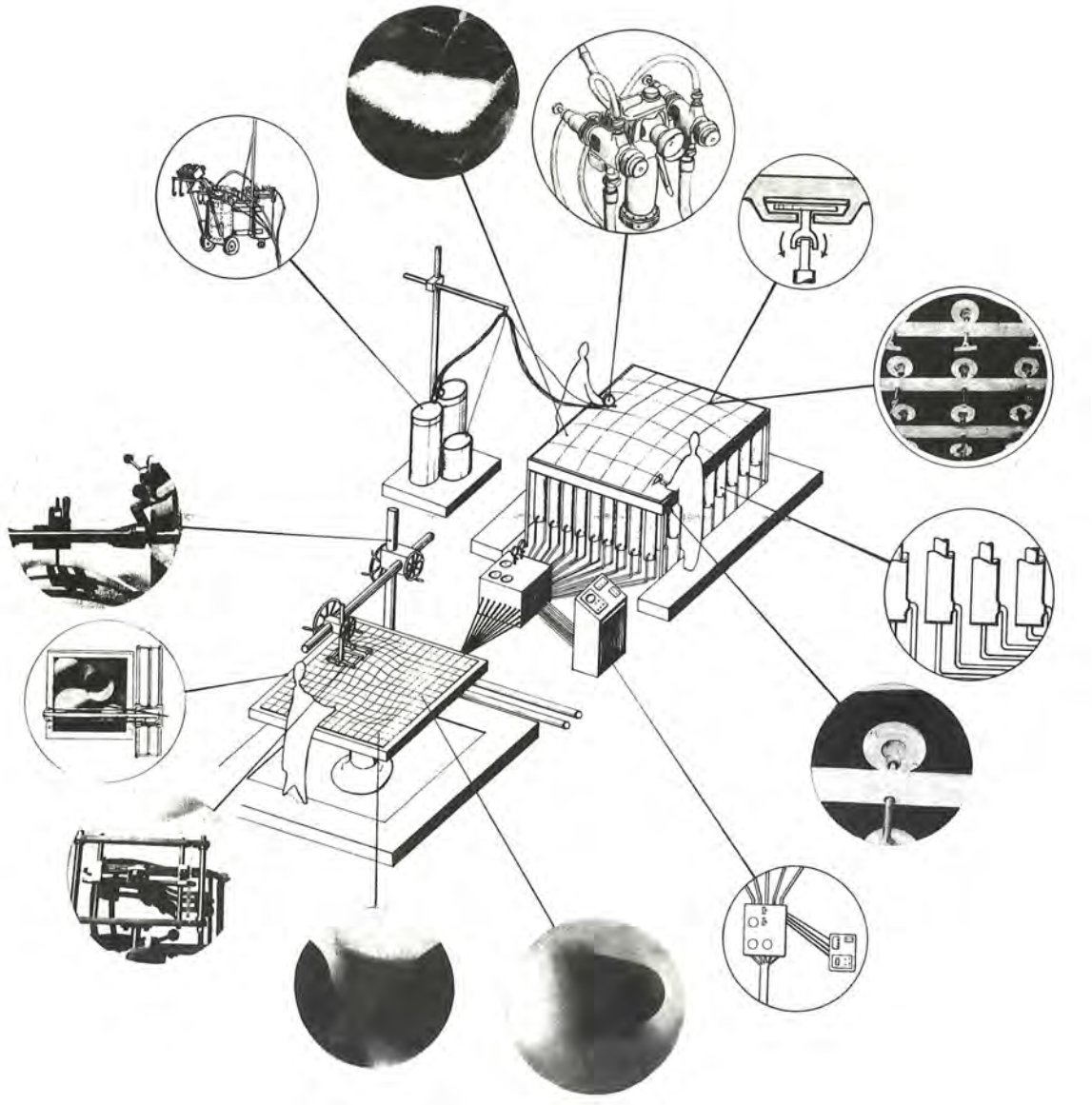


Figure 2-1: Sketch and details of a flexible mould by Renzo Piano. One of the first to describe a method for production of double curved elements. [Piano, 1969]

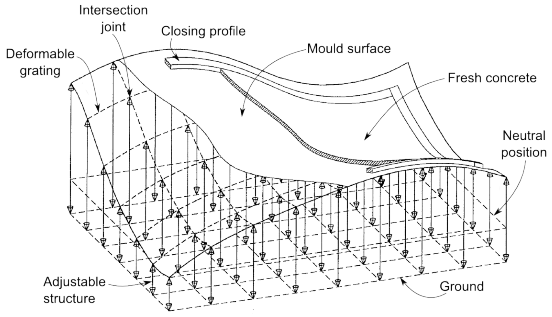


Figure 2-2: Drawing of flexible mould patented by Florian-Peter Kosche. Modified from [Kosche, 1998].

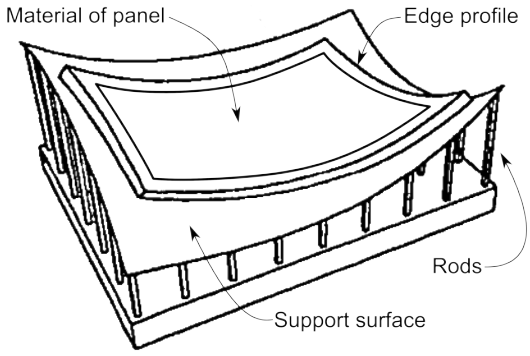


Figure 2-3: Modified drawing of flexible mould patented by Karel Vollers and Daan Rietbergen. Modified from [Vollers and Rietbergen, 2010].

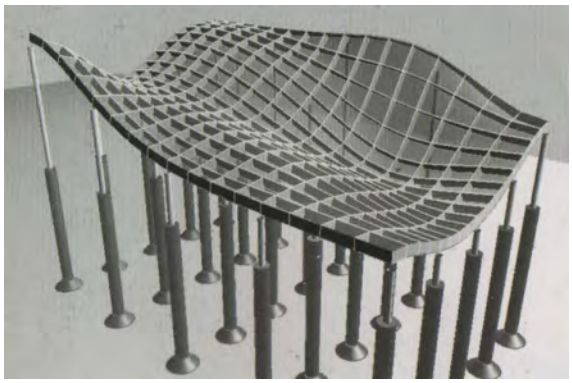


Figure 2-4: Flexible mould as described by Lars Spuybroek [2004].

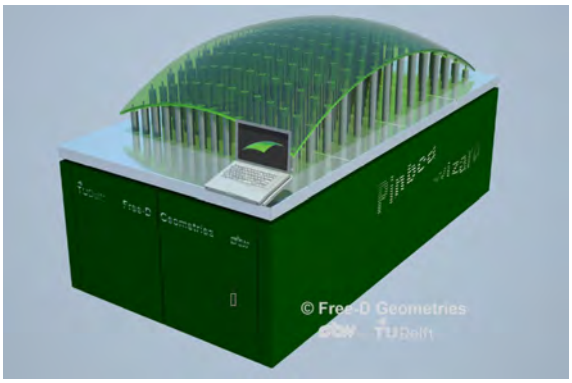


Figure 2-5: The "Pinbed Wizard" developed by Karel Vollers [2013].

2-1 The concept of the flexible mould

The concept of a flexible mould is very simple: A flexible surface or membrane is forced into a required shape. This can be any shape and is also referred to as freeform. The concept becomes more difficult when it is thought of how the surface is forced into a shape. For production of concrete elements the surface should be flexible but stiff enough to carry the weight of the concrete. For example it can be made of a thick rubber sheet or a wooden or plastic plate. The surface can be forced into shape by adjusting support heights which are in a grid supporting the surface. Moulds of this type have been can be seen in the Figures 2-2 and 2-3. These are patented respectively by Florian-Peter Kosche [1998] and Karel Vollers and Daan Rietbergen [2008]. Different approaches to adjust the shape of the surface have been chosen. Lars Spuybroek has described his idea of the concept and is shown in Figure 2-4 [Spuybroek, 2004]. Recently Karel Vollers introduced a machine called the "Pin bed Wizard". This is a computer controlled machine which adjusts numerous pins to required height. However for each material, a specific flexible surface must be made, that will be placed on top of the pins [Vollers, 2013]. The concept of a flexible mould can be used for different materials like plastics, glass and concrete. So far there is no commercial application

for concrete elements. This research aims to bring this one step closer.

2-2 Buildup of the flexible mould at the start of this research

The flexible mould as used in this research consist of a flexible surface with underlying grid of supports. In Figure 2-6 to 2-11 the different parts of the mould are shown. An important earlier development is the use of strips in the mould surface. In previous research several models have been made where the surface consisted of an solid plate. Tests by Bas Janssen showed local buckling of the surface after deformation [Janssen, 2011]. Therefore the required shape was not reached.

In Figure 2-6 a base plate with holes is shown which is used to fix the supports. A grid of holes determines the locations of the supports. The base plate is also the reference for adjusting the height of the supports. Therefore it should be sufficiently stiff. Would it deform, then its deformation directly influences the shape of the element.

In the holes the threaded ends are fixed by bolts on both sides of the base plate. The bolts are fastened hand tight which result in a connection that fixed but not rigid. A rigid connection could cause jamming of the mould during deformation. A very loose connection would allows too much horizontal displacement of the threaded ends.

Figure 2-7 shows aluminum tubes that slide over the threaded ends. An extra bold is put on each threaded end. This bold is used to set the height of the supports. During deformation of the mould tubes slide over the threaded ends until they reach the set height by the bold.

Figure 2-7 and 2-8 show an additional plate and temporary support structure which keep the supports in horizontal position. This is used when the concrete is cast on the mould. Later the temporary supports are removed and the mould is lowered by an crane, as shown in Figure 2-11, until all tubes rest on the bolds.

Figure 2-9 shows the surface of the mould. Strips span the distance over the supports, but are not connected. Only nails through the lower strips slot in the tubes. On top of this another layer of the strips spans in perpendicular direction creating the mould surface. The two layer are only connected at the middle lower strip. Therefore the two layer can slide over each other.

Figure 2-10 shows a sheet of foam and silicon is placed on the strips. Edges made of the same material and are fixed on the sheet using sealant. For testing they are placed in straight position. Concrete can be cast directly on the sheet. The foam is compressed by the weight of the concrete and compensates small gaps between the strips.

2-3 Setup and production procedure

Production of double curved concrete element requires two phases. An engineering phase and a production phase. During the engineering phase production specifications are determined. Examples of that are support heights, location of edges, element thickness, concrete mixture, drying times and element buildup in case of reinforcement. This research focuses solely on the geometry. The geometric description of a shape is required and is assumed to be provided.

Figure 2-6: Base plate with threaded ends fixed with both on both side of the plate. Bolds are hand tight. [Schipper, 2013].

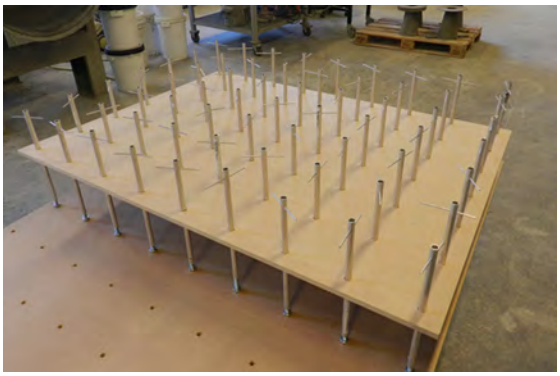


Figure 2-7: Tubes slide over the threaded ends. An support plate can keep supports leveled when concrete is cast. [Schipper, 2013].

Figure 2-8: Temporary structure keeps the support plate at fixed height. The structure is removed when the mould is deformed. [Schipper, 2013].

Figure 2-9: Two layers of strips. The bottom layer of strips is placed directly on the tubes. The top layer is only fixed to the lower middle strip with nails. [Schipper, 2013].

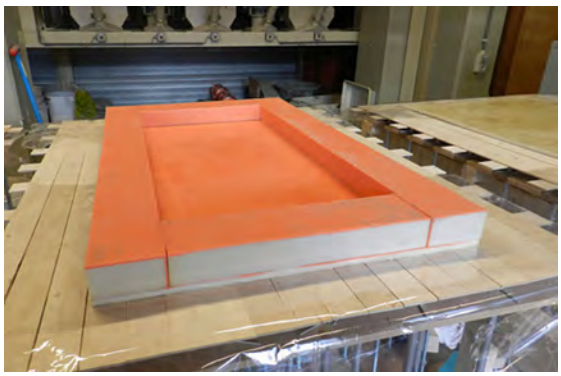


Figure 2-10: Foam with silicon mould to contain the concrete. Bottom and edges are made separately. [Schipper, 2013].

Figure 2-11: Finished mould. A crane lowers the support plate. [Schipper, 2013].

Figure 2-12: Fresh concrete is cast into the formwork until the required thickness is reached. [Schipper, 2013].

Figure 2-13: After the concrete has sufficiently hardened the temporary supports are removed and the surface is lowered into its required shape. [Schipper, 2013].

During the engineering phase the support heights be can determined in several ways. For shapes that can be described by mathematical functions, such as cylinders and spheres, support heights can be calculated. However the mould will produce freeform shapes. The use of geometry defining software provides the capability to analyze freeform shapes and obtain support heights according to the support grid. This is the starting point. Deviations occurs during deformation of the mould. These are caused by both mechanical and geometric effects. The resulting deviation can be predicted and compensated for by adjusting the support heights. Important factors that influence the deviation and determine the support heights are:

- Positioning and orientation of shape in mould
- Mechanical and material properties of mould surface
- Displacements and rotations due to deformation

When the engineering phase is completed the production of the elements can be started. The support heights are setup by adjusting the bolds. This is a time consuming activity which could be automated in future. The formwork (foam and silicon bottom and edges) that will contain the fresh concrete is placed and aligned on top of the mould surface. Next the concrete is mixed according to specifications and poured into the formwork (Figure 2-12). Reinforcement can be placed by adding alternating layers of concrete and reinforcement. At this stage the mould is kept in horizontal position to prevent the concrete from flowing out of the formwork. A curtain period of hardening is required to stiffen the concrete mixture. The optimal duration of this period and its consequences are not fully known yet. When the concrete is sufficiently stiff the temporary support structure is removed and the mould surface is lowered by means of a crane until all supporting tubes reach their required height (Figure 2-13). The mould has now reached its final shape and the concrete is left to harden. A foil is used to cover the concrete preventing fast evaporation of water in the top surface. After one day the element can be removed from its formwork and the process can be repeated. It takes

about 28 days to fully harden the concrete of the elements. Controlled humidity conditions are preferred for the hardening process.

The buildup of the mould, using simple materials and connections has curtain advantages. The simplicity of the buildup is the strong point of the mould. It is robust and parts of the mould are unlikely to be damaged or fail. It became clear that production of concrete elements is a messy job. For example concrete, water and sand is likely be dropped outside of the mould unintentionally and production of prefabricated concrete is mostly done in a dusty environment. A setup using automated pistons could jam due to sand entering in slits between moving parts. Additional measures would be needed to prevent material entering the slits which make the pistons sensitive and could lead to a unpractical working situation.

2-4 Shortcomings of the mould

The mould as used at the start of this research (described in Section 2-2) has been build to satisfy research requirements and within a limited budget. As a result there are multiple issues that could be improved. This would provide better control on the shape of the mould and increase accuracy. Some factors that influence the shape cannot be controlled and introduce inaccuracies. When accuracy needs to be determined uncontrolled influences need to be limited as much as possible. The mentioned factors have been noticed and observed during production of elements will be described here.

Deflection of the mould is caused by the weight of the concrete and is in some cases insufficient. No other force acts on the mould. The strips in the surface resist to the deformation. Extra weight can be added manually during the production process. Or in case of single curved elements the edges can be clamped. An additional factor is that the concrete needed for one element will not cover the total surface of the mould. Therefore the surface is unevenly loaded which also introduces inaccuracy. Figure 2-14 and 2-15 shows examples of these situations. The position of the added weights influence the shape and introducing a inaccuracy in production. For example there is a difference in adding point loads or distributed loads over the length of the strips.

This indicates, as can be expected, that the stiffness of the mould surface is an important aspect that influences the shape of the mould. Similar results where noticed by Koen Huyghe and Arnoud Schoofs on an alternative version of a flexible mould showing the importance of the stiffness of the mould surface [Huyge and Schoofs, 2009]. Calculations on the stiffness of the strips of the mould performed by Bas Janssen [2011] pointed out that a low stiffness would result in too large deflection between the supports. A high stiffness would cause detaching of the strips and supports, as was now observed during the tests. Therefore it can be concluded that the stiffness of the strips was too high for the applied load. This could be solved by reducing the stiffness of the strips or increasing the load. As described the later was applied during tests. This is considered a practical solution since changing the stiffness would require to rebuild the mould surface. In some situation it is needed to apply additional loading, but the applied method does not provide enough control over the surface. It would be favorable to find a alternative method that fulfills these requirements.

The number of strips in the two perpendicular directions differs. Therefore the stiffness of the surface for these two direction is also different. A result is that the strips deform different

Figure 2-14: Element ready for deformation. Bricks are added around the element to add weight to the surface. Weights are needed for deformation if the mould. [Schipper, 2013].

Figure 2-15: Single curved elements after deformation. Weight of the concrete is insufficient for deformation. Clamps are used to fix the shape. [Schipper, 2013].

under the same load. Equal stiffness in two directions is therefore preferred.

The accuracy of the mould surface cannot be ensured. The surface of the mould consists of strips in two directions. There is a bottom layer and top layer of strips. The layers only connected at the lower middle strip. Therefore the strips are free to slide over each other. This was done as alternative for using a solid plate. Tests by Bas Janssen [2011] showed that the edges of a solid plate buckle and therefore the required shape was not obtained. Since the plate has been replaced buckling has not been observed in tests. The strips are separated with small distance. There is room for in-plane deformation and stresses have not (yet) build up to a level that cause buckling. As mentioned before the strip are not connected and displacements cannot be controlled. Figure 2-18 shows uneven displacements of the strips. This creates a silhouettes of individual strips in the surface of the concrete element as can be seen in Figure 2-19. In theory neighbor strips deflect almost the same. In practice this does not happen. Uncontrolled displacements cause that accuracy over the mould cannot be ensured. An alternative solution that allows control over the displacements would increase accurate deformation of the mould.

The absence of a good connection between the supports and strips causes that accuracy of the shape cannot be controlled. The connection allows for detaching of the mould surface and uncontrolled displacements and rotations. The connection is made using a nail that slots in the supporting tube. The diameter of the nail is far less than the inner diameter of the tube. Figure 2-17 illustrates a situation where the strips do not deflect enough to touch the supports. This can cause the nail to pop out of the tube. Deforming and afterwards returning the mould surface to horizontal position causes fierce shaking of the mould. Figure 2-16 shows the connection after repeating tests. The nail no longer slots in the tube. Therefore large horizontal displacements are no longer prevented and accuracy cannot be ensured. A connection that allows controlled displacements and rotation would increase the accuracy of the mould.

Rotation of the strips cause a deviation of the required shape. The thickness of the mould creates a curtain offset from the top of the supporting tubes. In horizontal position this can easily be compensated by reducing the height with the thickness of the mould surface.

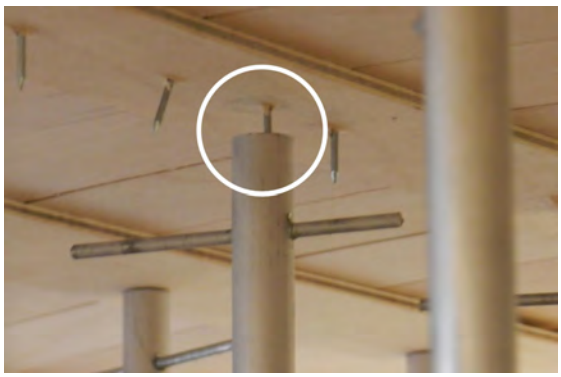


Figure 2-16: A situation during tests where the strip does not deflect enough to touch the support. As result the nail does not fully slot in the tube.

Figure 2-17: After repeating tests the nail no longer slots in the tube. Large horizontal displacements are no longer prevented.

Figure 2-18: Strips deflect unevenly causing an uneven surface of the concrete element. [Schipper, 2013].

Figure 2-19: Unsmooth surface of concrete element. [Schipper, 2013].

Figure 2-20: Strip at angle resting on support. At the center of the support the strips is lifted.

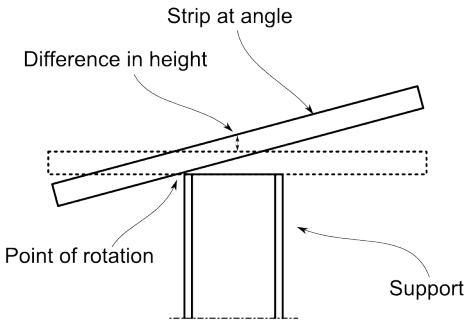


Figure 2-21: Vertical deviation from the height set by the support..

However when a rotation occurs the offset increases and the compensation is not sufficient anymore. The Figures 2-20 and 2-21 show a vertical deviation from the set height and should be compensated for when calculating the support heights. However currently this not included. The deviation is caused because the point of rotation is not the center of the support, but the outer diameter of the supporting tube. Also the thickness in vertical direction changes when the angle of the surface changes.

The height of the bolds are directly derived from the geometry of the required shape. However a deviation of the height is introduced by thickness of the strips and silicon resting on the supports at an angle. Since the tube has an curtain width the strip will tilt over the edge moving the surface at the center of the support.

2-5 The relation between change of Gaussian curvature and strain of the surface

During deformation of the mould the shape of the mould changes from a flat surface into a double curved surface. Due its 3-dimensional character it becomes difficult to precisely describe what happens in terms of displacements and strain in the surface. To describe properties of curved surfaces Gaussian curvature is well known. It has been described and studied by many people since Karl Friedrich Gauss describe already in 1827 a measure of curvature in his *General Investigations of curved Surfaces* [Gauss, 1827]. However the relation between change of Gaussian curvature and internal strain of the surface is less known. It was found that this relation plays an essential role in this research. This chapter describes this relation and the consequences for the flexible mould.

Gaussian curvature (k_G) is widely known as the product of the curvatures of a surface in two principal direction, which is a description in 3 dimensional Euclidean space. However there is an equivalent definition which can be thought of as 2 dimensional in the direction of the surface. Therefore it is possible to imagine the surface of the mould as a surface of infinitely small thickness and in-plane strain can be analyzed. This is an important step when trying to understand what happens within the surface of the mould during deformation. Both described definitions are given in Equation 2-1 and Figure 2-22 illustrates the equation parameters. A precise description is given by Calladine in Chapter 5 of his book *Theory of Shell Structures* [Calladine, 1983]. Three types of Gaussian curvature can be distinguished; zero, positive and negative.

$$k_G = K_1 \cdot K_2 = \frac{\mathrm{dB}}{\mathrm{dA}} \tag{2-1}$$

Changing the Gaussian curvature of a surface requires straining of the surface. The result of straining is that points on the surface change position relative to each other when distances are measured along the surface. This is illustrated in Figure 2-23. The Gaussian curvature is increased in steps. It can be seen that the corner part of the surface changes into a diamond shape. This is called shear deformation. Other strain, but less visible, is the change in length of the part edges. Change of Gaussian curvature (G) involves these three components. A change of length in two direction (ε_{xx}, ε_{xx}) and shear (γ_{xy}). The relation between the change

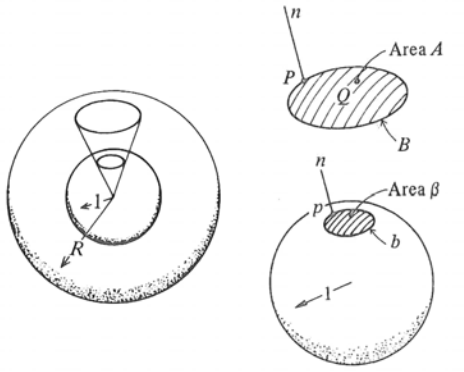


Figure 2-22: Mapping of a surface onto a unit sphere. Used for defining the measure of curvature [Calladine, 1983].

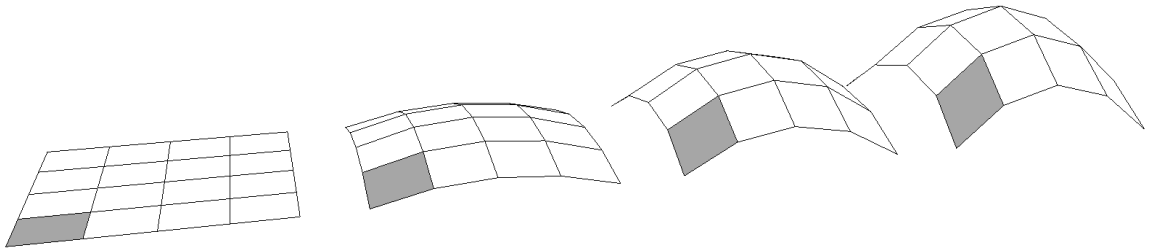


Figure 2-23: Deformation of surface.

of Gaussian curvature and strain is described in Equation 2-2. A derivation of this equation is given by Calladine in Chapter 6 of his book [Calladine, 1983]. It can be seen from Figure 2-23 that shear deformation is a substantial contribution and that it can even by itself cause a change of Gaussian curvature. The latter and the mathematical description may seem complicated. However the following example clearly illustrated their meaning;

Like in Figure 2-24, a flat sheet of paper can be divided in rectangles with a relative shear angle to each other according to $\gamma = C \cdot d^2$. Substituting in Equation 2-2 gives a constant change of curvature. When the squares are cut from the paper and the edges are fixed to each other and a double curved shape appears like in Figure 2-25. Since the paper is originally flat the Gaussian curvature is zero. The final Gaussian curvature is therefore equal to the change of curvature. By this example it is shown that change of Gaussian curvature involves a combination of strains. Only shear was considered in this example.

$$G = -\frac{\partial^2 \varepsilon_{xx}}{\partial y^2} - \frac{\partial^2 \varepsilon_{yy}}{\partial x^2} + \frac{\partial^2 \gamma_{xy}}{\partial x \partial y} \tag{2-2}$$

So far only straining is considered. And when changing the Gaussian curvature it is assumed straining can take place without any resistance. This experiment can only take place within our thoughts. In reality a surface will always consist of a material with a thickness and resistance to straining. Within the field of mechanics this relations (if linear) is known as Hooke's law. Straining of a surface introduces stresses. The material will resist a deformation. The result is that it is not possible to change the Gaussian curvature without using force.

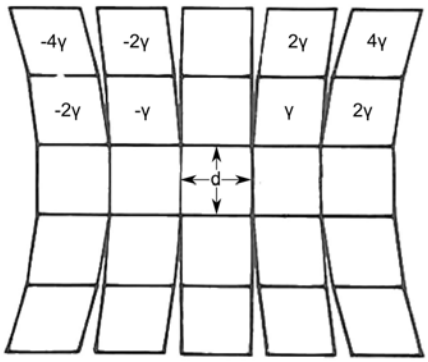


Figure 2-24: Strain of surface with shear angle according to $\gamma = C \cdot d^2$. Modified from [Calladine, 1983].

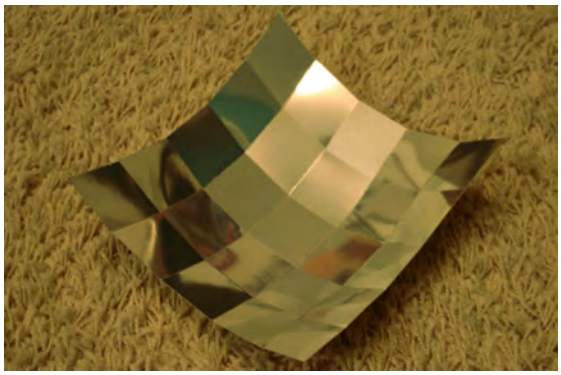


Figure 2-25: Double curved shape cut from a flat piece of paper.

The description of change of Gaussian curvature has consequences for the flexible mould. The flexible mould was designed for freeform shapes. Therefore deformation of the mould surface will always involve a change of Gaussian curvature of the surface. The deformation will therefore introduces in plane stresses. within a solid plate stresses can build up to high levels. For strips this is less, but can still be considerable. Strips are loaded in two directions. In their weak direction (perpendicular to the surface) and in their strong direction (in plane of the surface). In Figure 2-23 it can be seen that the edges are bend in the direction of the surface. In structural mechanics loading in two directions is known as double bending. Since shear deformation is an important factor it can be concluded that a mould surface is required that has low resistance to shear. Shear deformation involves a change of angle of initially perpendicular directions. Would the two layers of the strips be connected then the rotation should be allowed.

A important aspect is the translation of individual points on the mould surface. The above given example shows that points displace horizontally and vertically and also rotate in two directions. These are also called degrees of freedom. A connection between strips and supports should allow these degrees of freedom. Earlier it was found that these translations occur uncontrolled. Now it is shown that control of translations is essential to create a curtain shape.

2-6 Conclusions

A study was made of the flexible mould as it was used at the start of the research. The buildup of the mould was observed and several shortcomings were found. The accuracy could not be ensured due to uncontrolled displacement and rotations of the mould surface at the support connections. Control over these translations is essential for to increase the accuracy of the mould surface. Some details of the mould buildup cause deviations of the intended shape which should be compensated for. Also a study was performed on the deformation of the mould surface. Understanding the behavior of the mould during deformation is difficult due to its three dimensional character. It was found that during deformation the Gaussian curvature of the surface changes and that shear deformation has a large contribution to this.

The surface of the mould should be able to follow this deformation without resisting too much. The following aspect could be improved in order to gain more control over the geometry;

- An alternative method for applying additional loading of the surface
- A mould surface with low resistance to shear deformation
- Connections between two layer of strips that allows in-plane rotation
- Connections between strips and supports that allow horizontal and rotational translations
- Compensation method for deviations caused by horizontal displacements and rotations of the mould surface

Chapter 3

Design of a new flexible mould prototype

The accuracy of the concrete elements that will be produced with the flexible mould depend on the design of the mould. But the accuracy of the mould as it was at the start of this research could not be ensured in the version. Therefore a new mould prototype has been designed to improve control over the shape. This was done based on the results of Chapter 2. During the process of designing alternately a theoretical and a practical approach was chosen and multiple small scale tests have been performed to gain insight in the effectiveness of solutions. The test models have been made from everyday materials which can be bought in any hardware store. Figure 3-1 shows the design of the new prototype of the flexible mould. After the design was finished a prototype has been build which is shown Figure 3-2.

In this chapter the solutions will be introduced that improve control and accuracy over the shape of the flexible mould. The following aspects will be described; the flexible surface, the support connections and weights that can be added to the supports. The number of supports and size of a mould can vary according required size of the elements. For this research a mould surface of 630 x 630 millimeter is chosen with five supports in each direction. The distance between the supports is 150 millimeter.

3-1 Flexible surface

In an early version of the flexible mould the surface was made of a solid plate. Later the plate was replaced by two layers of strips in perpendicular direction. This was done after tests by Bas Janssen. In these tests local buckling was observed at the edges of the plate [Janssen, 2011, Chapter 10.3]. The strains and accompanied stresses became too high and caused bucking. Buckling is an unwanted effect because it results in a deviation of the intended shape. Separate strips have an advantage compared to a solid plate. The strips can deform separately with limited exchange of in plane stresses because there is a separation between the strips. The stiffness of the individual strips is an important aspect that influences the

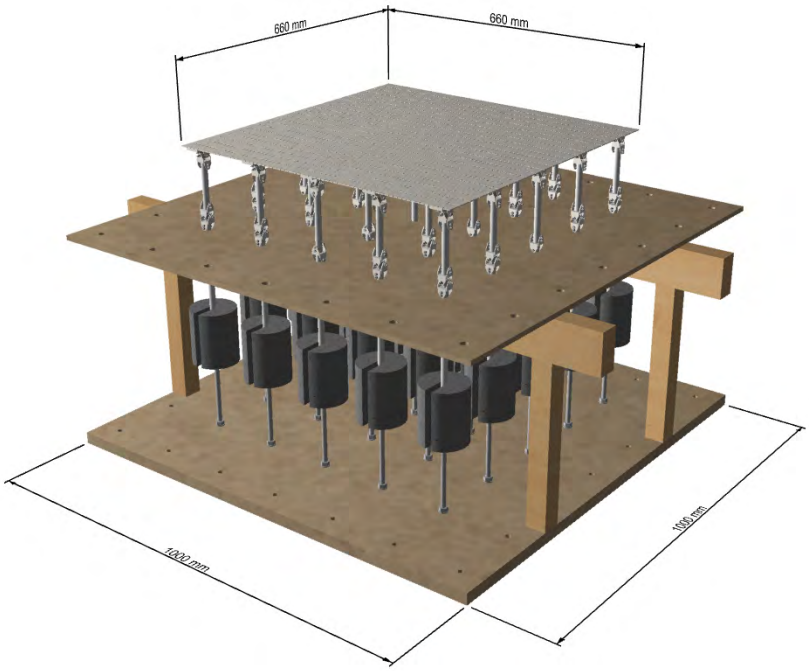


Figure 3-1: Design of the new prototype of the flexible mould.

Figure 3-2: Finished prototype of the flexible mould.

shape. This was described in Chapter 2. The strips in the previous version of the mould where too stiff in relation to the applied load. Also an important factor that was described is the absence of a connection between strips in perpendicular direction. The strips can therefore displace uncontrolled and the accuracy of the mould cannot be ensured.

In order to find a solution for these problems two ideas have been introduced. The first idea was to reduce the length of the strips and place multiple strips in line instead of one strips that spans the total surface or the mould. The shorter strips would span the distance between supports. Figure 2-23 of the previous chapter illustrated that strips make an angle relative to the next strip in line due to shear deformation. This can also be described as in-plane deformation. The connection of the strips should be such that an in-plane rotation allowed, but rotation out of plane is still transferred. The latter was required for continuous curvature of the strips. The resistance of the surface to shear deformation would be reduced significantly by making this type of connection. A solution that could fulfill the requirements was a multiple layered strip which partly overlapped with the next strip in line. When wooden strips would be used the thickness of the strips would become too high since layers of approximately 0.5 millimeter would be hard to obtain. Therefore the material was replaced with plastic which is available in this thickness. Figure 3-3 and 3-4 show two small scale versions of the connection. The layers of strips where glued together. The overlap and connecting bolt transferred a moment and since there is only one bolt there is a hinged connection in the in-plane direction. After testing the alternatives it was found that the connection was not applicable for the flexible mould. This was caused by the following aspects;

- Connection showed discontinuous curvature which was caused by local decrease of moment of inertia at the connection
- Permanent deformation of the used material
- Layers detached because the glue was not strong enough
- Fabricating of the strips took a lot of work
- A countersunk hole to fit the bolt was not possible because the material is to thin and the hole would further decrease the stiffness of the connection

The small tests showed multiple difficulties but the concept could be optimized further for better results. However the connection in this concept was likely to always show a discontinuity in curvature. This was caused by local decrease of stiffness at the connection. The overlap in the connection has no interaction with the other layers and the moment of inertia is therefore different from a cross section in the middle of the strips. At that point the strips are glued together and have a good interaction. For these reasons the idea was abandoned and another idea was worked out.

The second idea that was introduced was using continuous strips, like in the previous version of the mould. But an important difference would be that the strips in two directions are fixed together at the centerline and the point of intersection using a bolted connection. Only one bolt was used for each connection so that it allowed for in-plane rotations relative to each other. The nuts and bolts where not fully tightened but enough for the strips to exchange moments. When the nuts and bolts would tightened to much friction would prevent in-plane

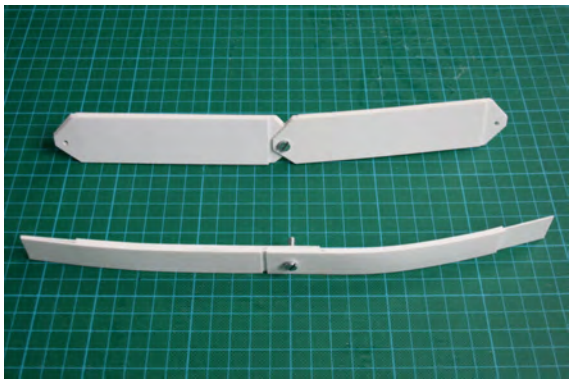


Figure 3-3: Two small scale tests of connection between strips.

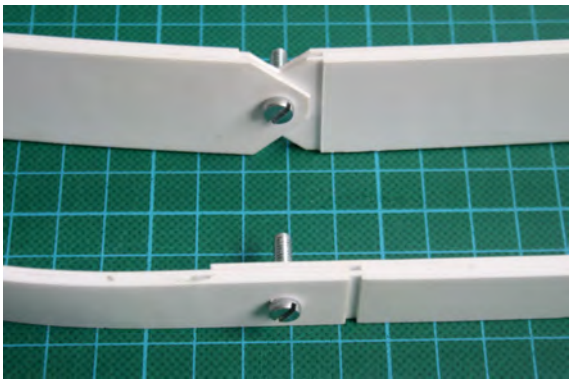


Figure 3-4: Overlapping connections of strips.

rotation of the strips. Self locking nuts where used to prevent the nut from detaching over time. Also the size of the strips was reduced to 29 x 3.8 mm. The imprint of the bolts in the concrete would will be reduced when the holes are countersunk in the top strip. This way the top of the bolt would be in line with the surface. A disadvantage of this is a local decrease of stiffness because the cross section is reduce. However when the hole is small the effect will hardly be noticed. Together the strips form a plate-like surface. This idea was worked out and can be seen in Figure 3-6. The result was a relative stiff surface. Therefore a great effort was needed to force the surface into shape. This was hardly an improvement compared to the previous version of the mould surface. Additionally, wood is a natural product. It is anisotropic, never exactly straight, does not fully return to its original shape and is sensitive to humidity conditions of the surrounding air. The shape of the strips in horizontal setup of the mould can be seen in Figure 3-7. Based on these findings it was concluded that the strips where still too stiff and that wood is not the ideal material for the mould surface.

A second version of the idea was worked out where four changes where implemented. The first change was a reduction of the thickness of the strips. It was reduced from 3.8 to 2.0 millimeter. Secondly, the wood was replaced with a plastic. Third, in the bottom layer two strips where placed on top of each other. This was done to compensate the difference in stiffness of the top and bottom layer of the surface. The strips where fixed by the bolted connection that also fixes the supports. The bolts where tightened strong. Thereby the interaction between the strips was increased. If the connection would be loose then the stiffness would double. But if the connection would be solid then the stiffness would increase by a factor eight because for the moment of inertia the height of the strips is taken into account to the power three. So the stiffness would have increased somewhere between a factor to and eight. The fourth change was that notches where made in the sides of the strips. The Figures 3-8 and 3-9 show small scale tests which were used to find an alternative for the wooden strips. The notches reduced the stiffness of the individual strips. From theory of mechanics it is known that for rectangular cross sections the moment of inertia takes the dimension of considered direction into account to the power three. For the strips in their strong direction this dimension is its width. As result of reducing the width by a half the stiffness reduced by a factor eight. The strips together formed the surface which will be subjected to shear deformation. The result can be seen in the Figure 3-10 and 3-11. Shear deformations was described in Section 2-5. The surface is required to have low resistance to shear deformation, so that large internal

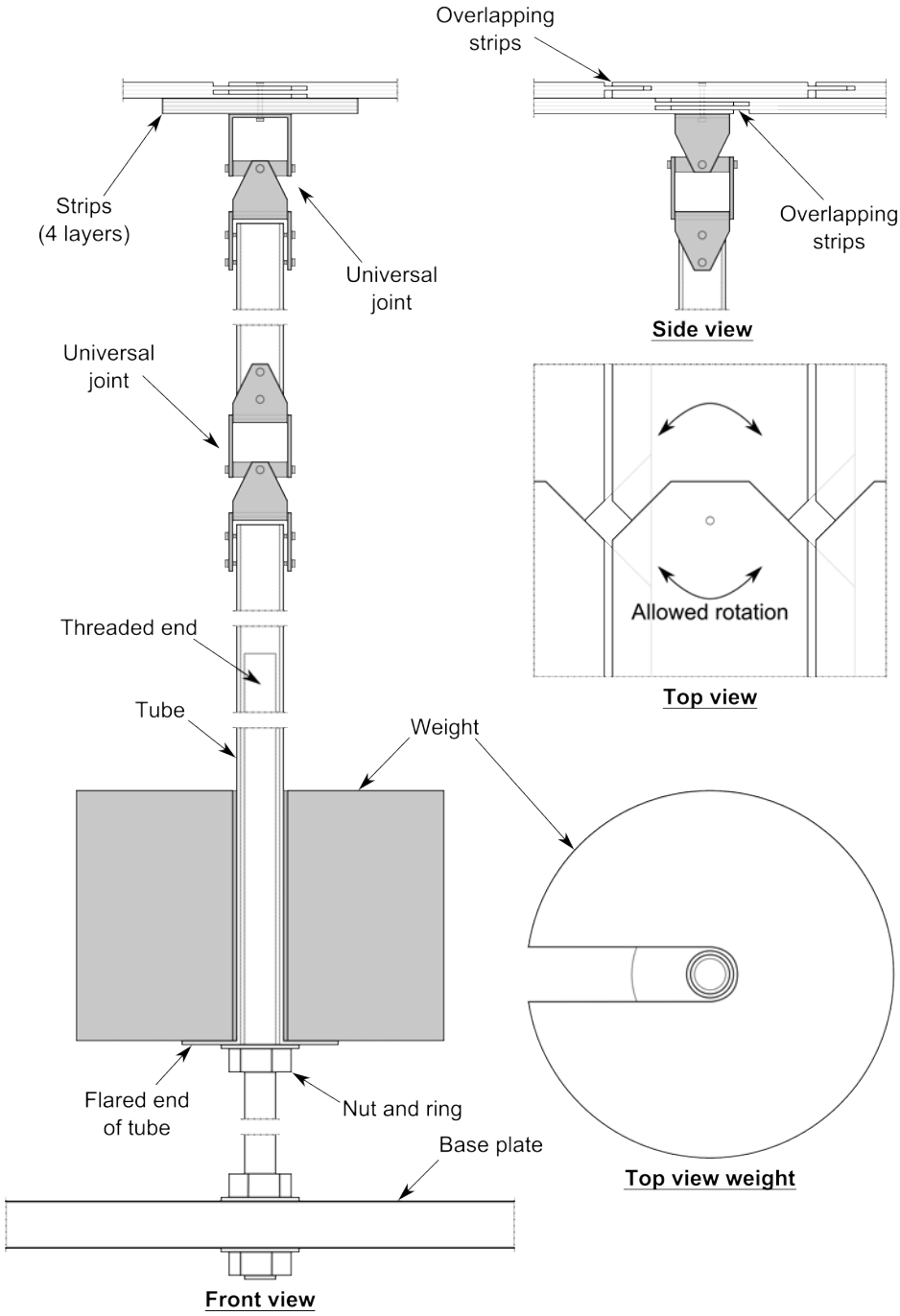


Figure 3-5: First design of the new prototype of the flexible mould

Figure 3-6: First version of a new surface of the mould. The surface consisted of wooden strips which are connected to each other by bolts.

Figure 3-7: The wooden strips are not straight as can be seen at the edge of the strips.

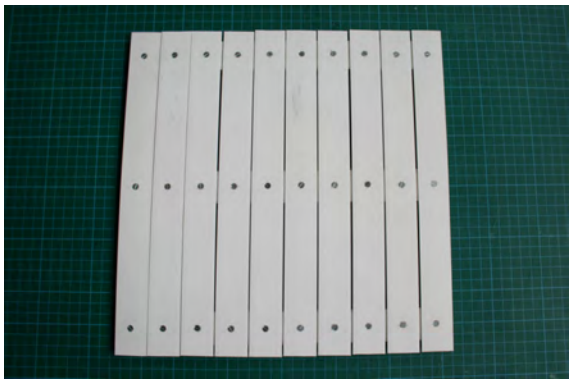


Figure 3-8: A small scale test in order to find an alternative for the wooden strips.

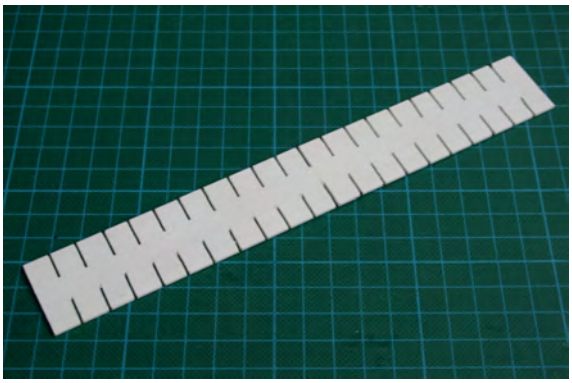


Figure 3-9: Small scale test of the strip with notches.

stresses are prevented. High stresses can cause buckling of the surface and a deviation from the intended geometry. Reduction of the strip width also reduced the resistance to bending deformation. Over the length of the strips the moment of inertia alternates because it is higher for cross sections that have not been reduced. This could result in discontinuous curvature if notches are unequally spaced and to far apart. To ensure that continuous curvature the notches where made at equal distance and depth. These two dimension are parameters that influence the stiffness of the strips can in the future could be used for optimization of the strips stiffness. By introducing the notches in the strips the mould surface should be able to deform more easily compared to a similar surface using strips without notches. Also deformation at the moment that buckling occurs will be larger. From these consideration is was concluded that the notches should make the mould surface very flexible.

3-2 Support connections

Strips and supporting tubes are not connected in the previous version of the mould. Therefore the strips can displace uncontrolled relative to the supports. Also negative support reactions

Figure 3-10: Plastic strips replaced the wooden strips and the thickness was reduced from 3.8 to 2.0 mm.

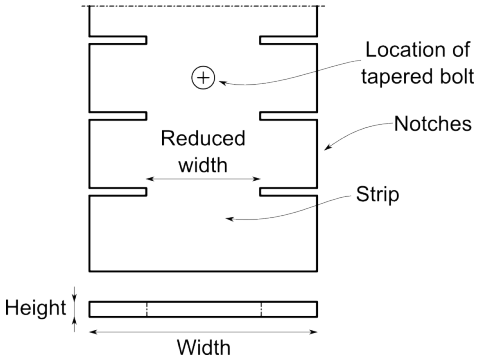


Figure 3-11: Top view and section of a strips. The notches in the strips should reduce the resistance to deformation of the whole surface.

cannot occur. Therefore the supports can only resist a force downwards which should fix the shape of the mould in curtain position. As result the weight of the concrete must be sufficient to cause positive support reactions on all supports. But for some shapes and thickness of the concrete this was not the case and additional weights had to be added to the surface. This is not ideal and therefore two solution where introduced which enabled positive and negative supports reactions and at the same time allow rotation of the strips.

The first idea was a connection using a couch bolt, a rounded block and a locking plate. This idea can be seen in the Figures 3-12 and 3-13. Here a couch bolt would slot in the supporting tube and would also be locked within a block and locking plate. The connection would be able to transfer negative reaction force because of the locking plate. The advantage of the rounded shape of the couch bolt would be that the strip rotates around a fixed point. The reason for the rounded shape of the block is that it would prevent local stiffening of the strip. Therefore the block should also be fixed with two bolts only perpendicular to the direction of curvature. Unfortunately there was still room for play in the connection which was not wanted. An alternative was therefore sought.

A new solution was found in two steps. Following the stated requirements a first version of a connection was made. This connection worked well. However rotations in different directions did not have the same point of rotation as can be seen in Figure 3-14. This would be unpractical in further research. Therefore a second version was made. The solution showed much resemblance to a universal joint and in fact is one. This connections finds its application in connections for axles and bars in machines which can be found in everyday equipment. For example a car. A stroke of inspiration dating from earlier days playing with Technic Lego was not excluded. Figure 3-15 shows the second version of the connection. The connection proved to work very good. It allowed for rotation of the strip and could transfer a force at the same time. Therefore it was implemented in the new prototype of the flexible mould.

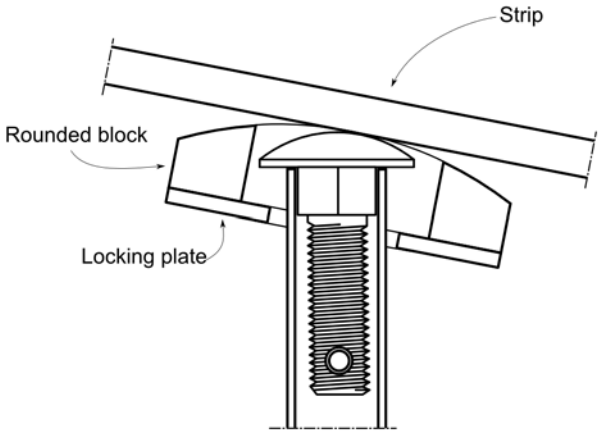


Figure 3-12: Section seen from the side of first idea for connection of strips and supports. The rounded shape of the block prevent local stiffening of the strip.

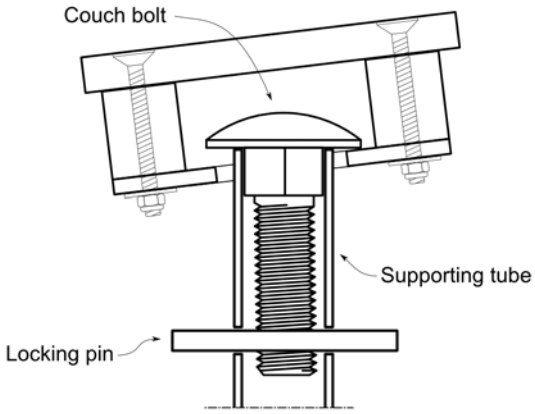


Figure 3-13: Section perpendicular to the strip. The locking plate enables transfer of negative reaction forces.

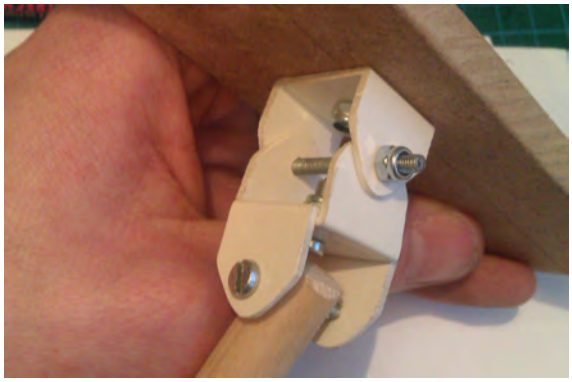


Figure 3-14: First version of a connection between strip and support.

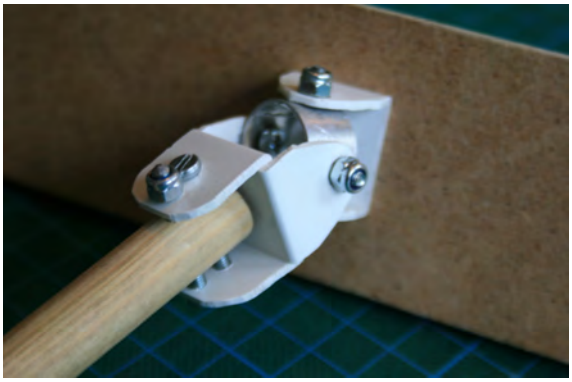


Figure 3-15: Second version of a connection. The connection is a universal joint.

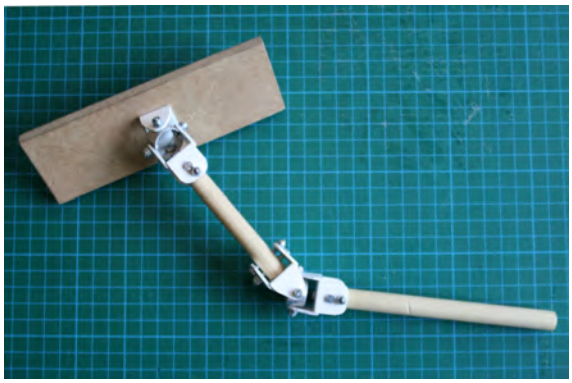


Figure 3-16: Two universal joints are combined to create a three dimensional pendulum support. This allows the strip to displace in all required degrees of freedom.

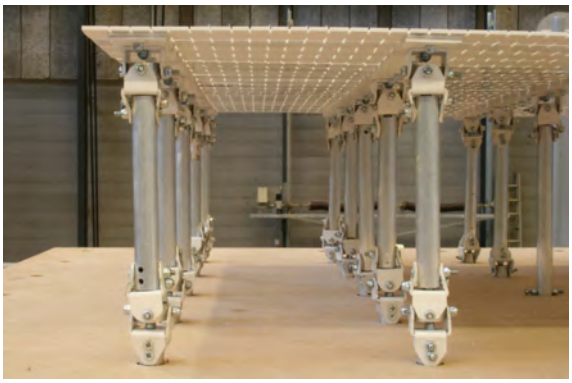


Figure 3-17: Implementation of the pendulum support in the new prototype of the flexible mould. Jamming of the mould is prevented.

3-3 Pendulum supports

The surface of the mould will be deformed during the production of the concrete elements. Therefore the supports are required to displace horizontally. A horizontal reaction force would be the result if the displacement is not allowed by the connection of the strips and the supports. This could jam or damage the mould. Also the required shape would not be reached. Therefore a solution has been introduced that allows horizontal displacement, but remains capable of transferring a force through the supports. A additional universal joint was added to the supporting tubes. Thereby a three dimensional pendulum rod was created which allows all the required degrees of freedom for displacements and rotations. A small scale model can be seen in Figure 3-16. This pendulum could displace freely while the lower part of the support remains in the same location. The interaction of forces between the surface and support is in line with the pendulum support. Only a normal force can be taken by this part since the universal joints do not transfer any bending moments. A small horizontal reaction force is taken by the lower part of the support due to the angle of the pendulum support. The angle is small, but will increase if the mould dimensions increase. A possible solution could be to increase the length of the pendulum support. Thereby the horizontal displacement cause a smaller angle. Due to the pendulum supports jamming or damage would be prevented. All parts of the mould where now connected. As result, uncontrolled displacements and rotations did not occur. Height of the supports could be set accurately and points of rotations where known to be the center of the universal joints. Like in the connection between strips and supports, the reaction forces could be transferred and at the same time all the required degrees of freedom are allowed by the combination of the universal joints. Figure 3-16 shows a model that was made to test the pendulum support.

The surface of the mould would not be stable if all supports would be made using the pendulum support. This is because the connections would allow displacements and rotation of the total surface. Therefore the middle support was not made using a pendulum support. Only a universal joint was used at the top. This prevent the surface to displace in horizontal direction and the middle support became a reference point. However rotation around this

Figure 3-18: Two types of pendulum supports. The left support allows displacements in two direction. The right connection only one direction.

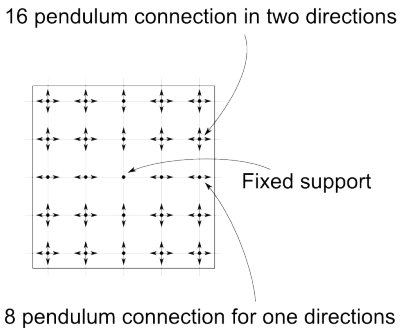


Figure 3-19: Allowed direction of degrees of freedom by the pendulum supports. Rotation of the surface is prevented by a combination of two type of pendulum supports.

support was not yet prevented. Therefore a second type of pendulum was introduced which could only rotate in one direction. This was done by using a simple hinged connection instead of a universal joint at the lower connection of the pendulum. The top connection remained the same because rotation of the strips in all directions need to be allowed. The two different types of pendulum supports can be seen in Figure 3-18. By applying these two of pendulums the surface could be made stable while all required degrees of freedom remain. This was done by placing the two types of pendulum supports in specific locations. The supports in line with the middle support have been made of the second type of pendulum support so that they can only displace towards the middle support. Rotations around the middle support are thereby prevented. In theory only one supports would be required to be of the second type. However in the connections that where made is some play. Therefore all supports in line with the middle support have been made using the second type. The location of the two types of pendulum supports is illustrated in Figure 3-19.

3-4 Additional weights for the supports

The support connections are capable of transferring forces. However, until now, no means of applying the force has been introduced. In some cases the weight of the concrete would be sufficient to lower all the support to their required height. But in some cases additional loading would be required because the surface resists to the deformation. Additional loading was applied by adding weights to the support. The shape of the weights was designed such that they can easily be added and removed according to the required location. For that purpose the weights where provided with slots as can be seen in Figure 3-20. The weights where made of concrete and their weight is approximately two kilogram. At the bottom of each supporting tube a ring was fixed which is larger than the slots in the weights. The tube where flared at the bottom. This made the outer diameter of the tube larger than the inner diameter of the used ring. Therefore the weights can be placed on the supports. This detail is shown in Figure 3-21.

Figure 3-20: Weights that can be added to the supports when the weight of the concrete is insufficient.

Figure 3-21: Detail of the bottom of the supports. The end of the supports have been flared. An additional ring provides support for the weight.

3-5 Conclusions

Based on the possible improvements which were found and described in Chapter 2 a design was made for a new prototype of the flexible mould. During the process of designing small scale tests were performed to test the functionality of new ideas. After finishing the design a version of the new prototype has been build. In this prototype solutions have been introduced which should improve control over the shape of the surface and increase accuracy of the mould:

- A flexible surface consisting of plastic strips in two perpendicular directions fixed together using a bolted connection. Notches in the strips reduce stiffness of individual strips. Therefore the resistance to bending and shear deformation of the surface should be reduced.
- In-plane rotation between initially perpendicular strips is allowed, so that shear deformation is made possible.
- No sliding parts are present in this design, making the kinematic predictability of the whole system much clearer. There are no uncontrolled translations.
- Support connections using universal joints allow rotation of the surface while positive and negative supports reactions can be transferred.
- Three dimensional pendulum supports allow horizontal displacements of the supports. Again positive and negative support reactions can still be transferred. The total surface is stable. Rotations and displacement as a whole are prevented by two types of pendulum supports in specific locations.
- Additional weights for the supports provide means to create negative support reactions. The weights can be placed and removed easily according to the required location.

Chapter 4

Method for setup of the flexible mould

When elements can be produced first the data from the geometry must be processed. This is needed to determine the setup of the mould. With this is meant that heights of the supports need to be determined. The geometry cannot directly be used because deviation occur during deformation of the mould. This was described in Chapter 2-4. These effect should be compensated for. In this chapter a method will be described which can be used to prepare the setup of the support heights of the flexible mould. The goal of the method is to provide support heights which would enable the manufacturer to obtain a surface which is as close to the intended geometry as is needed. Within this method mechanical and kinematic aspects are important. The method focuses on a single strips and can be extended for use of the flexible mould. First mechanic and kinematic aspects will be discussed. Then the proposed method will be described.

4-1 Mechanics

The mould will be deformed during the production process of the elements and the deflection can be calculated by structural calculations. Although the mould is small compared to the structure of a building or a bridge, the same calculations can be performed.

In earlier research by Bas Janssen [2011] the strips in the mould have been compared to a beam on multiple supports. This approach was continued. However there are multiple factors that complicate the description of the deformation of the mould:

- Three-dimensional character of the deformation including double bending
- Non-symmetric and discontinuous cross section properties of the deformed strips
- Assumption of small displacements are not fulfilled
- Complexity in large number of equations to set up for each strip

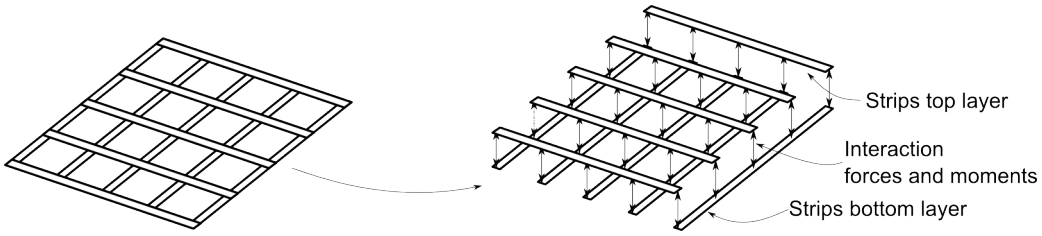


Figure 4-1: Model of mould surface for setup of structural calculation.

- Unknown support heights outside element area

Because of the complexity of the deformation many factor have been simplified. And some assumption for theory of mechanics are not fulfilled. But this does not mean the comparison cannot be made at all. Results of a simplified approach may not be exact, but could be accurate enough. Therefore this approach was continued.

4-1-1 Bending equation

For structural calculation the fourth order differential equation for bending has been used. There are many references that can be made for this theory. Within this research lecture notes by Coen Hartsuijker en Hans Welleman [2011] have been used. The used equation have been derived and can be found in the Appendix A.

At first this method makes use of the equations for deflection and rotation. Later also the reaction forces and calculated. The equations for deflection, rotation and support reactions according to the coordinate system in Figure 4-2 are;

$$w(x) = \frac{q(x) \cdot x^4}{24 \cdot EI} + C_1 \cdot x^3 + C_2 \cdot x^2 + C_3 \cdot x + C_4 \tag{4-1}$$

$$\varphi(x) = -\frac{q(x) \cdot x^3}{6 \cdot EI} - 3 \cdot C_1 \cdot x^2 - 2 \cdot C_2 \cdot x - C_3 \tag{4-2}$$

$$R(x) = V_{left}(x) - V_{right}(x) \tag{4-3}$$

There were lots of strips in the mould surface. To make the calculation less complex in quantity only the strips above the supports have been considered. The surface has been modeled according to Figure 4-1. It was not required to perform structural calculations for every strips to gain insight in the shape of the mould. The strips have been given equivalent stiffness properties for the part of the surface they represent. To solve the equations a boundary condition had to be found for every unknown constant. There were many equation to solve. These can be found at the supports according to Figure 4-3. There was a difference between supports at the end, where two boundary conditions can be found, and supports in the middle of the strips, where four boundary conditions can be found.

A distinction was made between the top layer and the bottom layer of strips. First the top layer was calculated. The resulting reaction forces where then used as load for the bottom strips. An equally distributed load could not be used because the support reactions vary

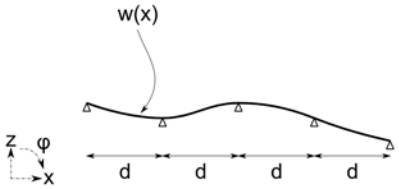


Figure 4-2: Scheme of strip. For single bending the deflection w of one strips is modeled as a function of the horizontal distance x.

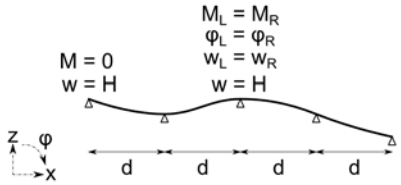


Figure 4-3: Boundary conditions to solve a set of equations. Different boundary conditions can be found at end supports and middle supports.

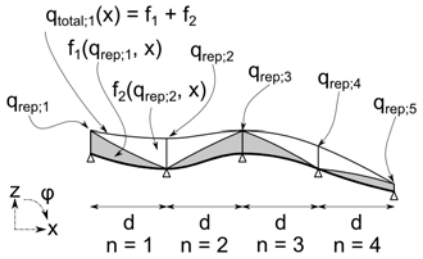


Figure 4-4: Load functions of the strips.

over location of the mould surface. However the reaction forces could not directly be used as load for the bottom strips because the forces would be above a support and no deflection would be caused by this load. To simulate the interaction with the strips that were not modeled according to Figure 4-1 equivalent load functions where calculated from the support reactions of the strips in the top layer. For each support reaction a triangular load function was setup. Then for each part of the strip a summation of the loads was used as input for the equations. This is illustrated in Figure 4-4. The equation was written so that only the value of the load above the supports needed to be chosen. This value was calculated using the support reaction of the top layer of strips. The equations are linear. Therefore the equations for the deflection become a fifth order polynomial instead of a fourth order when an equally distributed load would be used. In Equation 4-4 the factor thousand was taken into account due to the measurement in millimeters.

$$q_{total;n}(x) \quad = \quad q_{rep;n} \cdot \left(\frac{n \cdot 1000 - x}{d} \right) + q_{rep;n+1} \cdot \left(\frac{x}{d} - n \cdot 1000 - 1 \right) \tag{4-4}$$

4-1-2 Calculations for strips

When calculations for the strips where performed some aspect where noticed that are specific for application of the flexible mould.

First the Young's modulus of the used material must be known. However the used material was bought in a hardware store and the Young's modulus was not known. The Young's

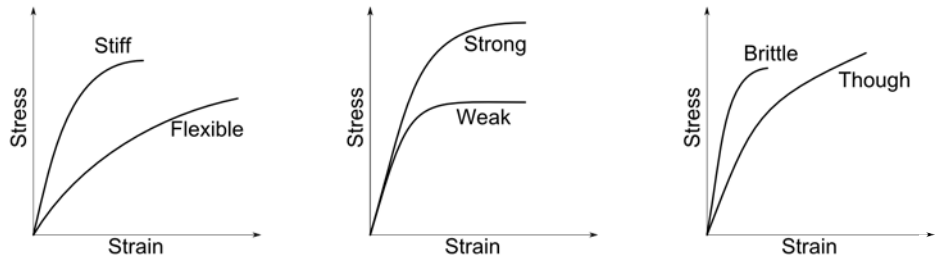


Figure 4-5: Various possible material properties expressed in σ-ε-diagrams for plastics. There is no linear clear elastic zone. Adapted from [van der Vegt, 1994].

modulus has be derived from a σ-ε-diagram obtained by tests. During these tests the material was loaded by weights and the train of the strips was measured. The setup and results of these test can be found in Appendix C. Figure C-1 shows a various possible material properties in σ-ε-diagrams for plastics. Here can be seen that plastic does always not have a linear elastic zone. However if the strain would be limited to low values an approximation could be made of quantity that could be used as if it was the Young's modulus of the material. This would fit research requirements although it is theoretically not correct. Three test showed an average value of 3851 N/mm^2 for low strain. This value was used in the performed calculations. In calculations it was assumed that the material of the strip was homogeneous and linear elastic. However the used material, plastic, is not. This could lead to a deviation in the shape.

A second important aspect was described in Section 3-1. There is a difference in stiffness for the strips in the top and bottom layer. The modeled strips represent there equivalent part of the surface. Therefore the strips in the middle represent the same distance as the support distance. However the edge strips represent only half of that. This was implemented in the performed calculations.

Third, during deformation of the mould displacement and rotations occur in multiple direction. However in the calculation only translation in the x-z-plane of the strips are considered. This is not in accordance with the actual deformation. This phenomenon is called double bending. Similar calculation as for single bending can be performed but are not implemented in this research. However the required equations have been derived and can be found in Appendix A. The equation have been modified from lecture notes by Coen Hartsuijker en Hans Welleman [2011]. Double bending should be considered if single bending would greatly influence the made calculation error or for if scientific reasons it is strove for more accurate results. However calculation become more complex. In the calculation during this research double bending is not included. First result of single bending are researched. Another result of translation in multiple directions was that the orientation of the strips change. Therefore cross sectional properties change too. The moments of inertia of a rectangular cross section has been written as a function of the angle as in Figure 4-6 and 4-7. It can be seen from the following the equations 4-5 to 4-7 that for an angle of zero degrees the well known function for the moment of inertia appear.

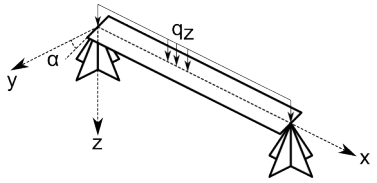


Figure 4-6: The orientation of a strips has changed due to an angle α.

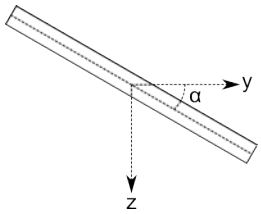


Figure 4-7: Cross section of a strip. The moment of inertia has changed.

$$I_{yy} = \frac{1}{48} \cdot w^3 \cdot h + \frac{1}{16} \cdot w^3 \cdot h \cdot cos(\alpha)^2 \tag{4-5}$$

$$I_{yz} = \frac{1}{16} \cdot w^3 \cdot h \cdot sin(\alpha) \cdot cos(\alpha) \tag{4-6}$$

$$I_{zz} = \frac{1}{12} \cdot w \cdot h^3 + \frac{1}{16} \cdot w^3 \cdot h \cdot sin(\alpha)^2 \tag{4-7}$$

A fourth aspect that was noticed is of importance of the accuracy of the mould surface. Calculation where performed for shapes with continuous curvature. From the used boundary condition at the end supports of the strips it followed that there was no bending moment. From theory of mechanics follows that the curvature will also be zero. But the required shape has continuous curvature. A deviation of the shape was the result. In Figure 4-8 the intended and calculated shape are shown when five support are used. Here can be seen that near the ends of the strip there is a deviation while in the middle it cannot be seen. This can be explained by the presence of moments. Figure 4-9 shows the difference between the intended shape and the calculated shape. At the point of the supports the deviation becomes zero. A clear distinction can be made in the deviation of the near ends and the middle of the strips. The effect can also be seen when the number of support increases. This could be solved by creating means to introduce moments at the end supports. However this would make the buildup of the mould unnecessary complicated. It is recommended not to use the end parts of the strips and thereby the edges of the mould surface.

4-1-3 Torsion

Strips which are perpendicular in the surface interact not only by forces, but also by bending and torsional moments. Bending moments of strips connected to a perpendicular strip must be in equilibrium. There are moments in two directions from strip main axis. A third moment is torsional. The equation for the torsional moment is:

$$\phi_t = \frac{M_t \cdot x}{GI_t} + C_1 \tag{4-8}$$

This equation involves two extra unknown constants: The torsional moment M_1 and a C_1. To solve these extra unknown two extra boundary conditions must be found for each strip. Therefore can be used that the bending moment in one strips must be equal to the torsional

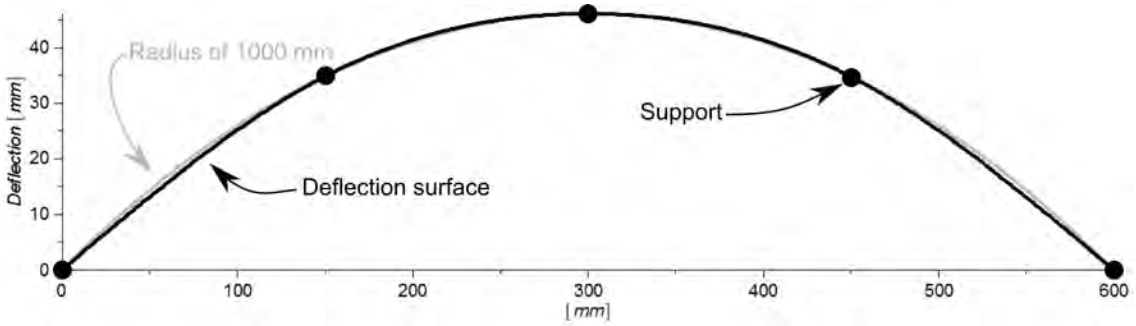


Figure 4-8: Intended and calculated shape of a strip on five supports. Conditions: R = 1000 mm, q = 0 kN/m, EI = 192550 Nmm^2.

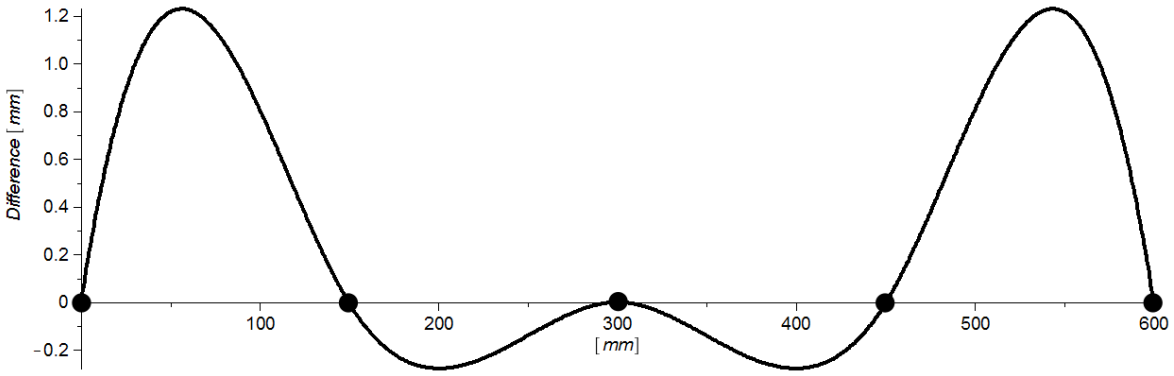


Figure 4-9: Difference between intended and calculated shape.

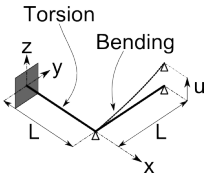


Figure 4-10: Interaction of bending and torsion between two connected strips. One strip is clamped at one end and the other end a deflection is imposed. Therefore one strips is subjected to bending and one to torsion.

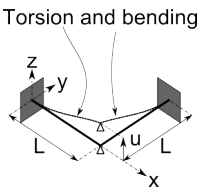


Figure 4-11: Interaction of bending and torsion between two connected strips. Both strips are clamped at one end and the shared end a deflection is imposed. Therefore both strips are subjected to bending and to torsion.

moment in the perpendicular strip. An other boundary condition can be found in the rotations of the strips with similar conditions. In total there are now six unknown constant for each part of the strips. The figures 4-10 and 4-11 illustrate a situation of interaction between bending moment and torsional moments.

When this method is applied to the strips of the flexible mould the number of unknowns grows rapidly because for each part of the strips between two supports a new equation is required. Therefore the number of unknowns becomes large. When the number of supports increase. An larger amount of boundary conditions is required to solve this system of equations. If m and n are the number of supports in two directions the number of unknowns can be calculated. Since only the strips on top of the support are considered there are equal amount of strips in the top layer as support m. For the setup of equations each strips is divided in pieces between the support which are $(n-1)$. So for the top later the number of unknown constants is equal to $6 \cdot m \cdot (n-1)$. A similar situation is present for the strips in the bottom layer which results in $5 \cdot n \cdot (m-1)$. Together these two provide insight in the quantitative complexity of solving structural calculation for the flexible mould. Table 4-1 shows the rapid increase of number of unknowns for m and n ranging up to 5. Luckily repetition can be found for the setup of the boundary conditions which can be programmed very well, for example in Matlab as in Appendix K.

Supports	2	3	4	5
2	24	42	60	78
3	42	72	102	132
4	60	102	144	186
5	78	132	186	240

Table 4-1: Number of unknowns

4-2 Kinematics

The method that will be described involves also kinematic aspects. These describe only displacements and rotations which cause deviation of the shape of the surface. Two main aspect are now described. The horizontal displacement of the supports and the rotation of the supports.

4-2-1 Horizontal displacement of the supports

When the strips in the flexible mould would deform there will occur horizontal displacements of the supports. This will cause a deviation of the intended shape. Figure 4-12 illustrates the horizontal displacements of a single strips. If the support would not displace in horizontal direction the strips should extend. However the strips will resist to this creating horizontal reaction forces. This is one of the reasons that in the new prototype of the flexible mould horizontal displacement are allowed due to the pendulum supports. In order to obtain an accurate shape of the mould surface the questions raise how large the horizontal displacement is and how this can be compensated for.

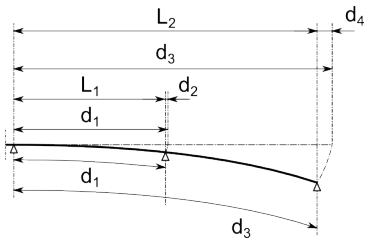


Figure 4-12: Horizontal displacements of supports after deformation.

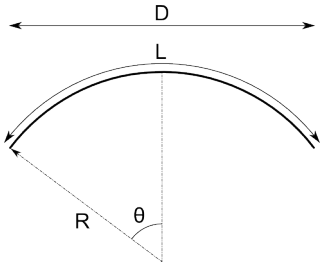


Figure 4-13: Length of an arc with constant radius

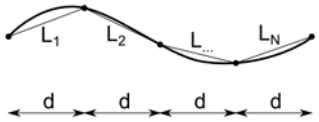


Figure 4-14: Approximation of curve length by linear distance between points.

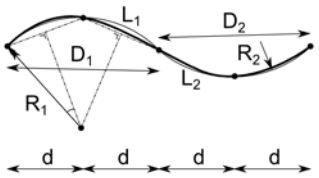


Figure 4-15: Approximation of the length of a curve by assuming continuous curvature between points.

It was assumed that the length of the strips after deformation would not change. To determine the horizontal displacement a method was required to obtain the length of a shape or curve. To gain insight in this first the arc length of a part of a circle was calculated. The circumference of an arc as in Figure 4-13 can be calculated with the formulas:

$$L = 2 \cdot R \cdot \theta \tag{4-9}$$

$$sin(\theta) = \frac{D}{2 \cdot R} \tag{4-10}$$

When the curvature of a curve is discontinuous it becomes more difficult to determine the length. Two methods, that use a geometric approach, are shown in Figure 4-14 and 4-15. Here the length of the curve is a summation of different parts. The first method uses the distances between points on the curve. The second method uses an approximation of the curvature based on three points on the curve. The lengths can be determined using the Equations 4-9 and 4-10. Both methods will approximate the length within a curtain accuracy. The accuracy can be enlarged by making the step size (d) smaller.

Another approach could be of more accurate. The arc length formula could be used to determine the length of a curve of which the function is known and if its derivative is continuous. The arc length formula is:

$$L = \int_a^b \sqrt{1 + \left(\frac{dy}{dx}\right)^2}\, dx \tag{4-11}$$

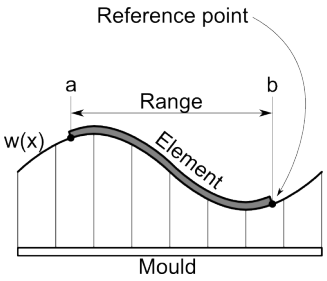


Figure 4-16: Range of element can be used to find reference points of the edge.

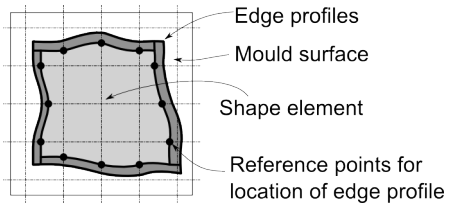


Figure 4-17: Reference points on the mould surface can be used to position the edge profiles.

An approximation of the horizontal displacement can be made using this formula and the function for deflection obtained by structural calculations. By taking the integral of this function the length of the curve between the original support distance in known. This will be larger than the horizontal support distance. The difference is the approximation of the horizontal displacement. This way the actual shape is used for an approximation of the horizontal displacement instead of making multiple approximations for the curvature.

4-2-2 Edge control

The arc-length formula in Equation 4-11 could also be used to find an approximation of the location of the edges. When the required geometry has virtually been placed in the mould the horizontal range of the element measured over the strips can be determined. Using these ranges in the arc-length formula result in a reference point of the edge. Figure 4-16 illustrates this. The reference points can be set out. An edge can be positioned based on these reference points. However there are limited points for positioning. A fluent edge profile could be a good approximation of the required edge. More reference points could be determined if the number of supports is increased. Figure 4-17 illustrates the placement of the edge profiles based on the reference points.

4-2-3 Compensation support heights

Due to horizontal displacements and rotations of the supports a deviation will occurs compared to the required geometry. This can be seen in Figure 4-18 and 4-19. The height could be compensated for these effects. This can be calculated using goniometric formulas. The input that is required for these calculations can be derived from the arc length formula and the equations of structural calculations. There is a difference in formulas for supports left and right of the middle support because displacement where described according to the global coordinate system. The horizontal displacement will always be towards the middle support. For the support left of the middle support this resulted in a positive displacement. For the support to the right of the middle supports this resulted in a negative displacement. Therefore the sign changes and this was implemented in the equations. At every support two strips cross. Therefore a compensation should be calculated for two direction. The final height to compensate is the sum of these two.

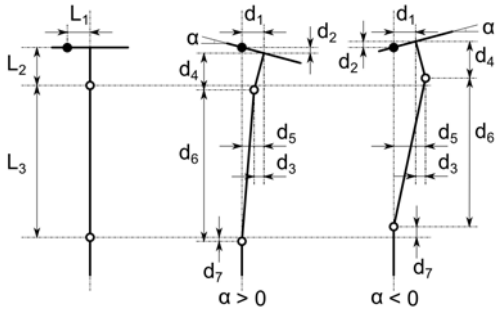


Figure 4-18: Used distances for compensation of support heights left of the middle supports.

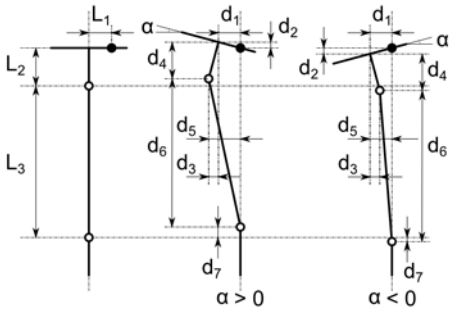


Figure 4-19: Used distances for compensation of support heights right of the middle supports.

Equations for supports left of the middle support:

$$\begin{aligned} d_1 &= L_1 \cdot cos(\alpha) \\ d_2 &= L_1 \cdot sin(\alpha) \\ d_3 &= L_2 \cdot sin(\alpha) \\ d_4 &= L_2 \cdot cos(\alpha) \\ d_5 &= d_1 - d_3 \\ d_6 &= \sqrt{(L_3)^2 - (d_5)^2} \\ d_7 &= L_2 + L_3 - d_2 - d_4 - d_6; \end{aligned}$$

Equations for supports right of the middle support:

$$\begin{aligned} d_1 &= -L_1 \cdot cos(\alpha) \\ d_2 &= L_1 \cdot sin(\alpha) \\ d_3 &= L_2 \cdot sin(\alpha) \\ d_4 &= L_2 \cdot cos(\alpha) \\ d_5 &= d_1 + d_3 \\ d_6 &= \sqrt{(L_3)^2 - (d_5)^2} \\ d_7 &= L_2 + L_3 - d_2 - d_4 - d_6; \end{aligned}$$

4-3 Proposed method for setup of flexible mould

The process to go from a geometry to the setup of the flexible mould involve the steps the where described in previous sections. But there are more steps to be taken that do not involve calculations. Together they for a method for the setup of the flexible mould which will be described in steps. In Appendix G the results of using this method can be recognized. The following steps are required to determine the setup of the mould;

Step 1 Determine the orientation of the geometry within the flexible mould: the support heights depend on the orientation of the geometry of the shape in the mould. The range at which the support can be set is limited. Thereby a bounding box is created for the shapes to fit within. Within this bounding box the shape can be placed in any position. However is would advantageous to limit the range of height differences of the supports because it would save time during setup of the mould, however this advantage will be small. A second and more important aspect is the influence on the deviation of the shape. The compensation of the support heights is based on an approximation of the horizontal displacement and the angle at the supports. When these can be taken as small as possible the made error when making the approximations is limited to a minimum. This will result in a more accurate shape. Therefore it is preferred that the orientation of the shape in the mould is such that the shape height difference and the angles at the supports are limited to a minimum.

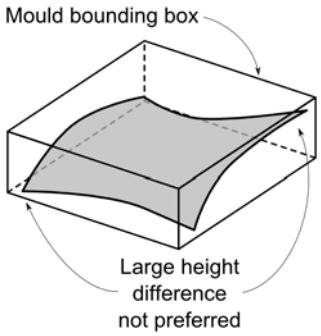


Figure 4-20: Orientation of the shape where large difference in support height is needed. This is not preferred.

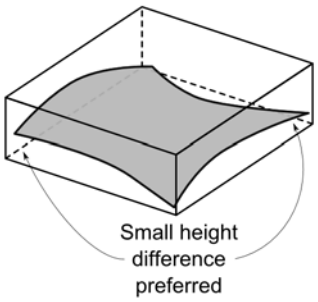


Figure 4-21: Orientation of the shape where small difference in support height is needed. This is preferred.

Step 2 Determine initial support height: for the boundary conditions of the structural calculation support heights are needed. These can be derived from the shape that was placed within the bounding box of the mould. This has been done using 3D modeling software called Rhinoceros and Grasshopper. A custom tool has been programmed to derive the support heights. Within this tool the intersection are made between the shape and the grid at which the supports are placed. The advantage of this software is the parametric setup. For example the support distance can be changed and the intersection heights change accordingly. In Figure 4-22 the tool is shown. The tool also provides information that can be used for structural calculations in the next step. Figure 4-23 shows the locations of the intersections with the geometry according to the parameters provided to the Grasshopper component.

Step 3 Calculate the shape and reaction forces of surface: structural calculation have been performed according to Section 4-1. First the strips in the top layer have been calculated. After that the strips in the bottom layer. Structural calculations involve repetition of the same calculations with varying parameters. Here the parametric setup of the Grasshopper tool reduces the calculation time. The most left box in Figure 4-22 can be exported as text file. Software packages like Matlab are able to read the data an perform the structural calculations. Structural calculation have been performed using the software packages Matlab and Maple. The advantage of Matlab is that it uses a similar programming language compared to the custom Grasshopper components. Therefore the calculations could be transferred to the Grasshopper component and all calculation can be done within the component. For future use this is preferred since it will be most practical.

Step 4 Calculate the compensation of the heights: the compensation of the support heights was based on the approximation of the horizontal displacement and angle at the supports. This was described in Section 4-2. Additionally the height was also compensated for the deviations caused by the buildup of the mould. Since it was built by hand some part may have been made a view millimeter longer or shorter as intended. A benchmark measurement was performed to determine this deviation. As result a list of support height is created which can be used to setup the supports of the flexible mould.

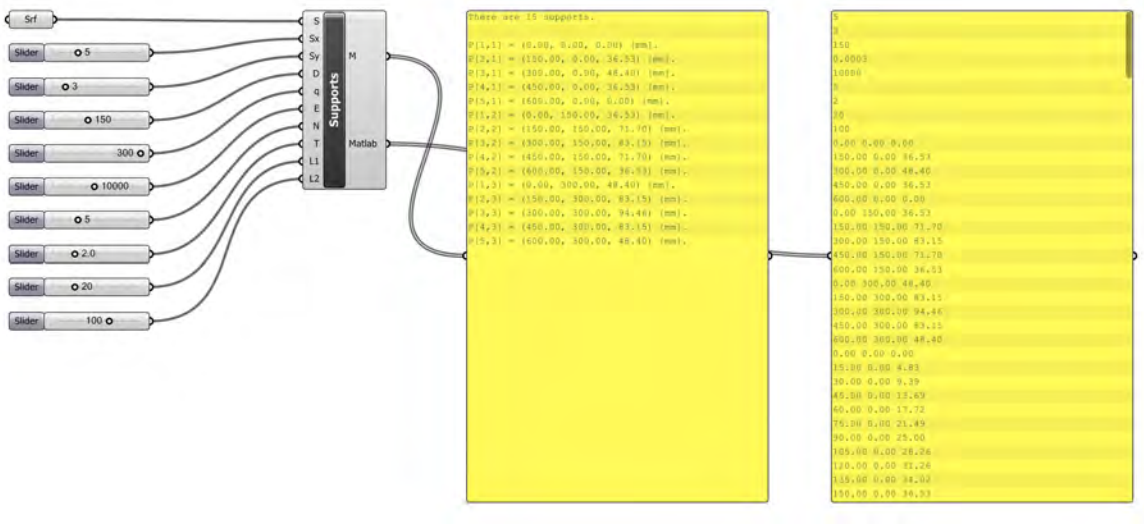


Figure 4-22: Custom Grasshopper component determines the initial support heights according to the distance between the support. The component also provides other information for structural calculation. This information can be read by other software. For example Matlab.

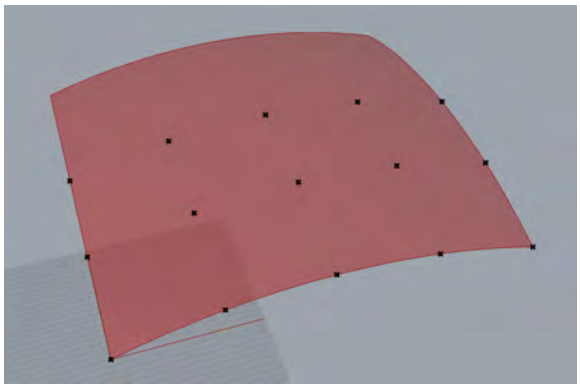


Figure 4-23: The initial support points determined by the grasshopper component.

4-4 Conclusions

A method has been developed which can be used to determine the required support heights given the geometry of a curtain shape. The method implements structural calculations for the strips of the mould and kinematic calculations for compensating of initial support heights for deviations caused by displacements and rotations of the mould surface.

For structural calculations single bending is considered although assumptions of the theory are not fulfilled. Would result be insufficient then double bending and torsion could be implemented for more accurate results.

If calculations are made by hand they take a lot of time. Currently calculations are made using Maple, but since there is a lot of repetition using varying parameters they could be programmed for Matlab or a Grasshopper parametric component. This would give the advantage of much shorter calculation time.

The proposed method for approximation of horizontal displacements could also be used to determine the location of the edges in horizontal position of the mould. This is not implemented within this research.

There is one disadvantage that can be observed. The shape of the fields between the outer ring of support points and the second-outer ring can only be influenced by the height of the support points. Since the bending moment on the end of each individual strip is zero, the curvature of the outer edge of the plane is also zero. This is contradictory to the wish that in a shape with constant curvature like for spherical shapes were the radius is constant over the full area. Therefore it could be decided not to use the outside of the mould surface.

Chapter 5

Testing of new prototype and accuracy of the mould surface

In order to check the functionality and performance tests have been performed with the new prototype. Therefore four different shapes have been tested. Deformations have been observed visually and by measurements which were performed using a 3D laser scanner. The results of these scans have been compared to the intended shape to gain insight in the accuracy of the mould.

5-1 Description of tests and measurements

For the test with the new prototype of the flexible mould four shapes have been chosen with varying Gaussian curvature. Namely; zero, positive and negative. Zero Gaussian curvature means that the shape is flat or single curved. The latter was applied in the first test. A second shape had positive curvature, which means that both principle curvatures point in the same direction. The third shape had negative Gaussian curvature. Therefore the principle curvatures point in perpendicular direction. The fourth and last shape had varying curvature because it was free of form. The shapes can be seen in the Figures 5-1 to 5-4. The radii of curvature for the first three shape was 1000 millimeter. These radii are quite small compared to earlier tests and were chosen to see if limitations of the mould would show. These shapes have been processed using the method which was described in the Chapter 4 which has resulted in heights of the supports. An overview of these results can be found Appendix E. This appendix also include all test results.

After setup of the support height and deforming of the mould the geometry has been obtained using a laser scanner which measures millions of points in three-dimensional space. The scanner that was used was a FARO Photon 120/20 as is shown in Figure 5-5. Using a 3D scanner has the advantage of being accurate (under certain conditions) and fast. The available alternative was measuring by hand as. These conditions were decisive for using this method. One scan was made of each tested shape, but the scan could be used for two types of

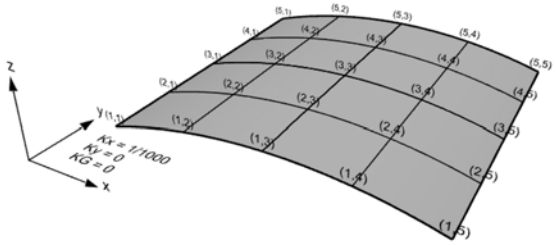


Figure 5-1: Shape 1 with zero Gaussian curvature. $R_{min} = 1000$ mm.

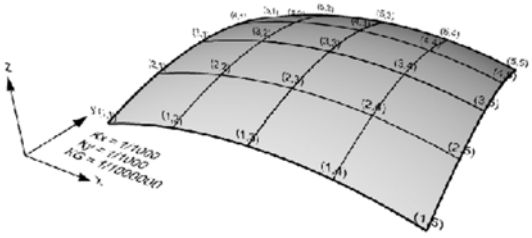


Figure 5-2: Shape 2 with positive Gaussian curvature. $R_{min} = R_{max} = 1000$ mm.

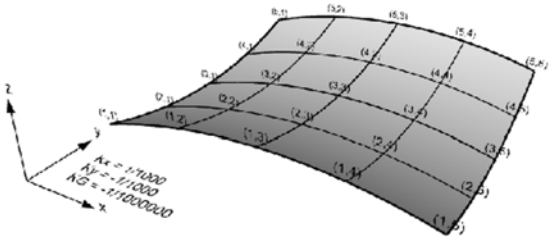


Figure 5-3: Shape 3 with negative Gaussian curvature. $R_{min} = R_{max} = 1000$ mm.

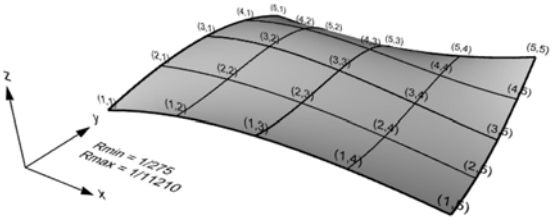


Figure 5-4: Shape 4 with positive and negative Gaussian curvature. $R_{min} = 275$ mm, $R_{max} = 11210$ mm.

Figure 5-5: FARO Laser Scanner Photon 120/20.

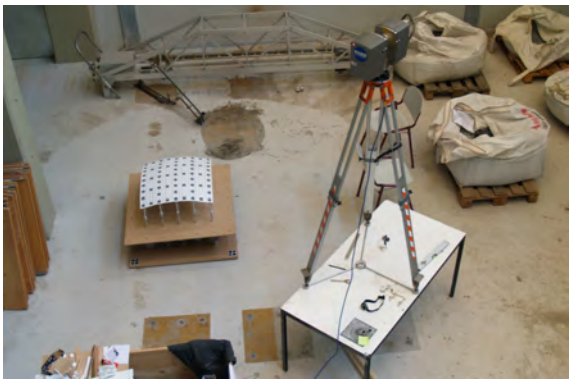


Figure 5-6: Scanning of the tested shape in the laboratory.

measurements. Namely, displacements of targets placed on the mould surface and the overall accuracy.

To measure a specific location on the mould targets were used. These targets were designed such that their location could be determined using the 3D scanner. The scanner can not only determine the location of points, but can also determine the gray scale of the measured object. This property was used to isolate the targets from the rest of the mould surface. The scanned data was imported in the software package Matlab and analyzed using a threshold value for the distinction between black and white points. White point were removed so that only black point would be used. The Figures 5-7 and 5-8 illustrated this method showing the scan of one of the tested shapes. For each target the middle point was determined by taking the average coordinate of all isolated points of that target. After the scans were processed it was found the accuracy of the target locations was limited. Earlier made trial scan showed more accurate results. A more detailed description of the accuracy of the scans can be found in Appendix E. It was found that the location of the targets was measured with an Root Mean Square Error of 1.14 millimeter in the x-direction, 1.33 millimeter in the y-direction and 1.54 millimeter in the z-direction. The result was not as good expected. The relative large error could be explained by bad reflection of the targets. The results of the measurements are described in the following section. However due to the relative large measuring error they are indicative for the horizontal displacements. For validation of the approximations of the displacement the measurements are insufficient. In some cases the measuring error is larger than the expected displacement. Therefore it would be required to perform more accurate tests. For that purpose a suggestion for a new target is described in Appendix E.

To determine the horizontal displacements of the mould surface during deformation the location of specific point were compared to a benchmark scan which was done. The difference between the points was taken as horizontal displacement. The middle support of the mould has been fixed so that it could be used as a reference point. Horizontal displacements have been determined only for the support near the edges of the surface. At these positions the largest displacements were expected. By comparing the displacements to the expected horizontal displacements insight could be gained in the accuracy of the approximation of displacements. These approximations have been described in Chapter 4.

The second type of measurement is the overall accuracy of the mould surface. Therefore the

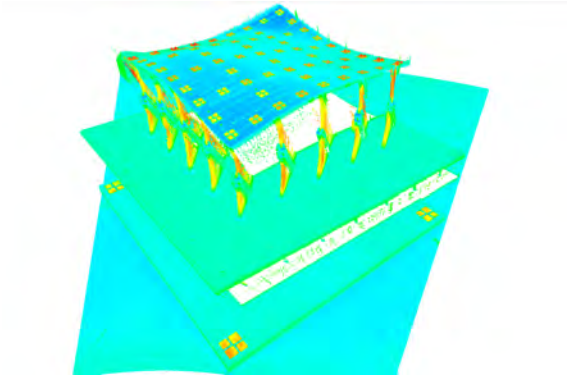


Figure 5-7: Full scan of one of the tested shapes.

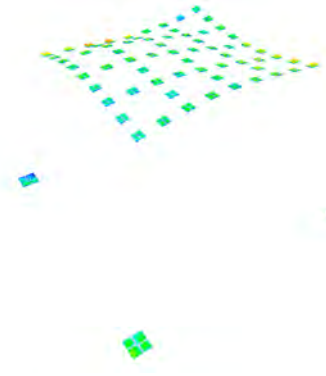


Figure 5-8: Isolated target point from the scan.

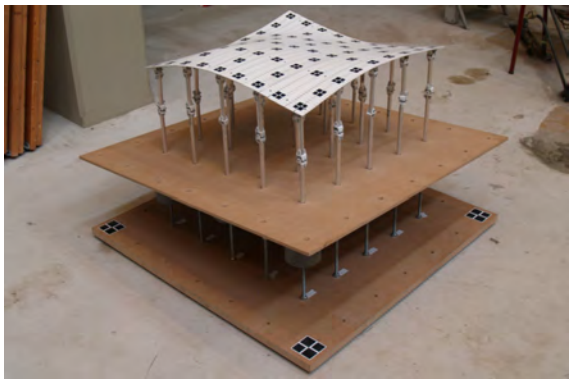


Figure 5-9: Shape of fourth and last test.

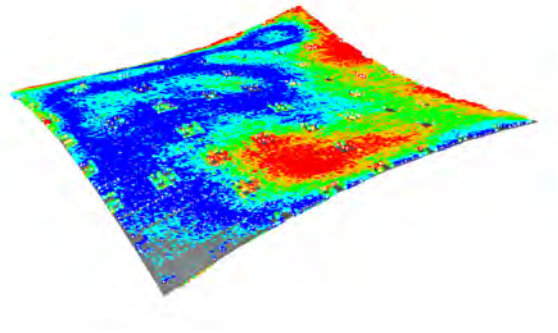


Figure 5-10: Point deviation of the scanned points compared the intended surface.

scanned data of the total surface was compared to the intended geometry of the surface. For this comparison the scanned points of the surface were isolated instead of the targets and a point cloud was the result. The isolated points were imported in the modeling software package Rhinoceros which included a point deviation tool. This tool determine the shortest distance of a point to a reference surface. The Figures 5-9 and 5-10 show a tested shape and the point deviation in Rhinoceros. As result it provides statistic information about the points. A better insight in the overall accuracy of the mould could be gained. As reference a scan was made of the horizontal leveled surface. This scan showed a mean distance was 0.73 millimeter, median distance 0.59 millimeter and the standard deviation 0.59 millimeter. Therefore the measurements of the total mould surface geometry is more accuracy than measurements of specific points on the surface.

5-2 Test results

Four test have been performed with the new prototype of the flexible mould. During these tests better insight was gained in the performance and accuracy of the mould surface. The measurements of the scans can be found in the Appendix G.

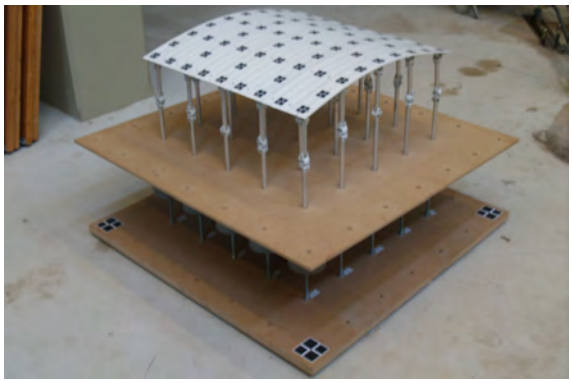


Figure 5-11: Shape of first test with the new flexible mould prototype.

Figure 5-12: The surface deformed into a smooth shape with continuous looking curvature in the middle.

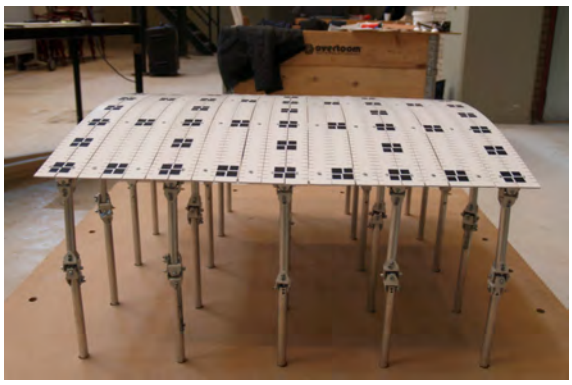


Figure 5-13: A smooth and flat surface perpendicular to the direction of the curvature.

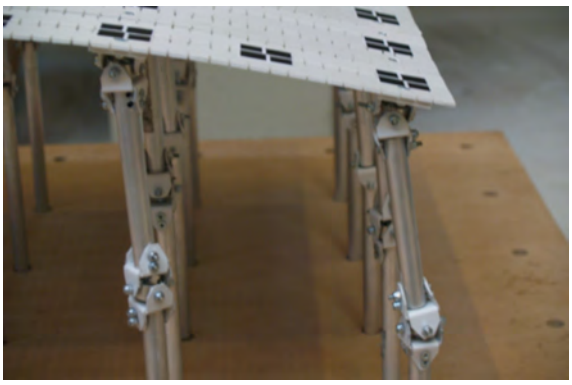


Figure 5-14: Horizontal displacement was observed at the support. No jamming of the support was found.

Test 1: zero Gaussian curvature

The first test is based on a surface which is single curved. Therefore strips are loading by bending only and supports should displace horizontally in the same direction as the curvature. The Figures 5-11 and 5-12 show the results of this first test. The mould appeared smooth with a continues curvature at the middle of the surface. Near the end of the strips the shape became more flat. This was expected because there is no bending moment in the strips at that point. In perpendicular direction the mould was flat. Compared to the previous version of the mould neighbor strips where consistent in height as can be seen in Figure 5-13. Horizontal displacements of the supports were observed as can be seen in Figure 5-14. The mould could handle the radius of curvature easily and would be able to manage smaller radii. During deformation no sign of jamming of the supports could be found.

Measurements of the horizontal displacement showed that direction of the expected displacements was correct. The Root mean Square Error of the displacements in x-direction was 1.27 millimeter. For the point deviation of the overall mould surface compared to the intended surface the mean distance was 0.91 millimeter, median distance 0.75 millimeter and the standard deviation 0.73 millimeter. Near the edge and only locally a maximum point deviation

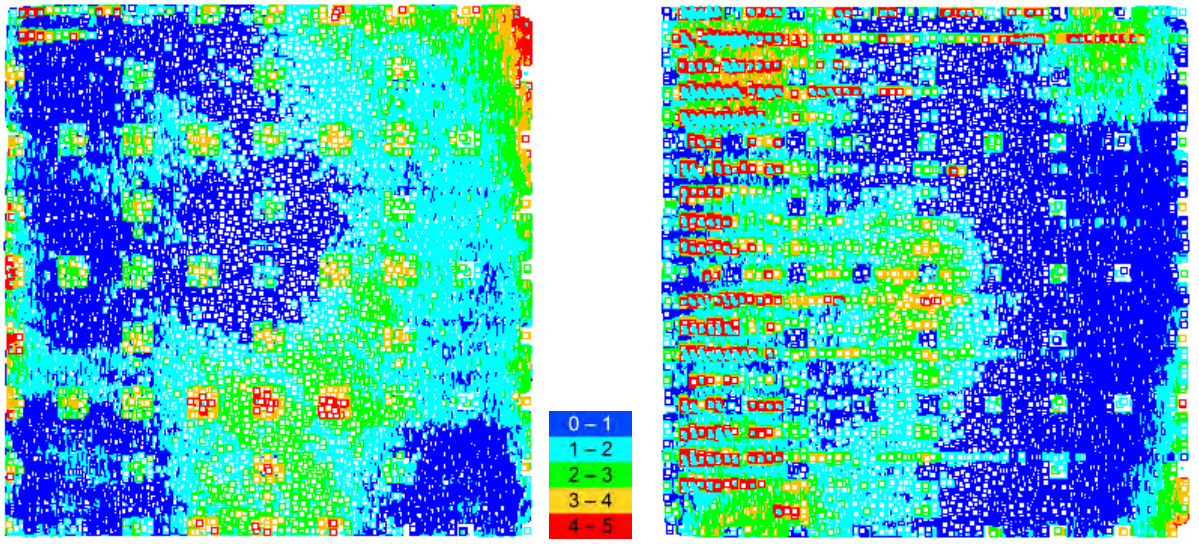


Figure 5-15: Top view of point deviation for the first test.

Figure 5-16: Bottom view of point deviation for the first test.

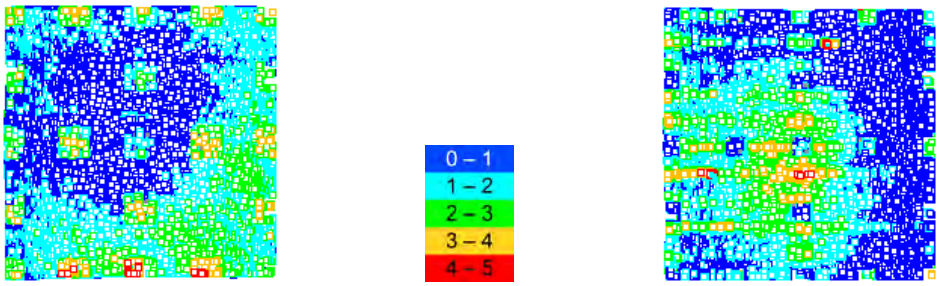


Figure 5-17: Top view of point deviation for middle surface of the first test.

Figure 5-18: Bottom view of point deviation for middle surface of the first test.

was found between 4 and 5 millimeter. For the middle part of the surface the point deviation reduced. The mean distance was 0.78 millimeter, median distance 0.67 millimeter and the standard deviation 0.58 millimeter. In the middle of the surface a maximum point deviation was found between three and four millimeter. Results are shown in the Figures 5-15 to 5-18.

Test 2: positive Gaussian curvature

The second test was performed using double curved shape with positive Gaussian curvature. The Figures 5-19 and 5-20 show the result after deformation of the mould. Similar to the first test, continuous curvature was observed in the middle of the surface. At the edges of the surface the shape became more flat. It was observed that the paper targets, that were glued on the surface, deformed. The targets appear to come loose from the surface which can be seen in Figure 5-21. This was expected as the surface would be subjected to shear and the paper target would not able to follow this deformation but the strips could. The shear deformation could also be observed at the edges of the surface. Initially the edges were

Figure 5-19: Shape of second test with the new flexible mould prototype.

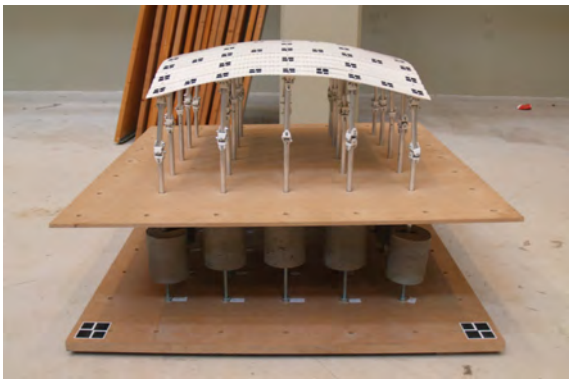


Figure 5-20: A smooth surface with continuous looking curvature in the middle of the surface.

straight. Figure 5-22 shows that the edge has deform in-plane. These two observations shows that the expected shear deformation occurred.

A second observation was done when weights were added to the supports. At the moment that not all weights were placed an effect appeared which showed resemblance to the observation by Bas Janssen for a solid plate. The edge deformed in a way that could be a sign of buckling. However negative supports reaction where expected and the additional weights were not placed yet. Therefore the shape could also be the result resistance to bending deformation. The deformation can be seen in Figure 5-23. Here the white line is the exaggerated shape of the edge. The effect was compensated after all the required weights were placed. Thereby the surface was forced into the set shape. The effect on the shape of the middle part of the surface is therefore small. This effect is expected to increase when a larger mould surface would be made. Forcing the surface into the required shape will be a solution to a curtain extent but has its limitations. Additional optimization of the surface stiffness would be required. This could be done by varying the depth and distance between the notches, which allow the surface to deform in plane.

Locally differences in height were found at the edges strips. This can be seen in Figure 5-24. This would reduce when a layer of foam is placed on the surface. This is used to cast the concrete. However a more precisely produced strips would also result in smaller differences, since the strips where now made by hand. Also the tightness of the nut and bolt connection between the strips is of influence.

The measurements of the horizontal displacement showed not as good results as the first test. The directions of the displacements was correct except for one point. For the displacements in x-directions the difference between the expected and the measured displacements was large. The Root mean Square Error of the displacements was 3.18 millimeter. This could be explained by bad reflection of the targets. More scans would be required to confirm this. For the displacement in y-direction the measurements showed better results. The Root mean Square Error of the displacements was 1.56 millimeter. For the point deviation of the overall mould surface compared to the intended surface the mean distance was 1.25 millimeter, median distance 1.04 millimeter and the standard deviation 0.98 millimeter. Near the edges a maximum point deviation was found between 4 and 5 millimeter. For the middle part of the

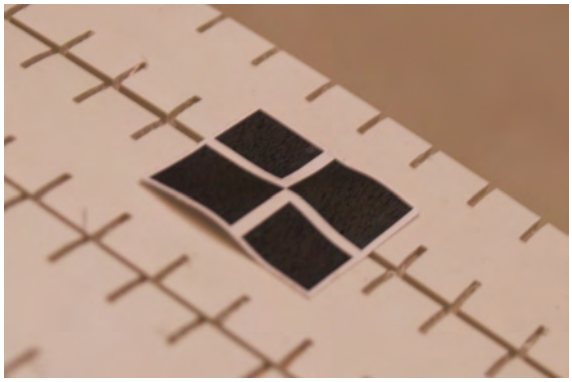


Figure 5-21: A paper target was not capable of following the deformation of the mould surface. The surface is subjected to shear deformation.

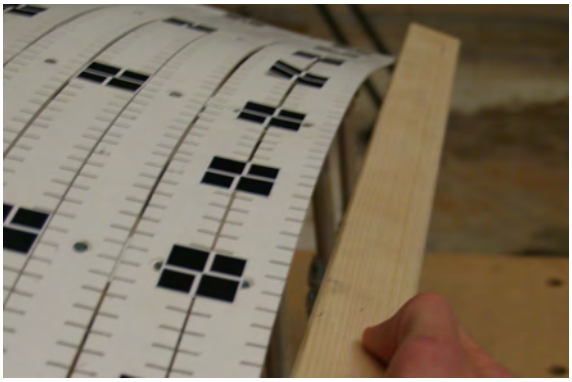


Figure 5-22: Shear deformation of the edges was also observed at the edges.

Figure 5-23: Shape of the edge of the surface when additional weights have not been placed yet. The white line is the exaggerated shape of the edge.

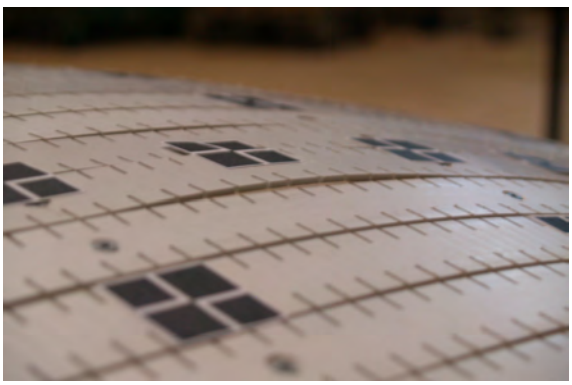


Figure 5-24: Locally the edges of the strips showed a difference in height.

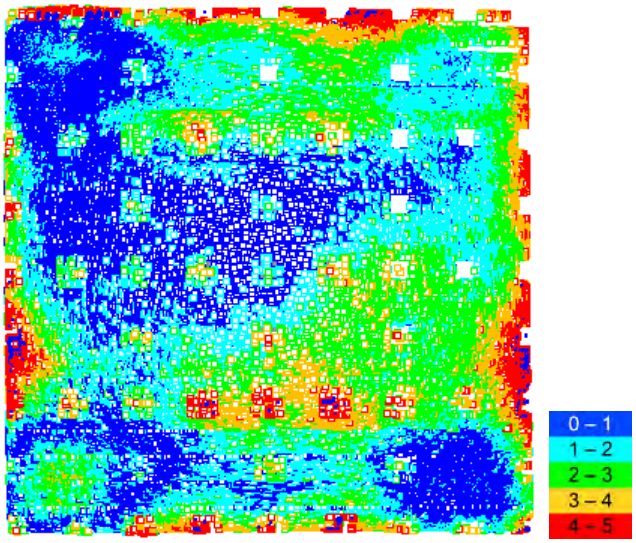


Figure 5-25: Top view of point deviation for the second test.

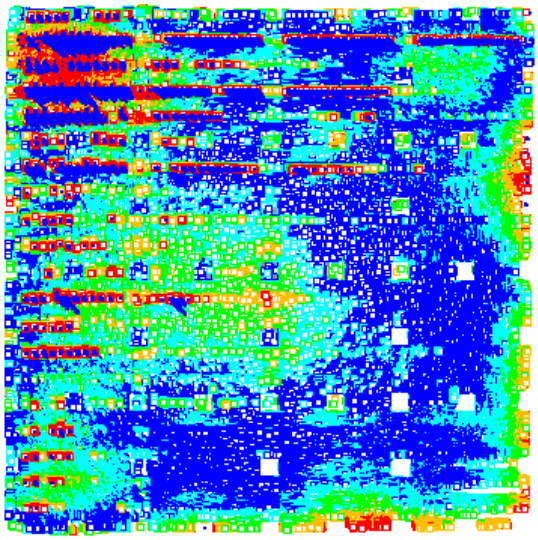


Figure 5-26: Bottom view of point deviation for the second test.

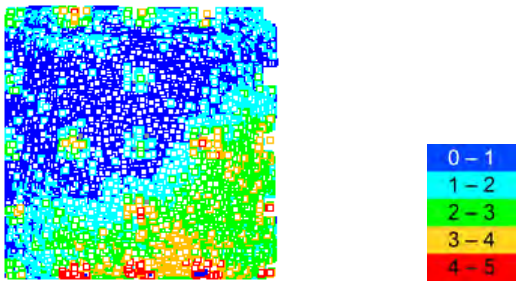


Figure 5-27: Top view of point deviation for middle surface of the second test.

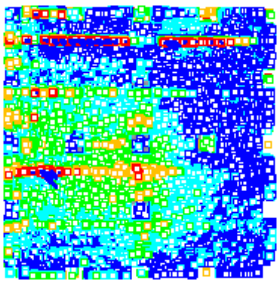


Figure 5-28: Bottom view of point deviation for middle surface of the second test.

surface the point deviation reduced. The mean distance was 0.96 millimeter, median distance 0.82 millimeter and the standard deviation 0.72 millimeter. Like in the first test a maximum point deviation between three and four millimeter was found for the middle surface. Results are shown in the Figures 5-25 to 5-28.

Test 3: negative Gaussian curvature

The third shape is also double curved. However principle curvatures are pointed in perpendicular directions. The results of the test can be seen in the Figures 5-29 and 5-30. The surface appeared smooth and horizontal displacements of the support were observed as expected. Compared to the previous shape it was notable that the surface also showed very smooth at the edges. Like in the first test, it seemed that the limitations of the mould where not yet reached by these radii of 1000 millimeter. The Figures 5-21 and 5-32 show this smoothness of the mould surface.

Like in the second test the measurement of the horizontal displacement showed better result

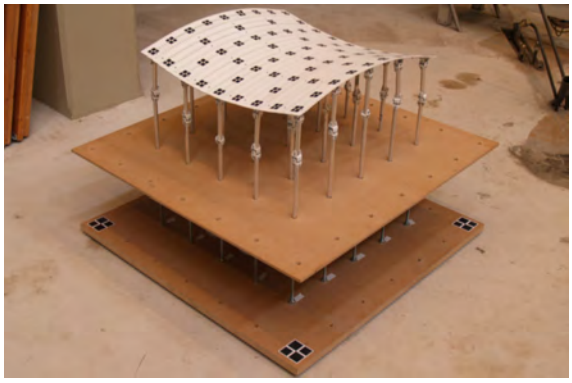


Figure 5-29: Shape of third test with the new flexible mould prototype.

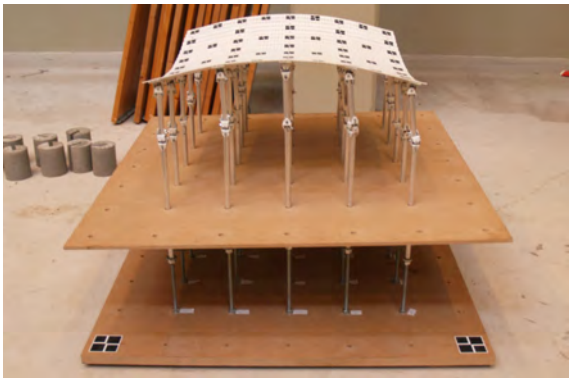


Figure 5-30: A smooth surface with continuous looking curvature in the middle of the surface.

Figure 5-31: A smooth surface after deformation.

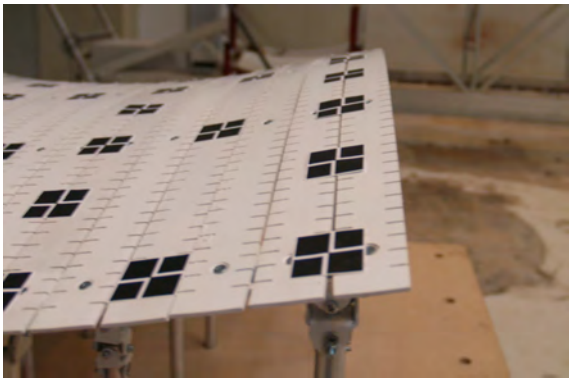


Figure 5-32: Edges showed no sign of local buckling.

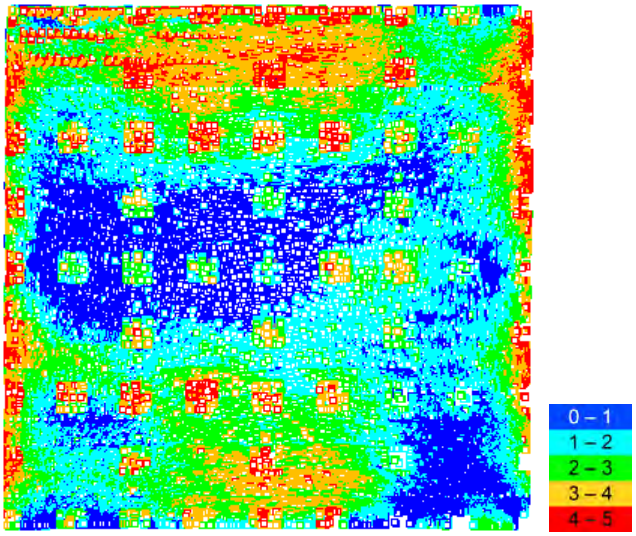


Figure 5-33: Top view of point deviation for the third test.

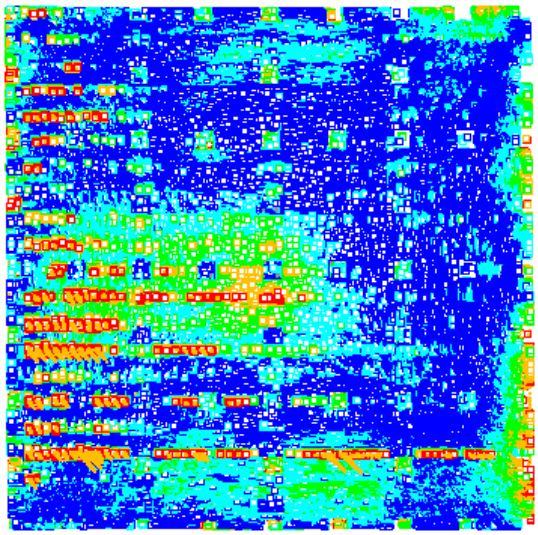


Figure 5-34: Bottom view of point deviation for the third test.

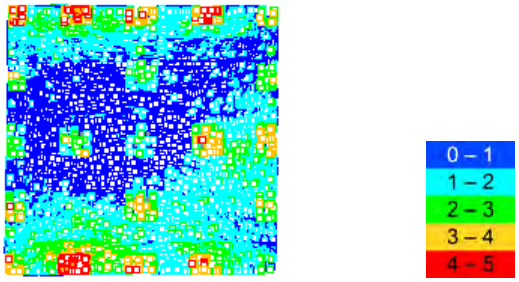


Figure 5-35: Top view of point deviation for middle surface of the third test.

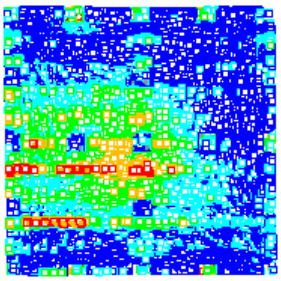


Figure 5-36: Bottom view of point deviation for middle surface of the third test.

for the displacements in the y-direction. However the directions of the displacements was correct for all points. For the displacements in x-directions the Root mean Square Error of the displacements was 2.01 millimeter. For the displacement in y-direction the Root mean Square Error of the displacements was 1.14 millimeter. For the point deviation of the overall mould surface compared to the intended surface the mean distance was 1.22 millimeter, median distance 1.05 millimeter and the standard deviation 0.90 millimeter. Near the edges a larger area showed a maximum point deviation was found between 3 and 5 millimeter. For the middle part of the surface the point deviation reduced. The mean distance was 0.86 millimeter, median distance 0.72 millimeter and the standard deviation 0.67 millimeter. A maximum point deviation between three and four millimeter was found for the middle surface. Results are shown in the Figures 5-33 to 5-36.

Test 4: varying Gaussian curvature

The fourth shape was free of form. The Figures 5-37 to 5-38 show the result of the last test which looked smooth. However locally the height differences between the strips, like in the

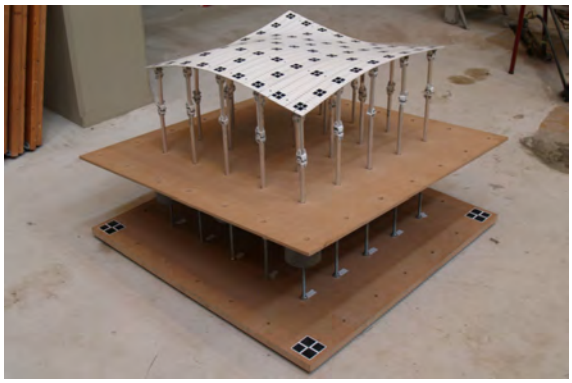


Figure 5-37: Shape of fourth and last test with the new flexible mould prototype.

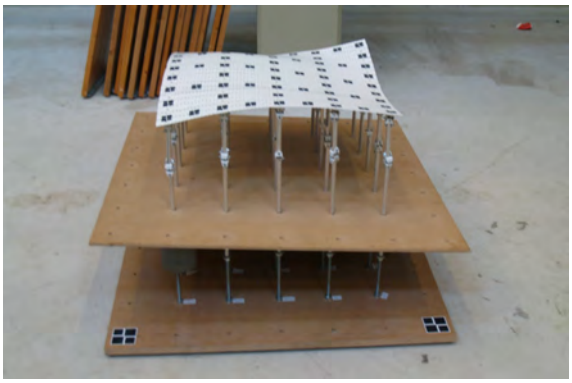


Figure 5-38: A smooth surface with locally differences in height between the strips.

Figure 5-39: A slight wave-like shape can be seen in the surface. This can be seen in the gaps between the strips.

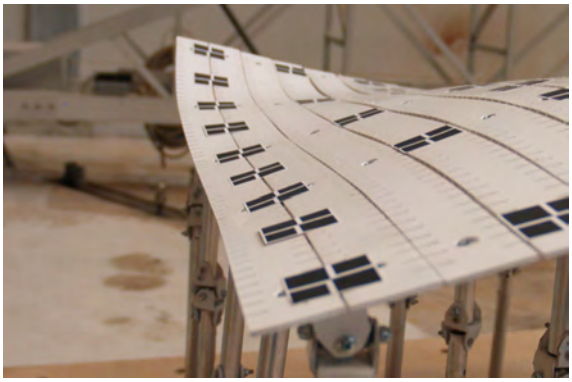


Figure 5-40: The edge strips clearly demonstrates deformation in two directions and torsion.

previous test was present more clearly. This could be caused by a locally small radius of curvature. Also a slight wave-like shape was found. In Figure 5-39 this can be seen at the gaps between the strips. The shape looked very similar to the intended shape and it can be expected the local height differences will reduce due to an additional layer of foam with silicon and the weight of the concrete. This shape was closest to the limitations of the mould, but still performed well. The edge of this shape demonstrates the deformations which the strips are subjected to. It clearly showed deformation in two directions and torsion as can be seen in Figure 5-40.

The measurement of the horizontal displacement showed larger differences for both x- and y-direction. However, again all the directions of the displacements were correct for all points. For the displacements in x-directions the Root mean Square Error of the displacements was 2.60 millimeter. For the displacement in y-direction the Root mean Square Error of the displacements was 1.55 millimeter. For the point deviation of the overall mould surface compared to the intended surface the mean distance was 1.63 millimeter, median distance 1.44 millimeter and the standard deviation 1.15 millimeter. Near the edges a large areas showed a maximum point deviation was found between 3 and 7 millimeter. For the middle part of

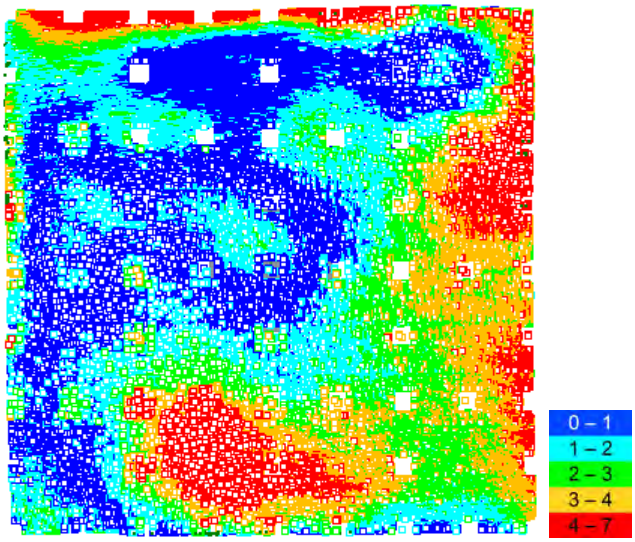


Figure 5-41: Top view of point deviation for the third test.

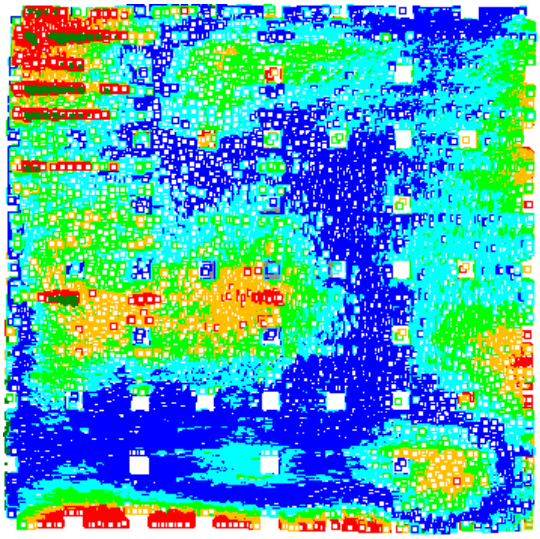


Figure 5-42: Bottom view of point deviation for the third test.

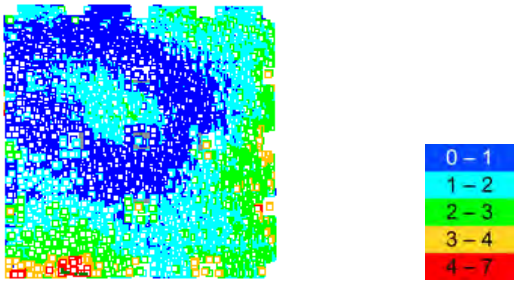


Figure 5-43: Top view of point deviation for middle surface of the third test.

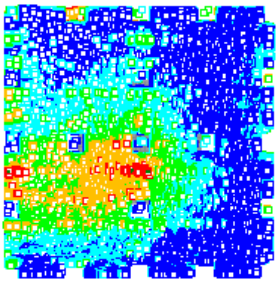


Figure 5-44: Bottom view of point deviation for middle surface of the third test.

the surface the point deviation reduced. The mean distance was 1.11 millimeter, median distance 0.98 millimeter and the standard deviation 0.78 millimeter. Again a maximum point deviation between three and four millimeter was found for the middle surface. Results are shown in the Figures 5-41 to 5-44.

5-3 Conclusions

To gain more insight in the performance and the accuracy of the new mould prototype four different shapes have been tested. The set shapes showed smooth surfaces and predicted displacements and rotations have been observed visually. This showed that the conceptual ideas introduced in previous chapters worked.

Two types of measurements have been performed. Displacements of supports and accuracy of the mould surface have been obtained. The accuracy of the measurements of displacements was insufficient to validate the approximations of the horizontal displacements. Bad reflection by the targets caused a relative large measuring error. However most measurements showed

displacements in directions as where expected. Additional more accurate measurements would be required are required.

The accuracy of the total mould surface was determined more accurately than the horizontal displacements. The mean deviation of the four tested shaped were respectively 0.91, 1.25, 1.22 and 1.63 millimeter. Locally three of the four test showed a maximum point deviation between 4 and 5 millimeter. Only the last test, which was free of form, showed a maximum point deviation between 4 and 7 millimeter.

If only the middle of the surface is considered the mean deviation reduces to respectively 0.78, 0.96, 0.86 and 1.11 millimeter. Locally all tests showed a maximum point deviation between 3 and 4 millimeter.

Chapter 6

Discussion

This research was started to gain more insight in behavior of the flexible mould during the deformation. It was expected that the mould would have some shortcomings that influence the accuracy of the mould since the mould was build to fulfill research requirement within a limited budget. The shortcomings were observed during a series of test with the, at that time, version of the flexible mould. The buildup of the mould did not provide sufficient control over horizontal displacement and rotations of the mould surface. Therefore the accuracy of the mould could not be ensured.

The behavior of the mould during deformation is complex due to its three-dimensional character. Insight in this has been gained after a study on the relation between the change of Gaussian curvature and strain of a surface. There are three components of strain that are important. Two components of extension and one of shear. The latter has a significant contribution to the deformation as can be seen in Figure 6-7. This study has led to a basic understanding of strain, displacements and rotations of the mould surface. The surface of the mould is an topic which is could be researched more. In existing concepts for a flexible moulds the surface is often not described in detail. This was a surprising finding since in this research the deformation and buildup of the surface and its support have an essential role.

During this research a balance was sought between theory and practice. A solely theoretical approach would have resulted in level of complexity which would hardly have added value to the practical side. A solely practical approach would not have lead to the gained results. The research has been alternating between studying theoretical aspects and trial and error of small test of the performance of new ideas. Eventually this has led to the introduction of a new prototype as in the Figures 6-1 and 6-2. The prototype, which was described in described in Chapter 3, could be used to study the behavior of the mould in more detail. The prototype implemented the following ideas, which can also be seen in the Figures 6-3 to 6-6:

- Plastic strips with notches. Strips have been bolted together, but not fully tightened. Self locking nuts have been used. Notches where made to reduce the resistance to deformation of the strips. Most importantly in their strong direction.

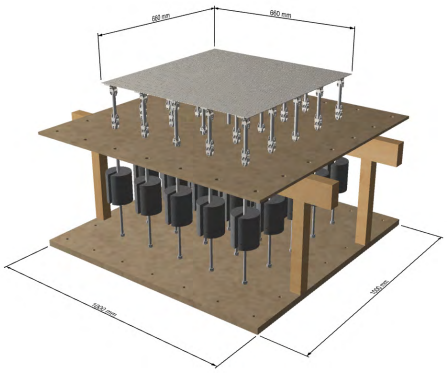


Figure 6-1: Design of a new prototype of the flexible mould.

Figure 6-2: Finished prototype of the flexible mould.

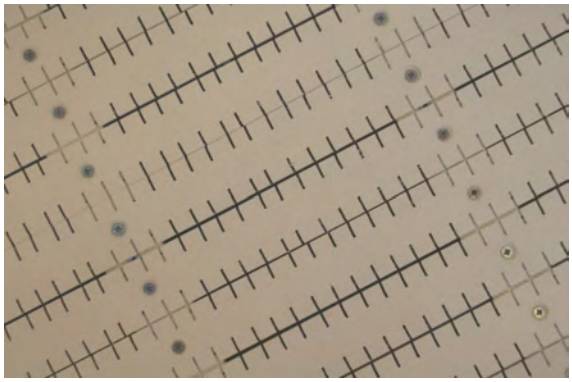


Figure 6-3: Plastic strips with notches. Notches reduce the resistance to deformation of the strips. Most importantly in their strong direction.

Figure 6-4: Universal joint connections allow rotation of the mould surface and are capable to transfer a force.

- Universal joints as connections that allow rotation while being capable of transferring a force
- Three-dimensional pendulum supports that allow horizontal displacement while being capable of transferring a force
- Weights as alternative way to apply additional loading to the surface

Deformation of the mould surface results in horizontal displacements and rotations of the mould surface. These cause a deviation of the intended shape. A method has been developed compensates the aims to compensate for these deviations by calculating and adjusting the support heights. Therefore the method approximates the horizontal displacements and rotations. These cause deviations from the intended shape. However these deviations can be compensated for by adjusting the support heights. The compensation can be calculated by kinematic calculations which use the result of structural calculations. Within this method structural calculations are performed based on equations for single bending beams. However the assumptions for this theory are not met. For example the change of angle of the strips is not small, but rather large. It was chosen to perform these calculations because their

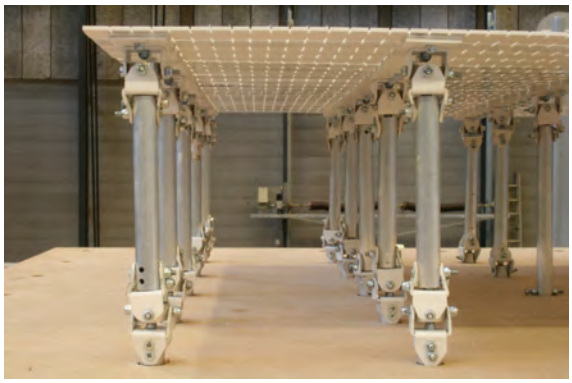


Figure 6-5: Three dimensional pendulum supports allow horizontal displacements and are capable to transfer a force.

Figure 6-6: Weight applied for additional loading such that no extra loading of the mould surface is required.

limited complexity. Double bending and theory of large displacement could be implemented to obtain more accurate results. However the complexity of calculations would be increased significantly. A justification of applying the single bending beam theory was sought in the deviation of the intended shape from the measured shape during test. If the deviation would be small, there would be less reason make the calculations more complex.

Tests with the new prototype mould showed that the mould performed as intended. All expected displacements and rotations were allowed by the connections and also appeared. Two types of measurements have been performed. The horizontal displacements where measured using targets and the overall shape of the tests was scanned. However the accuracy of the applied measuring method was not sufficient to confirm the predicted displacements. It should be noted that this does not imply that they were not correctly predicted. But the measurements did show that displacement in the were predicted in the correct direction. Also visually the displacements have been observed. Additional measurements would be needed to obtain more accurate results. For this purpose suggestions have been made to increase accuracy of measurements. The suggestions focused on the used targets. The second type of measurement was a scan of the total surface. The mean deviation of the four tested shaped were respectively 0.91, 1.25, 1.22 and 1.63 millimeter. These result show that the final shape of the surface come close to the intended shape. Locally three of the four test showed a maximum point deviation between 4 and 5 millimeter. Only the last test, which was free of form, showed a maximum point deviation between 4 and 7 millimeter. Therefore it would be possible to produce element with an accuracy of approximately 7 millimeter. Since only the support heights can be adjusted no moments can occur at the ends of the strips. Therefore the curvature becomes zero. Based on this it was expected that near the edges a larger deviation from the intended surface would occur. And if these parts would not be used the deviation should be reduced. The mean deviation of the surface without the edges for the tested shapes were respectively 0.78, 0.96, 0.86 and 1.11 millimeter. Which is less than the total surface. Locally all tests showed a maximum point deviation between 3 and 4 millimeter. Therefore, if the edge parts of the mould surface would not be used the elements would be produced with an accuracy of approximately 4 millimeter.

Something which could give rise to problems in the future is the scale of the mould. Strain of the surface will increase in case the mould would be bigger. Buckling could become a

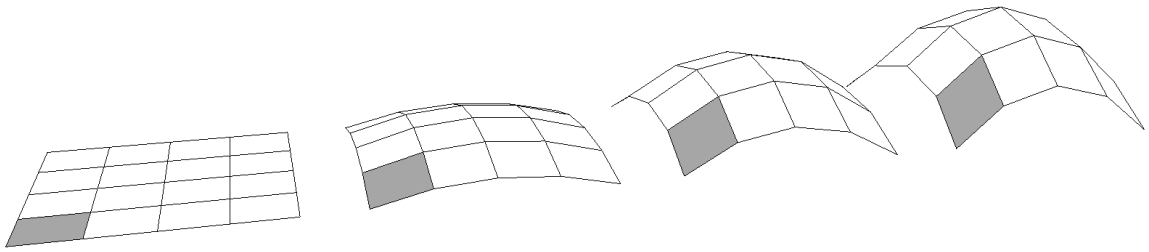


Figure 6-7: Shear deformation in gradually increasing Gaussian curvature of a surface.

problem again. However since a new mould surface has been designed to follow the in-plane deformation it is not expected that the effect will be the same compared to a solid plate. An aspect of the behavior of the mould during deformation that could be researched min more detail is the influence of the interaction between bending moments and torsional moment of the strips in perpendicular direction. It was assumed that the influence is small because strips where very flexible in torsional direction.

Considering the varies types of flexible moulds that have been developed by different people it is clear that there is no one superior way for production of double curved elements. This research aimed to contributed to the understanding of deformation of the mould surface and increasing the control over the geometry. This has resulted in the introduction of solutions that where implemented in a new prototype mould.

The described relation between change of Gaussian curvature and straining of a surface has not been obtained by me. I solely associated this relation with the topic of the flexible mould. This I have not found in any other description related to the flexible mould. From the relation I have deduced the consequences for the flexible mould and implemented this in the design for the new prototype of the flexible mould. It should be noted that the relation is relevant for all types of flexible moulds.

Not much is known yet about application of plastic as material for the strips. Plastics can have a large variety of material properties. For example varying non-linear elasticity, a non-homogeneous cross section or long term deformation. The latter would be important if it would occur already during the curing time of the concrete. The applied material could be optimized for the application in the flexible mould.

The goal of this research was to gain more insight in the behavior of the mould surface during the deformation and the research question was aimed at finding the aspect that influence the geometry of the mould surface. It was found that the following aspects have significant influence on the geometry of the mould surface are:

- Level of change of Gaussian curvature following directly from the required shape
- Stiffness of the mould surface. And especially the resistance to in plane shear deformation
- Loading of the surface
- Direction and size of displacements and rotations of the mould surface

- Type of material used for the strips
- Buildup and connection details of the mould

Chapter 7

Conclusions and recommendations

This research was set out to find aspect that influence the accuracy of the flexible mould for production of double curved concrete elements. Therefore observations have been done and studies have been performed on the deformation of the mould surface. A new prototype of a flexible mould has been designed and build. And finally the mould has been tested.

Based on the observations of the previous mould it was concluded that the accuracy of the mould could not be ensured. Displacements and rotations caused deviation from the intended shape and could not be controlled sufficiently.

From a study on the relation between change of Gaussian curvature and straining of a general surface it was concluded that shear deformation of the mould surface plays an essential role. The design of the mould surface should be such that this deformation can occur without too much resistance.

A design for a new prototype was made and the following solutions were introduced that aimed to improved control over the mould surface and increase the accuracy of the mould surface:

- Plastic strips with notches. Strips have been bolted together. Notches where made to reduce the resistance of the strips in their strong direction
- Universal joints as connections that allow rotation while being capable of transferring a force
- Three-dimensional pendulum supports that allow horizontal displacement while being capable of transferring a force
- Weights as alternative way to apply additional loading to the surface

For preparation of production of element a method has been developed that provides the support heights which need are needed to obtain a required shape. Therefore approximations

are made of horizontal displacements and rotations of the surface. Structural and kinematic calculation are needed to determine the support heights.

Structural calculation of the strips in the mould showed that the largest deviation from the intended shape can be found between the two last supports at the end of each strip. The curvature reduces because the moment in the strips reduces to zero at the last support.

A 3D scanner was used to obtain two types of measurements for four shapes that were tested with the new prototype of the flexible mould. Namely; displacement of the supports and the accuracy of the mould surface. The measurements of horizontal displacements at the supports after deformation were not sufficiently accurate to confirm the expected horizontal displacements. However the directions of the displacement were correct.

By measuring the geometry of the mould surface more insight has been gained in the accuracy of the mould. However no references was found to compare the results. The scans showed that the maximum deviation from the intended shape was 7 millimeter if the full surface was taken into account. When the area between the last two supports would not be used the maximum deviation reduced to 4 millimeter.

The accuracy of the measurements could be improved, but the overall measured geometry of the mould showed good resemblance to the intended geometry. No indication was found that the used theory for structural calculations of single bending beams would be insufficient. More accurate tests would provide better insight in this.

During the research process and development of the new mould aspects were found which could be developed further or optimized. A distinction can be made between recommendation for further research and development of the mould.

For further research:

- Accuracy of horizontal displacements and rotations of the strips
- Performance of the mould for larger surfaces
- Material of the strips: non-linear material properties, long term deformation, durability and maximum amount of load cycles
- Positioning of the mould edges

For further development:

- Ratio of stiffness of the strips versus loading by the concrete
- Reducing the amount of play in the universal joint connections
- More accurate setting of support height

Appendix A

Equations beam theory

For the purpose of structural calculations equations will here be derived for single bending, double bending and torsion according to the coordinate system and positive direction in Figure A-1.

A-1 Single bending

For single bending of a beam a fourth order differential equation can be used to derived equations for structural calculations. The differential equation describes a relation between a distributed load $q_z(x)$ and deflection $w_z(x)$ of beam with axis in x-direction according to Figure A-1. Adapted from Hartsuijker and Welleman [2011]. For calculations the following assumptions have been made before deriving the equations:

- Plane cross sections remain plane after deformation. This is also known the hypothesis of Bernoulli.
- Rotation of the cross section remains small
- The beam is considered to be slender. Extensional and shear deformation are small and therefore neglected
- Cross section is prismatic.
- Cross section is homogeneous and made of linear elastic material. Its behavior can be described by Hooke's law: $\sigma = E \cdot \varepsilon$

The governing fourth order differential equation for a single bending beam is:

$$q_z = E \cdot I_{zz} \cdot \frac{d^4 w_z}{dx^4} \tag{A-1}$$

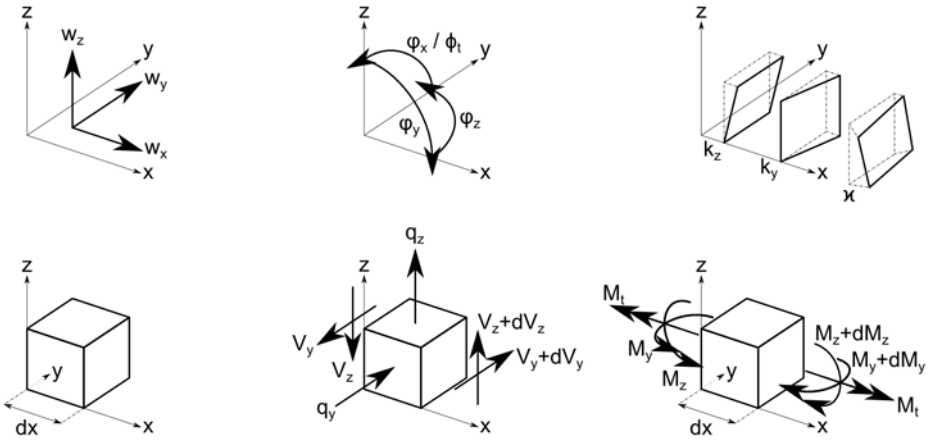


Figure A-1: Coordinate system and assumptions for positive directions.

This equation can be integrated four times. Doing so, an equation for the deflection is derived and four unknown constants are introduced. When only distributed loads are considered a equation is required for all parts of the strips between two supports. The equation for the deflection is:

$$w_z(x) \quad = \quad \frac{q_z(x) \cdot x^4}{24 \cdot EI} + C_1 \cdot x^3 + C_2 \cdot x^2 + C_3 \cdot x + C_4 \tag{A-2}$$

The rotation of the cross section can be described with a kinematic relation:

$$\phi_y = -\frac{dw_z}{dx} \tag{A-3}$$

The curvature of a cross section can also be described by a kinematic relation:

$$\kappa_z = \frac{d\varphi_y}{dx} \tag{A-4}$$

The moment can be described by a constitutive relation:

$$M_z = E \cdot I_{zz} \cdot \kappa_z \tag{A-5}$$

The shear force is described by:

$$V_z = \frac{dM_z}{dx} \tag{A-6}$$

The equilibrium equation for an equally distributed load and shear force is:

$$q_z = -\frac{dV_z}{dx} \tag{A-7}$$

Substitution of Equations A-2 to A-7 results in the governing differential equation:

$$q_z = E \cdot I_{zz} \cdot \frac{d^4 w_z}{dx^4}$$

A-2 Double bending

For double bending of a beam two fourth order differential equations can be used to derived equations for structural calculations in y- and z-direction. The differential equations describe relations between a distributed loads $q_z(x)$ and $q_y(x)$ and deflection $w_z(x)$ and $w_y(x)$ of a beam with axis in x-direction according to Figure A-1. Adapted from Hartsuijker and Welleman [2011]. For calculations the same assumptions have been made as described in Section A-1.

The governing fourth order differential equations for double bending are:

$$q_z = E \cdot I_{zz} \cdot \frac{d^4 w_z}{dx^4} + E \cdot I_{zy} \cdot \frac{d^4 w_y}{dx^4} \tag{A-8}$$

$$q_y = E \cdot I_{yz} \cdot \frac{d^4 w_z}{dx^4} + E \cdot I_{yy} \cdot \frac{d^4 w_y}{dx^4} \tag{A-9}$$

These equations can be integrated four times. Doing so, equations for the deflection in z- and y-direction are derived and eight unknown constants are introduced. When only distributed loads are considered equations are required for all parts of the strips between two supports. The equations for the deflection in z- and y-direction are:

$$w_z = \frac{(q_z \cdot I_{yy} - q_y \cdot I_{zy}) \cdot x^4}{24 \cdot E \cdot (I_{zz} \cdot I_{yy} - I_{yz} \cdot I_{zy})} + C5 \cdot x^3 + C6 \cdot x^2 + C7 \cdot x + C8 \tag{A-10}$$

$$w_y = \frac{(q_y \cdot I_{zz} - q_z \cdot I_{yz}) \cdot x^4}{24 \cdot E \cdot (I_{yy} \cdot I_{zz} - I_{zy} \cdot I_{yz})} + C1 \cdot x^3 + C2 \cdot x^2 + C3 \cdot x + C4 \tag{A-11}$$

The rotation of the cross section around the y- and z-axis can be described with two kinematic relations:

$$\varphi_y = -\frac{dw_z}{dx} \tag{A-12}$$

$$\varphi_z = \frac{dw_y}{dx} \tag{A-13}$$

The curvature of a cross section in y- and z-direction can also be described by two kinematic relations:

$$\kappa_y = -\frac{d\varphi_z}{dx} \tag{A-14}$$

$$\kappa_z = \frac{d\varphi_y}{dx} \tag{A-15}$$

The moment in z- and y-direction can be described by constitutive relations:

$$M_z = E \cdot I_{zz} \cdot \kappa_z + E \cdot I_{zy} \cdot \kappa_y \tag{A-16}$$

$$M_y = E \cdot I_{yz} \cdot \kappa_z + E \cdot I_{yy} \cdot \kappa_y \tag{A-17}$$

The shear forces in z- and y-direction are described by:

$$V_z = \frac{dM_z}{dx} \tag{A-18}$$

$$V_y = \frac{dM_y}{dx} \tag{A-19}$$

The equilibrium equations for equally distributed loads and shear forces in z- and y-direction are:

$$q_z = -\frac{dV_z}{dx} \tag{A-20}$$

$$q_y = -\frac{dV_y}{dx} \tag{A-21}$$

Substitution of Equations A-10 to A-21 results in the governing differential equations:

$$q_z = E \cdot I_{zz} \cdot \frac{d^4 w_z}{dx^4} + E \cdot I_{zy} \cdot \frac{d^4 w_y}{dx^4}$$

$$q_y = E \cdot I_{yz} \cdot \frac{d^4 w_z}{dx^4} + E \cdot I_{yy} \cdot \frac{d^4 w_y}{dx^4}$$

A-3 Torsion

For torsion of a beam a thirst order differential equation can be used to derived equations for structural calculations. The differential equation describes a relation between a torsional moment $M_t(x)$ and rotation $\phi_t(x)$ of beam with axis in x-direction according to Figure A-1. Adapted from Hartsuijker [2007]. For calculations the same assumptions have been made as described in Section A-1.

The governing differential equation for torsion of a beam is:

$$M_t = G \cdot I_t \cdot \frac{d\phi_x}{dx} \tag{A-22}$$

This equation can be integrated. Doing so, an equation for the rotation is derived and one unknown constant is introduced. An equation is required for all parts of the strips between two supports. The equation for the rotation is:

$$\phi_x = \frac{M_t \cdot x}{G \cdot I_t} + C_1 \tag{A-23}$$

The torsion of the cross section can be described with a kinematic relation:

$$\varkappa = \frac{d\phi_x}{dx} \tag{A-24}$$

The torsional moment can be described by a constitutive relation:

$$M_t = G \cdot I_t \cdot \varkappa \tag{A-25}$$

Substitution of Equations A-23 and A-25 results in the governing differential equation:

$$M_t = G \cdot I_t \cdot \frac{d\phi_x}{dx}$$

Appendix B

Building the new mould prototype

After the design for the new prototype of the flexible mould was finished it has been build. This appendix mainly shows visually the process of building the prototype and highlights a view aspects.

At first al the material where ordered and cut to size. The number of support that was chosen was 25 in a grid of five times five. However the dimensions of the base plate and the middle plate where chosen larger so that in the future the mould could be enlarged without having to order new plates.

Figure B-1 shows the base plate and threaded ends that were used for the supports. The threaded ends where cut approximately thirty centimeters so that three support could be taken from one bar of one meter. The mould was build in the Stevin two laboratory of the faculty of Civil Engineering and Geosciences. Figure B-2 shows and additional plate to keep the support in horizontal position. Also the mould surface, including the universal joint connectors, was already finished and waiting to be placed onto the mould.

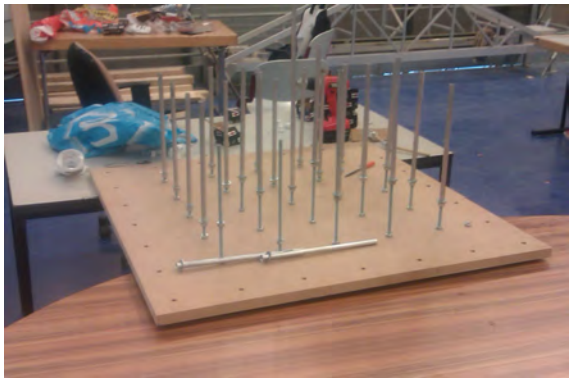


Figure B-1: Base plate with threaded ends which will be used as supports.

Figure B-2: An additional plate was placed over the supports. On the left the mould surface can be seen.

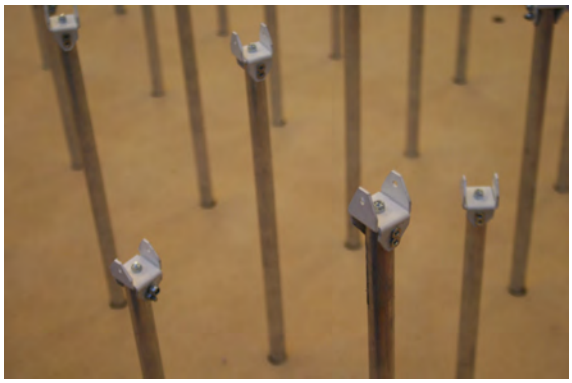


Figure B-3: Aluminum tube and plastic connectors for universal joints.

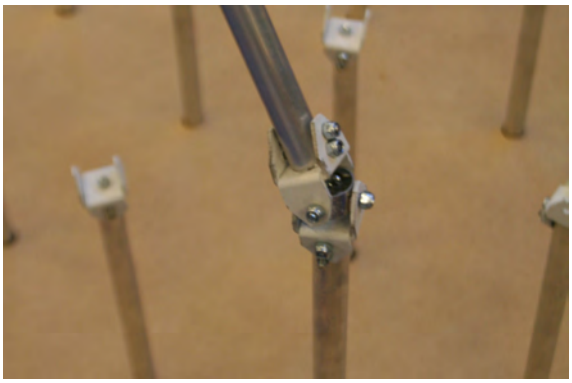


Figure B-4: First finished universal joint. The ring came from an old tent pole.

The universal joints where made with basic materials that can be bought in a local hardware store. The tube are aluminum and the white connectors have been cut from an u-shaped plastic profile. The result can be seen in Figure B-3. The rings in the universal joints where cut from an old tent pole. This pole happened to have the required diameter. The first finished universal joint can be seen in Figure B-4. Fixing all the parts of the universal joints toke the most time of assembling the mould.

The complete mould consists of many parts which. Al parts have been made by hand. Total assembly time of the mould was about fifteen days. Which was more than expected. It was found during building of them mould that some connections needed small adjustments. For example the direction from which the bolts where placed determined whether it could be reached with a screwdriver. In some cases this was not possible. Another example was the dimensions of the parts needed to be changed slightly because the universal joints could not make an large enough angle. These problems where typical during building. It was difficult to make all part the same dimension because everything was made by hand. It would be advantageous to order the parts and make them using machines. The parts would be more accurately produced. This would also be possible since the mould has been designed using as less different parts as possible. Therefore production costs would be limited and less different spare parts are needed.

Figure B-5 shows a stage of assembly where the lower joints of the pendulum supports are finished. After long and hard work finally the mould was finished as can be seen in Figure B-6.

After finishing the mould the weight needed to be made. This was done using a regular sewage pipe which was cut to the required length. An additional cut was made so that a piece of wood could be fitted in. This way there would be a slot in the shape of the weights. The combination was used as formwork for the concrete as can be seen in Figure B-7. An foil was needed to prevent the wood to become stuck in the hardened concrete. The tubes where fixed to a plate using silicon. This was to prevent the formwork from drifting up and concrete flowing out of the formwork. Also the tubes where oiled so that the formwork could be detached easily. This can be seen in Figure B-8.

After the formwork was ready a concrete mixture was made as can be seen in Figure B-9. After pouring the concrete it was found that an additional piece of tape was needed around

Figure B-5: Finished pendulum supports.

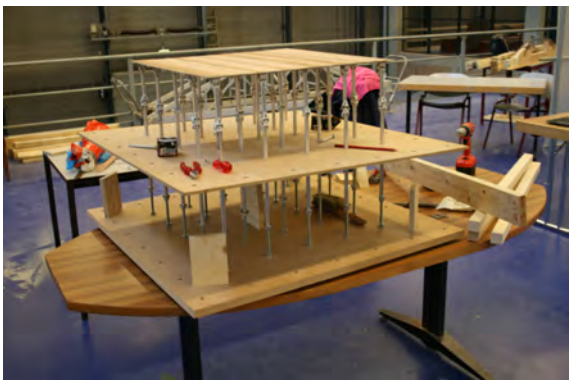


Figure B-6: Finished prototype of the flexible mould.

Figure B-7: Sewage pipes which were cut to required length where use as formwork for the concrete.

Figure B-8: A foil would prevent the wood from becoming stuck in the hardened concrete. Also the formwork was oiled to make so that the formwork would detach easily.

Figure B-9: Mixing of the concrete using an ordinary mixer.

Figure B-10: Filled formworks. tape was placed around the formwork to prevent expanding.

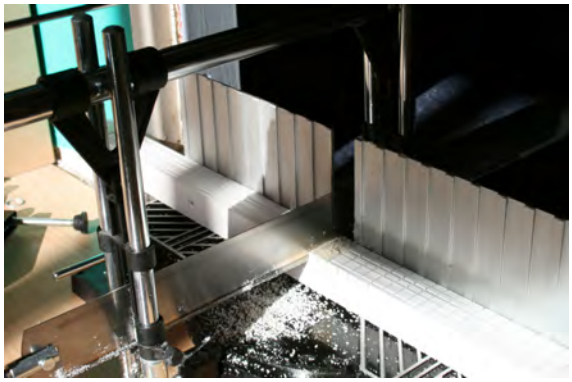


Figure B-11: A miter saw was used to make notches in the strips.

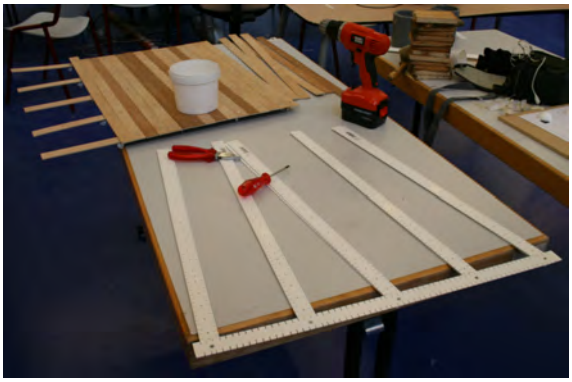


Figure B-12: Wooden strips in the mould surface have been replaced with plastic strips with notches.

the formwork. The formwork expanded due to the pressure of the concrete.

After the flexible mould was finished it was found that the mould surface needed to be redesigned. The surface was build using wooden strips which were too stiff although the dimensions where reduced. When a new surface was designed new strips needed to be made. Notches where made in the strips by using a miter saw as in Figure B-11. When the strips where finished the wooden surface could be replaced. Figure B-12 shows how the strips where replaced. Notches in the strips make them more flexible. In Figure B-13 the distance and depth of the notches can be seen. The nuts and bolts in the wooden surface where reused. The final result is shown in Figure B-14. At this point the mould was ready for testing.

Designing and building was a good learning and developing experience. It was found that the details are very important for a good performance of the mould.

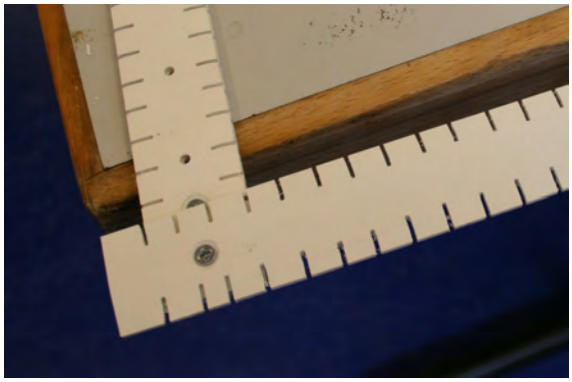


Figure B-13: Notches in the strips.

Figure B-14: Final result. The mould was ready for testing.

Appendix C

Strain test for material of strips

Plastic was used as material for the strips of the mould surface. Plastic are available many different variations. For this research it was chosen because of the flexibility and available size of the strips. Also the strips where very straight which was a advantage compared to wooden strips.

C-1 Material properties of plastics

Plastic is an artificial material which consists of polymers. Polymers are made from molecules which together form chains. Together the chains form a network which determine the material properties. Plastics can have varying material properties which could be advantageous and disadvantageous for application of the flexible mould. In this research only the Young's modulus was determined by test. However to obtain good insight more research is required on the type of plastic to use.

The material properties of plastics can be influenced during production. Thereby a wide range of material properties can be created. This is illustrated in Figure C-1. Here is can be seen that plastics may not have a Young's modulus as clear as for example steel. In these cases the Young's modulus at the start of the curve as used to characterize the material. In hardened state this values often varies between 3000 and 4000 N/mm^2 van der Vegt [1994]. For application of plastics the following aspects are of importance:

- Orientation of the polymers
- Admixtures during production
- Temperature
- Creep

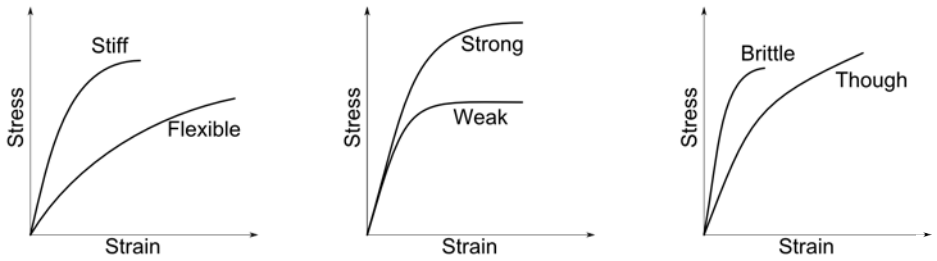


Figure C-1: Stress and strain relations for the range of different types of plastic. Adapted from van der Vegt [1994].

Figure C-2: Setup of the strain tests.

Figure C-3: Weight where added in steps. The extension was measured after each weight.

C-2 Strain tests

A strain test has been performed for the material of the strips in the mould surface. Because the material was bought in a local hardware store no properties where available. However a value for the Young's modulus has been derived from the tests. Thereby a value was obtain that could be used in structural calculation.

The strain tests were performed in a very basic setup. A strip was clamped at the top and weights where added in steps. The extension was measured after each weight. The Figures C-2 and C-3 show the setup of the tests.

The extension of the strip was measured accurately on two sides of the strips. Clamps held the sensors on place and determined the distance over which the extension was measured. The sensor reached half way the distance. Therefore an additional bar was placed on the lower part so the distance could be measured. The Figures C-4 and C-5 show the sensors and the clamps.

C-3 Test results

The results of the test have been processed to obtain σ-ε-diagrams. Therefore it was needed to calculated the stress and the strain. By dividing the load by the area of the cross section

Figure C-4: Two sensors where clamped on two sides of the strip.

Figure C-5: An bar was placed at the lower part. The sensor and bar meet in the middle.

the stress was obtained. The load was calculated by mass times the gravitational constant. From the dimensions of the strips, 30 times 2 millimeters, the area of the cross section was known. To obtain the strain the extension was divided by the distance between the clamps. These two calculations are written in the Equation C-1 and C-2.

$$\sigma = \frac{m \cdot g}{A} \tag{C-1}$$

$$\varepsilon = \frac{\Delta l}{l} \tag{C-2}$$

The Young's modulus can be determined by calculating the ratio of σ and ε of the data;

$$E = \frac{\sigma}{\varepsilon} \tag{C-3}$$

The results of the test are shown in Figure C-6. In each plot processed data can be seen for two measurements. The line in between is the average of the data. To determine the ratio of σ and ε a first order polynomial function was fitted thorough the average data. The directional coefficient is taken as this ratio. The results are respectively: 4321 N/mm^2 (RMSE: 0.04563), 3633 N/mm^2 (RMSE: 0.02406) and 3599 N/mm^2 (RMSE: 0.06922). The average of these three measurements is 3851 N/mm^2 which is in the range as could be expected.

From the diagrams can be seen that measurements on both sides of the strips do not strain equally. Possible explanations of this are; at first the applied test setup was not accurately enough or the way the weight where added would have influenced the results. More tests would be needed to gain more insight in this. A second possibility would be that the material is not homogeneous. Which would not be strange since the plastic consists of chains of polymer and its network depend on the production method.

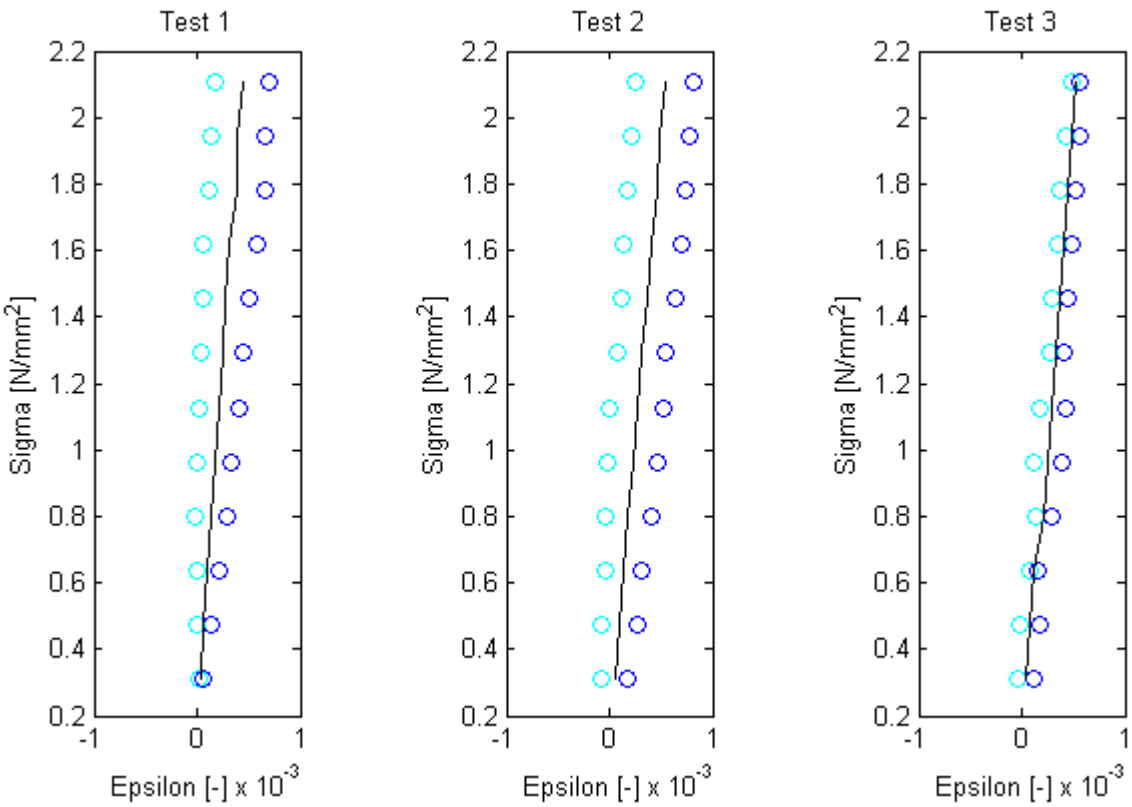


Figure C-6: Stress-strain diagrams for three tests.

C-4 Matlab script

```
%% 2013 Peter Eigenraam
% This script processes the data obtained from three strain tests of
% plastic strips. The goal is to obtain an Youngs modulus.

% Clear
clc;clear;close all;
format long

% Get results of first test.
filename1 = 'strain_test_1.txt';
test1 = textscan(fopen(filename1), '%f %f %f');

% Get results of second test.
filename2 = 'strain_test_2.txt';
test2 = textscan(fopen(filename2), '%f %f %f');

% Get results of third test.
filename3 = 'strain_test_3.txt';
test3 = textscan(fopen(filename3), '%f %f %f');

% Length of test strips.
l1 = test1{1,1}(1);
l2 = test2{1,1}(1);
l3 = test3{1,1}(1);

% Area of crosssection of strips.
a1 = test1{1,1}(2);
a2 = test2{1,1}(2);
a3 = test3{1,1}(2);

% Number of points to calculate.
nd1 = length(test1{1,1}) - 3;
nd2 = length(test2{1,1}) - 3;
nd3 = length(test3{1,1}) - 3;

% Calculate values for sigma-epsilon diagram
R1 = zeros(nd1, 4);
R2 = zeros(nd2, 4);
R3 = zeros(nd3, 4);

% For first test.
kg_temp = test1{1,1}(3);
e_1_temp = 0;
e_2_temp = 0;
e_ave_temp = 0;
for i = 1 : nd1
    % Sigma.
    kg_temp = kg_temp + test1{1,1}(i + 3);
    R1(i, 1) = kg_temp * 9.81 / a1;

```

```
51      % Epsilon 1.
52      e_1_temp = e_1_temp + ((test1{1,2}(i + 3) - test1{1,2}(i + 2)) / l1);
53      R1(i, 2) = e_1_temp;
54
55      % Epsilon 2.
56      e_2_temp = e_2_temp + ((test1{1,3}(i + 3) - test1{1,3}(i + 2)) / l1);
57      R1(i, 3) = e_2_temp;
58
59      % Epsilon average.
60      e_ave_temp = (e_1_temp + e_2_temp) / 2;
61      R1(i, 4) = e_ave_temp;
62  end
63
64  % Fit a first order polynome.
65  cftool(R1(:,4),R1(:,1));
66  fit(R1(:,4),R1(:,1),'poly1');
67
68  % For second test.
69  kg_temp = test2{1,1}(3);
70  e_1_temp = 0;
71  e_2_temp = 0;
72  e_ave_temp = 0;
73  for i = 1 : nd2
74      % Sigma.
75      kg_temp = kg_temp + test2{1,1}(i + 3);
76      R2(i, 1) = kg_temp * 9.81 / a2;
77
78      % Epsilon 1.
79      e_1_temp = e_1_temp + ((test2{1,2}(i + 3) - test2{1,2}(i + 2)) / l2);
80      R2(i, 2) = e_1_temp;
81
82      % Epsilon 2
83      e_2_temp = e_2_temp + ((test2{1,3}(i + 3) - test2{1,3}(i + 2)) / l2);
84      R2(i, 3) = e_2_temp;
85
86      % Epsilon average.
87      e_ave_temp = (e_1_temp + e_2_temp) / 2;
88      R2(i, 4) = e_ave_temp;
89  end
90
91  % Fit a first order polynome.
92  cftool(R2(:,4),R2(:,1));
93
94  % For third test.
95  kg_temp = test3{1,1}(3);
96  e_1_temp = 0;
97  e_2_temp = 0;
98  e_ave_temp = 0;
99  for i = 1 : nd3
100     % Sigma.
101     kg_temp = kg_temp + test3{1,1}(i + 3);
102     R3(i, 1) = kg_temp * 9.81 / a3;
103
```

```
104     % Epsilon 1.
105     e_1_temp = e_1_temp + ((test3{1,2}(i + 3) - test3{1,2}(i + 2)) / l3);
106     R3(i, 2) = e_1_temp;
107
108     % Epsilon 2
109     e_2_temp = e_2_temp + ((test3{1,3}(i + 3) - test3{1,3}(i + 2)) / l3);
110     R3(i, 3) = e_2_temp;
111
112     % Epsilon average.
113     e_ave_temp = (e_1_temp + e_2_temp) / 2;
114     R3(i, 4) = e_ave_temp;
115 end
116
117 % Fit a first order polynome.
118 cftool(R3(:,4),R3(:,1));
119
120 % Plot results of first test.
121 subplot(1,3,1)
122 scatter(R1((1 : nd1),2), R1((1 : nd1),1), 'cyan');
123 hold on
124 scatter(R1((1 : nd1),3), R1((1 : nd1),1), 'blue');
125 plot(R1((1 : nd1),4), R1((1 : nd1),1), 'black');
126 %plot(f1, 'green');
127 hold off
128 title('Test 1')
129 xlabel('Epsilon [-]');
130 ylabel('Sigma [N/mm^2]');
131
132 % Plot results of second test.
133 subplot(1,3,2)
134 scatter(R2((1 : nd2),2), R2((1 : nd2),1), 'cyan');
135 hold on
136 scatter(R2((1 : nd2),3), R2((1 : nd2),1), 'blue');
137 plot(R2((1 : nd2),4), R2((1 : nd2),1), 'black');
138 hold off
139 title('Test 2')
140 xlabel('Epsilon [-]');
141 ylabel('Sigma [N/mm^2]');
142
143 % Plot results of third test.
144 subplot(1,3,3)
145 scatter(R3((1 : nd3),2), R3((1 : nd3),1), 'cyan');
146 hold on
147 scatter(R3((1 : nd3),3), R3((1 : nd3),1), 'blue');
148 plot(R3((1 : nd3),4), R3((1 : nd3),1), 'black');
149 hold off
150 title('Test 3')
151 xlabel('Epsilon [-]');
152 ylabel('Sigma [N/mm^2]');
```

Appendix D

Maple sheet for structural and kinematic calculations

```
> restart;
```

This calculationsheet performs structutal and kinematic calculations required for the setup of a flexible mould. The calculations a bending beam (or strips) is considered on five supports. The beams (strips) are loaded by a combination of triangular distributed loads on each beam (strip). After solving the equations the sheet perform kinematic calculation for the setup of support heights of the flexible mould.

Define parameters for distance between support (d), height of each support (h), lengths of parts in connection of supports (l), value of load above supports (qrep) (remark: load is linear between supports) and bending stiffness (EI):

```
> d := d :
  h1 := h1 :
  h2 := h2 :
  h3 := h3 :
  h4 := h4 :
  h5 := h5 :
  l1 := l1 :
  l2 := l2 :
  l3 := l3 :
  qrep1 := qrep1 :
  qrep2 := qrep2 :
  qrep3 := qrep3 :
  qrep4 := qrep4 :
  qrep5 := qrep5 :
  EI := EI :
```

Define load functions:

```
> q1 := qrep1·(1 - x/d) + qrep2·(x/d) :
  q2 := qrep2·(2 - x/d) + qrep3·(x/d - 1) :
  q3 := qrep3·(3 - x/d) + qrep4·(x/d - 2) :
  q4 := qrep4·(4 - x/d) + qrep5·(x/d - 3) :
```

Define equations for each part of the beam (strip) between the supports. Start with the basic 4th order differential equation for a bending beam. Integrate this four time to derive the basic function for the deflection of the beam. There are four unknown constants for each beam (strip).

For the first part:

```
> EIw41 := q1 :
  EIw31 := int(q1, x) :
  EIw21 := int(EIw31, x) :
  EIw11 := int(EIw21, x) :
  w1 := int(EIw11, x)/EI + C1·x^3 + C2·x^2 + C3·x + C4;
  phi1 := -diff(w1, x) :
  k1 := diff(phi1, x) :
  m1 := EI·diff(phi1, x) :
  v1 := diff(m1, x) :
```

$$w1 := \frac{qrep1\left(\frac{1}{24}x^4 - \frac{1}{120}\frac{x^5}{d}\right) + \frac{1}{120}\frac{qrep2\,x^5}{d}}{EI} + C1\,x^3 + C2\,x^2 + C3\,x + C4 \qquad \textbf{(1)}$$

For the second part:

```
> EIw42 := q2 :
  EIw32 := int(q2, x) :
  EIw22 := int(EIw32, x) :
  EIw12 := int(EIw22, x) :
  w2 := int(EIw12, x)/EI + C5·x^3 + C6·x^2 + C7·x + C8;
  phi2 := -diff(w2, x) :
  k2 := diff(phi2, x) :
  m2 := EI·diff(phi2, x) :
  v2 := diff(m2, x) :
```

$$w2 := \frac{qrep2\left(\frac{1}{12}x^4 - \frac{1}{120}\frac{x^5}{d}\right) + qrep3\left(\frac{1}{120}\frac{x^5}{d} - \frac{1}{24}x^4\right)}{EI} + C5\,x^3 + C6\,x^2 + C7\,x + C8 \qquad (2)$$

For the third part:

```
> EIw43 := q3 :
  EIw33 := int(q3, x) :
  EIw23 := int(EIw33, x) :
  EIw13 := int(EIw23, x) :
  w3 := int(EIw13, x)/EI + C9·x^3 + C10·x^2 + C11·x + C12;
  phi3 := -diff(w3, x) :
  k3 := diff(phi3, x) :
  m3 := EI·diff(phi3, x) :
  v3 := diff(m3, x) :
```

$$w3 := \frac{qrep3\left(\frac{1}{8}x^4 - \frac{1}{120}\frac{x^5}{d}\right) + qrep4\left(\frac{1}{120}\frac{x^5}{d} - \frac{1}{12}x^4\right)}{EI} + C9\,x^3 + C10\,x^2 + C11\,x + C12 \qquad (3)$$

For the fouth part:

```
> EIw44 := q4 :
  EIw34 := int(q4, x) :
  EIw24 := int(EIw34, x) :
  EIw14 := int(EIw24, x) :
  w4 := int(EIw14, x)/EI + C13·x^3 + C14·x^2 + C15·x + C16;
  phi4 := -diff(w4, x) :
  k4 := diff(phi4, x) :
  m4 := EI·diff(phi4, x) :
  v4 := diff(m4, x) :
```

$$w4 := \frac{qrep4\left(\frac{1}{6}x^4 - \frac{1}{120}\frac{x^5}{d}\right) + qrep5\left(\frac{1}{120}\frac{x^5}{d} - \frac{1}{8}x^4\right)}{EI} + C13\,x^3 + C14\,x^2 + C15\,x + C16 \qquad (4)$$

Define the boundary conditions at the support. There are 16 unknown constants. Therefore 16 boundary condition are required.

Define boundary conditions at first support:

```
> x := 0 :
  eq1 := w1 = h1 :
  eq2 := m1 = 0 :
```

Define boundary conditions at second support:

```
> x := d :
  eq3 := w1 = h2 :
  eq4 := w1 = w2 :
  eq5 := phi1 = phi2 :
  eq6 := m1 = m2 :
```

Define boundary condition at third support:

```
> x := 2·d :
  eq7 := w2 = h3 :
  eq8 := w2 = w3 :
  eq9 := phi2 = phi3 :
  eq10 := m2 = m3 :
```

Define boundary conditions at fouth support:

```
> x := 3·d :
  eq11 := w3 = h4 :
  eq12 := w3 = w4 :
  eq13 := phi3 = phi4 :
  eq14 := m3 = m4 :
```

Define boundary conditions at fifth support:

```
> x := 4·d :
  eq15 := w4 = h5 :
  eq16 := m4 = 0 :
```

Solve the constants:

```
> solution := solve({eq1, eq2, eq3, eq4, eq5, eq6, eq7, eq8, eq9, eq10, eq11, eq12, eq13, eq14,
      eq15, eq16}, {C1, C2, C3, C4, C5, C6, C7, C8, C9, C10, C11, C12, C13, C14, C15, C16}) :
      assign(solution) :
```

Set variable x back to variable:

```
> x :='x':
```

The equation are now solved. The values for supports heights, loads and bending stiffness can be assigned.

Set values of dimensional parameters:

```
> d := 150 :
  h1 := 0 :
  h2 := 34.75 :
  h3 := 46.06 :
  h4 := 34.75 :
  h5 := 0 :
  l1 := 18.3 :
  l2 := 127.7 :
  l3 := 145.7 :
```

Set values for loads and bending stiffness:

```
> qrep1 := -0.01 :
  qrep2 := -0.01 :
```

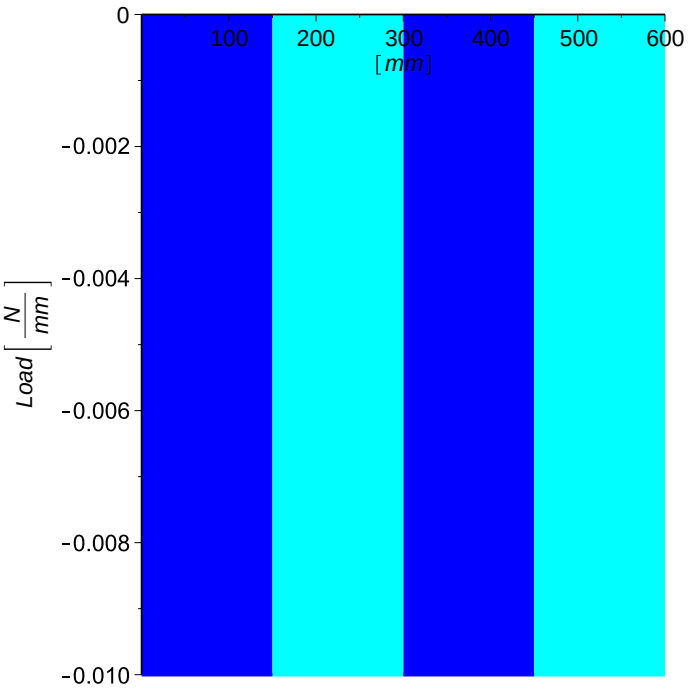
```
qrep3 := -0.01 :
qrep4 := -0.01 :
qrep5 := -0.01 :
EI := 3851·50 :
```

Possible check of constants:

```
> C1 : C2 : C3 : C4 : C5 : C6 : C7 : C8 : C9 : C10 : C11 : C12 : C13 : C14 : C15 : C16 :
```

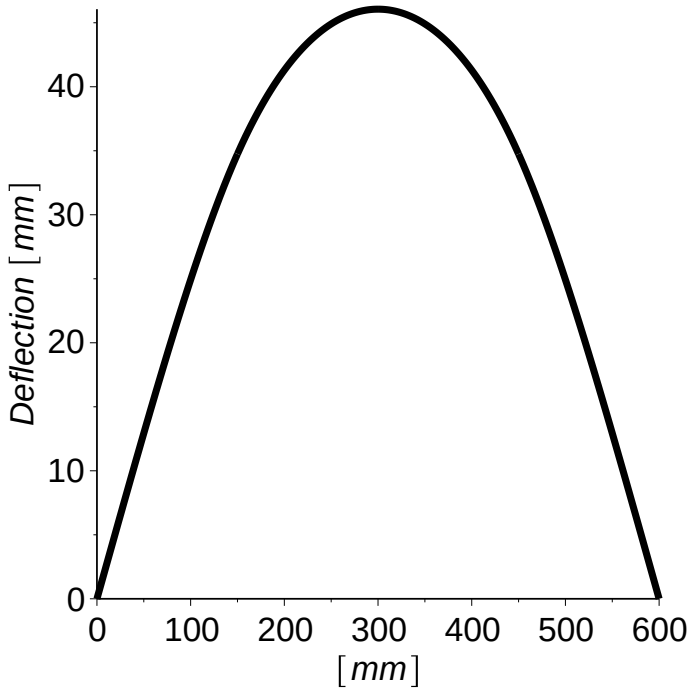
Plot functions for loading:

```
> with(plots) :
Q1 := plot(q1, x = 0 ..d, color = blue, thickness = 2 , filled = [color = "blue", transparency
      = 0.7]) :
Q2 := plot(q2, x = d ..2· d, color = cyan, thickness = 2, filled = [color = "cyan", transparency
      = 0.7]) :
Q3 := plot(q3, x = 2· d ..3· d, color = blue, thickness = 2, filled = [color = "blue", transparency
      = 0.7]) :
Q4 := plot(q4, x = 3· d ..4· d, color = cyan, thickness = 2, filled = [color = "cyan", transparency
      = 0.7]) :
display({Q1, Q2, Q3, Q4}, labeldirections = ["horizontal", "vertical"], axesfont = ["Arial", 10],
    legendstyle = [font = ["Arial", 10], location = right], labels = [[mm], Load[N/mm]], labelfont
    = ["Arial", 10]);
```



Plot functions of deflection:

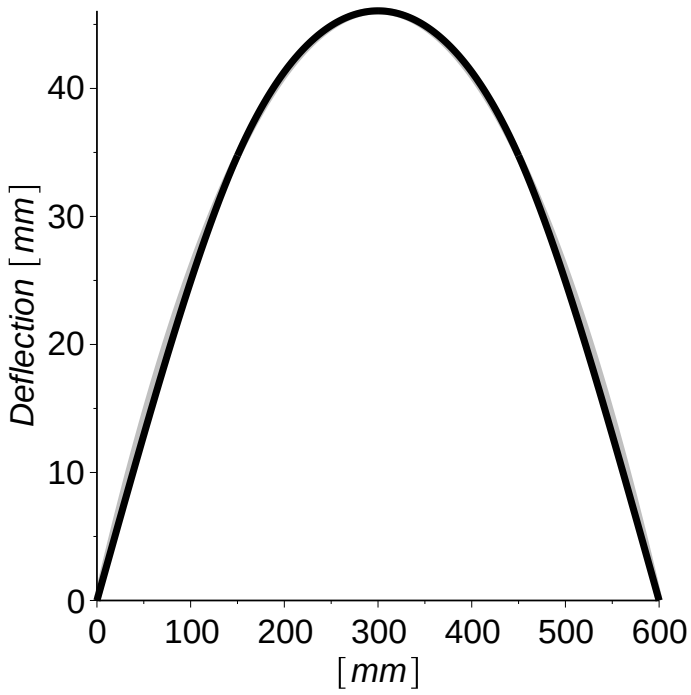
```
> with(plots):
  W1 := plot(w1, x = 0..d, color = black, thickness = 4):
  W2 := plot(w2, x = d..2·d, color = black, thickness = 4):
  W3 := plot(w3, x = 2·d..3·d, color = black, thickness = 4):
  W4 := plot(w4, x = 3·d..4·d, color = black, thickness = 4):
  display({W1, W2, W3, W4}, labeldirections = ["horizontal", "vertical"], axesfont = ["Arial", 15],
      legendstyle = [font = ["Arial", 15], location = right], labels = [[mm], Deflection [mm]],
      labelfont = ["Arial", 15]);
```

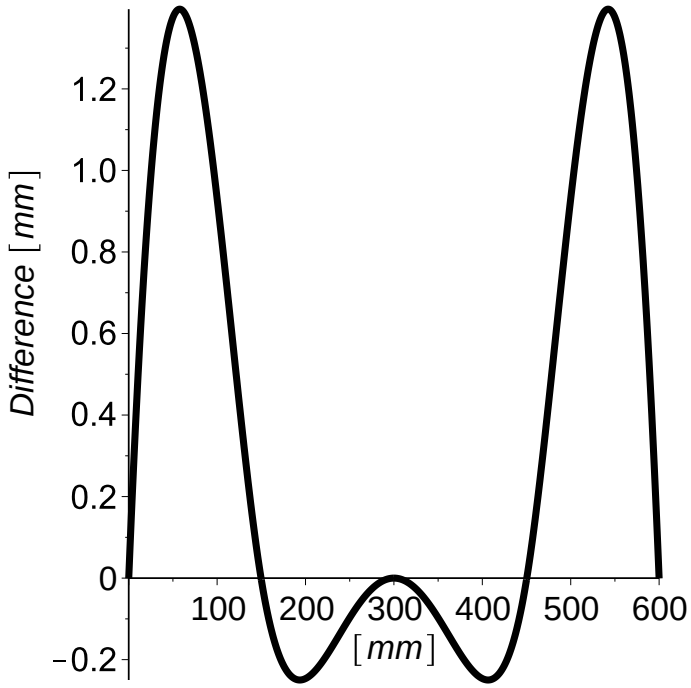


Plot difference between deflection and continious curvature (optional):

```
> x := 'x':
  r := 1000 :
  sh := sqrt(r^2 - (x - 2·d)^2) - r + h3 :
  SH := plot(sh, x = 0..4·d, color = gray, thickness = 4) :
  display({W1, W2, W3, W4, SH}, labeldirections = ["horizontal", "vertical"], axesfont = ["Arial",
      15], legendstyle = [font = ["Arial", 15], location = right], labels = [[mm], Deflection[mm]],
      labelfont = ["Arial", 15]);

  sd1 := sh - w1 :
  sd2 := sh - w2 :
  sd3 := sh - w3 :
  sd4 := sh - w4 :
  with(plots) :
  SD1 := plot(sd1, x = 0..d, color = black, thickness = 4) :
  SD2 := plot(sd2, x = d..2·d, color = black, thickness = 4) :
  SD3 := plot(sd3, x = 2·d..3·d, color = black, thickness = 4) :
  SD4 := plot(sd4, x = 3·d..4·d, color = black, thickness = 4) :
  display({SD1, SD2, SD3, SD4}, labeldirections = ["horizontal", "vertical"], axesfont = ["Arial",
      15], legendstyle = [font = ["Arial", 15], location = right], labels = [[mm], Difference[mm]],
      labelfont = ["Arial", 15]);
```


Deflection [mm]
40
30
20
10
0
0
100
200
300
400
500
600
[mm]



Plot angle (optional):

```
> with(plots):
  PHI1 := plot(phi1, x = 0..d, color = blue, thickness = 2):
  PHI2 := plot(phi2, x = d..2·d, color = cyan, thickness = 2):
  PHI3 := plot(phi3, x = 2·d..3·d, color = blue, thickness = 2):
  PHI4 := plot(phi4, x = 3·d..4·d, color = cyan, thickness = 2):
  display({PHI1, PHI2, PHI3, PHI4}, labeldirections = ["horizontal", "vertical"], axesfont
      = ["Arial", 10], legendstyle = [font = ["Arial", 10], location = right], labels = [[mm],
      Rotation[rad]], labelfont = ["Arial", 10]):
```

Plot curvature (optional):

```
> with(plots):
  K1 := plot(k1, x = 0..d, color = blue, thickness = 2):
  K2 := plot(k2, x = d..2·d, color = cyan, thickness = 2):
  K3 := plot(k3, x = 2·d..3·d, color = blue, thickness = 2):
  K4 := plot(k4, x = 3·d..4·d, color = cyan, thickness = 2):
  display({K1, K2, K3, K4}, labeldirections = ["horizontal", "vertical"], axesfont = ["Arial", 10],
      legendstyle = [font = ["Arial", 10], location = right], labels = [[mm], Curvature [1/mm]],
```

```
        labelfont = ["Arial", 10]) :
```

Plot moment (optional):

```
> with(plots) :
  M1 := plot(m1, x = 0..d, color = blue, thickness = 2) :
  M2 := plot(m2, x = d..2·d, color = cyan, thickness = 2) :
  M3 := plot(m3, x = 2·d..3·d, color = blue, thickness = 2) :
  M4 := plot(m4, x = 3·d..4·d, color = cyan, thickness = 2) :
  display({M1, M2, M3, M4}, labeldirections = ["horizontal", "vertical"], axesfont = ["Arial", 10],
      legendstyle = [font = ["Arial", 10], location = right], labels = [[mm], Moment [Nmm]],
      labelfont = ["Arial", 10]) :
```

Plot shear (optional):

```
> with(plots) :
  V1 := plot(v1, x = 0..d, color = blue, thickness = 2) :
  V2 := plot(v2, x = d..2·d, color = cyan, thickness = 2) :
  V3 := plot(v3, x = 2·d..3·d, color = blue, thickness = 2) :
  V4 := plot(v4, x = 3·d..4·d, color = cyan, thickness = 2) :
  display({V1, V2, V3, V4}, labeldirections = ["horizontal", "vertical"], axesfont = ["Arial", 10],
      legendstyle = [font = ["Arial", 10], location = right], labels = [[mm], Shear [N]], labelfont
      = ["Arial", 10]) :
```

Displacements of the beams (strips) cause deviation of the required shape. This can be compensated by adjusting the the support heights. Based on a approximation of the horizontal displacement of supprts and the angle at the supports the compensation is calculated. Since the beams (strips) deflect relative large distance there will be a horizontal displacement as well. This is not included in the solution of the equations. An approximation of the horizontal displacement is made by calculating the the length of the function between the support distance and comparing to the original length.

Define the approximation of the horizontal displacements:

```
> with(Student[Calculus1]) :
  L1 := ApproximateInt(√(1 + (phi1)^2), x = 0 .. d, method = trapezoid) :
  L2 := ApproximateInt(√(1 + (phi2)^2), x = d .. 2·d, method = trapezoid) :
  L3 := ApproximateInt(√(1 + (phi3)^2), x = 2·d .. 3·d, method = trapezoid) :
  L4 := ApproximateInt(√(1 + (phi4)^2), x = 3·d .. 4·d, method = trapezoid) :
  HDS1 := evalf(L1 + L2 - 2·d);
  HDS2 := evalf(L2 - d);
  HDS3 := 0;
  HDS4 := evalf(d - L3);
  HDS5 := evalf(2·d - L3 - L4);
```

$$HDS1 := 4.6171089$$
$$HDS2 := 0.5935072$$
$$HDS3 := 0$$
$$HDS4 := -0.5935072$$
$$HDS5 := -4.6171090 \quad \textbf{(5)}$$

Define approximation of horizontal displacement at midspan of beams (strips):

```
> LM1 := ApproximateInt(√(1 + (phi1)^2), x = 0.5·d .. d, method = trapezoid) :
  LM2 := ApproximateInt(√(1 + (phi2)^2), x = 1.5·d .. 2·d, method = trapezoid) :
```

$$LM3 := ApproximateInt\left(\sqrt{1 + (phi3)^2}, x = 2\cdot d\ ..2.5\cdot d, method = trapezoid\right):$$
$$LM4 := ApproximateInt\left(\sqrt{1 + (phi4)^2}, x = 3\cdot d\ ..\ 3.5\cdot d, method = trapezoid\right):$$
$$HDM1 := evalf(LM1 + L2 - 1.5\cdot d);$$
$$HDM2 := evalf(LM2 - 0.5\cdot d);$$
$$HDM3 := evalf(0.5\cdot d - LM3);$$
$$HDM4 := evalf(1.5\cdot d - LM4 - L3);$$

$$HDM1 := 2.2225285$$
$$HDM2 := 0.06241589$$
$$HDM3 := -0.06241589$$
$$HDM4 := -2.2225286 \quad \textbf{(6)}$$

Calculate the angle at the supports:

> $x := 0:$
$AS1 := evalf(phi1);$
$x := d:$
$AS2 := evalf(phi1);$
$x := 2\cdot d:$
$AS3 := evalf(phi2);$
$x := 3\cdot d:$
$AS4 := evalf(phi4);$
$x := 4\cdot d:$
$AS5 := evalf(phi4);$

$$AS1 := -0.2613695440$$
$$AS2 := -0.1649576139$$
$$AS3 := 1.\ 10^{-10}$$
$$AS4 := 0.164957615$$
$$AS5 := 0.261369547 \quad \textbf{(7)}$$

Calculate compensation of the height of the supports:

> $d11 := \cos(AS1)\cdot HDS1:$
$d21 := \sin(AS1)\cdot HDS1:$
$d31 := \sin(AS1)\cdot l1:$
$d41 := \cos(AS1)\cdot l1:$
$d51 := d11 - d31:$
$d61 := \sqrt{l2^2 - d51^2}:$
$d71 := d21 + d41 + d61:$
$HCS1 := evalf(l1 + l2 - d71);$
$dR11 := \cos(AS1)\cdot HDS1:$
$dR21 := \sin(AS1)\cdot HDS1:$
$dR31 := \sin(AS1)\cdot l1:$
$dR41 := \cos(AS1)\cdot l1:$
$dR51 := dR11 - dR31:$
$dR61 := \sqrt{l3^2 - dR51^2}:$
$dR71 := dR21 + dR41 + dR61:$

$$HCRS1 := evalf(\ l1 + l3 - dR71);$$
$$d12 := \cos(AS2)\cdot HDS2:$$
$$d22 := \sin(AS2)\cdot HDS2:$$
$$d32 := \sin(AS2)\cdot l1:$$
$$d42 := \cos(AS2)\cdot l1:$$
$$d52 := d12 - d32:$$
$$d62 := \sqrt{l2^2 - d52^2}:$$
$$d72 := d22 + d42 + d62:$$
$$HCS2 := evalf(\ l1 + l2 - d72);$$
$$dR12 := \cos(AS2)\cdot HDS2:$$
$$dR22 := \sin(AS2)\cdot HDS2:$$
$$dR32 := \sin(AS2)\cdot l1:$$
$$dR42 := \cos(AS2)\cdot l1:$$
$$dR52 := dR12 - dR32:$$
$$dR62 := \sqrt{l3^2 - dR52^2}:$$
$$dR72 := dR22 + dR42 + dR62:$$
$$HCRS2 := evalf(\ l1 + l3 - dR72);$$
$$HCS3 := l1 - \frac{l1}{\cos(AS3)};$$
$$d14 := -\cos(AS4)\cdot HDS4:$$
$$d24 := \sin(AS4)\cdot HDS4:$$
$$d34 := \sin(AS4)\cdot l1:$$
$$d44 := \cos(AS4)\cdot l1:$$
$$d54 := d14 + d34:$$
$$d64 := \sqrt{l2^2 - d54^2}:$$
$$d74 := d24 + d44 + d64:$$
$$HCS4 := evalf(\ l1 + l2 - d74);$$
$$dR14 := -\cos(AS4)\cdot HDS4:$$
$$dR24 := \sin(AS4)\cdot HDS4:$$
$$dR34 := \sin(AS4)\cdot l1:$$
$$dR44 := \cos(AS4)\cdot l1:$$
$$dR54 := dR14 + dR34:$$
$$dR64 := \sqrt{l3^2 - dR54^2}:$$
$$dR74 := dR24 + dR44 + dR64:$$
$$HCR4 := evalf(\ l1 + l3 - dR74);$$
$$d15 := -\cos(AS5)\cdot HDS5:$$
$$d25 := \sin(AS5)\cdot HDS5:$$
$$d35 := \sin(AS5)\cdot l1:$$
$$d45 := \cos(AS5)\cdot l1:$$
$$d55 := d15 + d35:$$
$$d65 := \sqrt{l2^2 - d55^2}:$$
$$d75 := d25 + d45 + d65:$$
$$HCS5 := evalf(\ l1 + l2 - d75);$$
$$dR15 := -\cos(AS5)\cdot HDS5:$$
$$dR25 := \sin(AS5)\cdot HDS5:$$
$$dR35 := \sin(AS5)\cdot l1:$$
$$dR45 := \cos(AS5)\cdot l1:$$
$$dR55 := dR15 + dR35:$$
$$dR65 := \sqrt{l3^2 - dR55^2}:$$

$dR75 := dR25 + dR45 + dR65$:
$HCRS5 := evalf(l1 + l3 - dR75)$;

$$HCS1 := 2.1456468$$
$$HCRS1 := 2.1046617$$
$$HCS2 := 0.3963633$$
$$HCRS2 := 0.3901241$$
$$HCS3 := 0.$$
$$HCS4 := 0.3963633$$
$$HCR4 := 0.3901241$$
$$HCS5 := 2.1456469$$
$$HCRS5 := 2.1046618 \quad \textbf{(8)}$$

Calculate support reactions:

> $x := 0$:
$RS1 := evalf(0 - v1)$;
$x := d$:
$RS2 := evalf(v1 - v2)$;
$x := 2 \cdot d$:
$RS3 := evalf(v2 - v3)$;
$x := 3 \cdot d$:
$RS4 := evalf(v3 - v4)$;
$x := 4 \cdot d$:
$RS5 := evalf(v4 - 0)$;
$Rtot := evalf(R1 + R2 + R3 + R4 + R5)$;

$$RS1 := -1.150143746$$
$$RS2 := 4.127090034$$
$$RS3 := 0.04610742417$$
$$RS4 := 4.127090033$$
$$RS5 := -1.150143746$$
$$Rtot := R1 + R2 + R3 + R4 + R5 \quad \textbf{(9)}$$

Check vertical equilibrium.

> $x :='x'$:
$VE1 := evalf\left(\int_0^d q1 \, dx\right)$:
$VE2 := evalf\left(\int_d^{2 \cdot d} q2 \, dx\right)$:
$VE3 := evalf\left(\int_{2 \cdot d}^{3 \cdot d} q3 \, dx\right)$:
$VE4 := evalf\left(\int_{3 \cdot d}^{4 \cdot d} q4 \, dx\right)$:
$VEtot := evalf(VE1 + VE2 + VE3 + VE4)$;

$$VEtot := -6.000000000 \quad \textbf{(10)}$$

Calculate heights of displaced support:

```
> x := HDS1 :
  HS1 := evalf(w1);
  x := d + HDS2 :
  HS2 := evalf(w2);
  x := 2· d :
  HS3 := evalf(w3);
  x := 3·d + HDS4 :
  HS4 := evalf(w3);
  x := 4·d + HDS5 :
  HS5 := evalf(w4);
  x :='x':
```

$$HS1 := 1.206672678$$
$$HS2 := 34.84764309$$
$$HS3 := 46.06000014$$
$$HS4 := 34.84764317$$
$$HS5 := 1.2066713 \quad \textbf{(11)}$$

Calculate height of displaced at midspan:

```
> x := 0.5·d + HDM1 :
  HSM1 := evalf(w1);
  x := 1.5 d + HDM2 :
  HSM2 := evalf(w2);
  x := 2.5 d  + HDM3 :
  HSM3 := evalf(w3);
  x := 3.5 d + HDM4 :
  HSM4 := evalf(w4);
```

$$HSM1 := 19.64821753$$
$$HSM2 := 43.43397002$$
$$HSM3 := 43.43397007$$
$$HSM4 := 19.6482173 \quad \textbf{(12)}$$

```
>
```

Appendix E

Accuracy of measurements with 3D scanner and processing data

The geometry of the mould during tests was by measurements with a 3D scanner. The used scanner was a Faro laser scanner Photon120/20. This scanner is capable of measuring millions of points. At the end of this annex a tech sheet is included of the manufacturer FARO [2012]. According to this information the accuracy of the laser is 2 [mm] at a distance of 25 [m]. However this accuracy cannot always be obtained. To determine the accuracy for application of the flexible mould test scans were made as shown in Figure E-1. To determine horizontal displacement of locations on the mould surface it is required to obtain specific locations. Therefore targets were used. There are different types of targets that can be used for this purpose. Two of these have been compared to determine the most suitable for application with the flexible mould. The two type of targets are half spheres and black and white flat targets as in Figure E-2. The position of the target is shown in Figure E-3. After the scans are made the data which is available is a list of coordinates of many points. Also called a point cloud. Since not all data is needed it must be processed using point cloud modification software. Here the program Leica Cyclone was used Geosystems [2013]. This software was able to select a region of points.

E-1 Comparison of targets

The two types of target were placed on a test sheet and then scanned. Two different diameters, 30 and 40 millimeter, of half spheres were tested and compared to flat black and white targets. A location could be obtained from the half spheres by fitting a virtual sphere onto the data of the spheres. This is a method which was included in the software package that was used. The center point of the sphere was taken as the location of the point that was required. The black and white targets were made. The scanner does not only determines the location of point, but also the gray scale. This property was used to isolate the targets from the rest of the mould surface. The scanned data was imported in Matlab and analyzed using a threshold value for

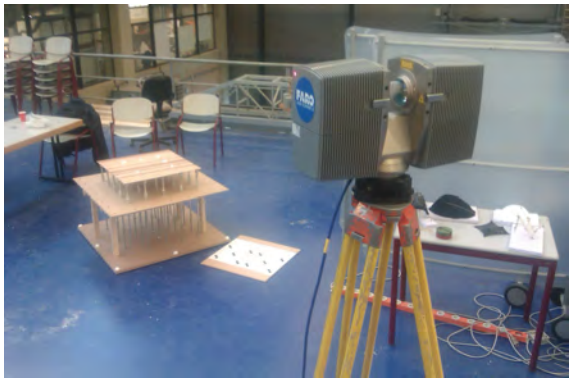


Figure E-1: Test scan of mould and targets.

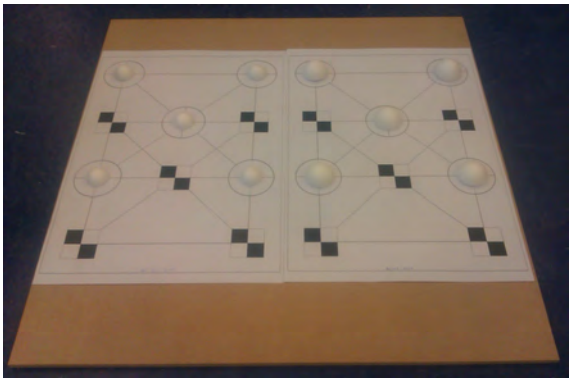


Figure E-2: Two types of targets were scanned and compared for accuracy.

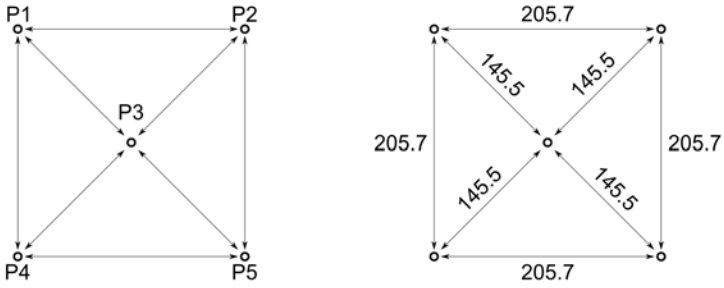


Figure E-3: Measured distances between targets

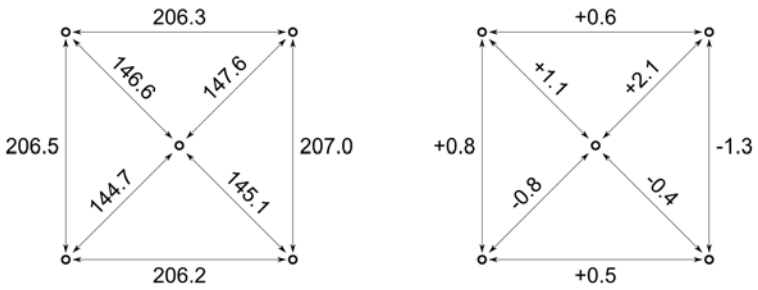


Figure E-4: Distances and deviation of sphere fitting d = 40mm. Average deviation 0.95 [mm].

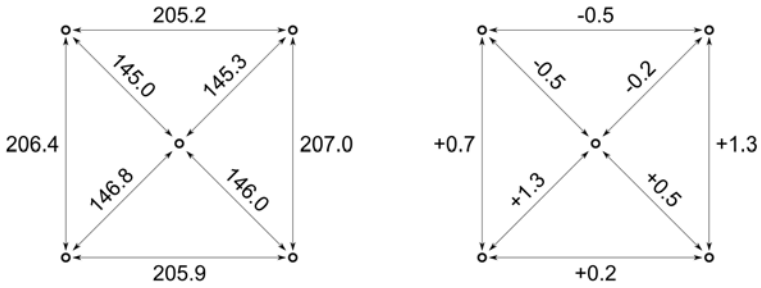


Figure E-5: Distances and deviation of sphere fitting d = 30mm. Average deviation 0.95 [mm].

the distinction between black and white points. The middle of a target was determined by taking the average coordinate of all isolated black points of one target. The Matlab sheet is included at the end of this Appendix.

The result of processing the data for the half spheres can be found in the Figures E-4 to E-6. For both the spheres with 30 and 40 millimeter half the average error is 0.95 millimeter. These measurement are the distance between two points on horizontal direction. It was found that for the vertical direction the fit of the sphere object was as accurate. The spheres were fitted too low varying between 5 and 10 millimeter. This could be due the amount of points that can be used. This is reduced since half spheres are used and the scanner is only capable of measuring point from its own viewpoint. Points at the back of the sphere cannot be measured. From a practical point of view these target are difficult to place accurately.

The flat black and white targets had an average error on 1 millimeter, which more than for the half sphere targets. Results can be seen in Figure E-7. In vertical direction a difference of 0.6 millimeter was found between the average target height and the surrounding surface. The flat targets were easier to place and more accurate since the location can be measured easier by hand. These two aspect are an advantage compared to the half sphere targets. Therefore these targets were chosen to implement in the test with the flexible mould.

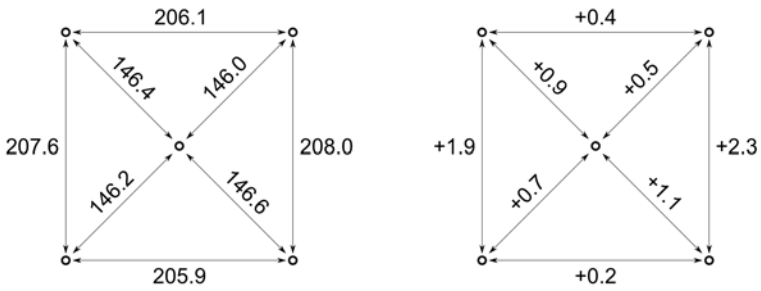


Figure E-6: Distances and deviation of square targets. Average deviation 1.00 [mm].

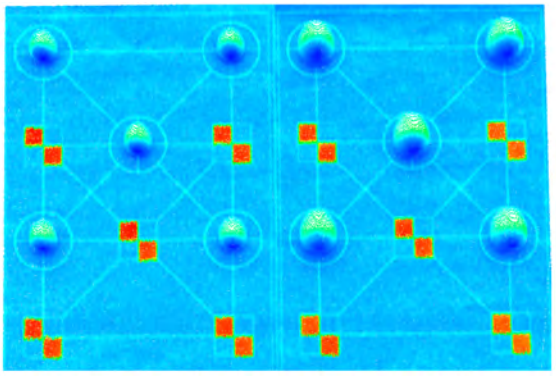


Figure E-7: Pointcloud in Leica Cyclone.

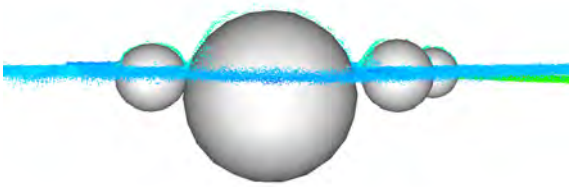


Figure E-8: Sphere fitted on data. Fit of spheres is too low varying between 5 and 10 millimeter

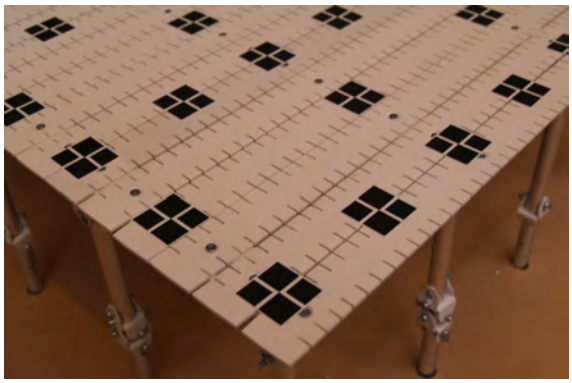


Figure E-9: Targets placed on the mould surface.

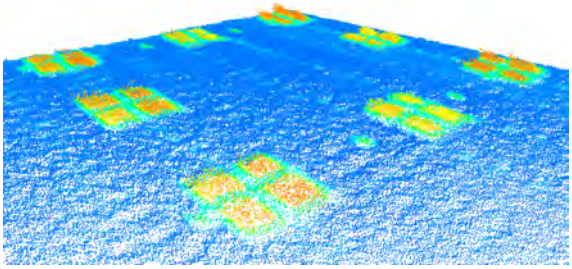


Figure E-10: Scan of the mould surface and targets. Points on the target have been measured in an area around the real target.

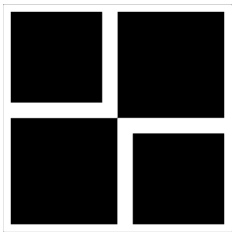


Figure E-11: Used target for test scans.

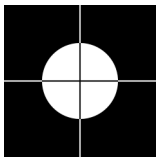


Figure E-12: Suggestion for alternative target that can be used for future scans.

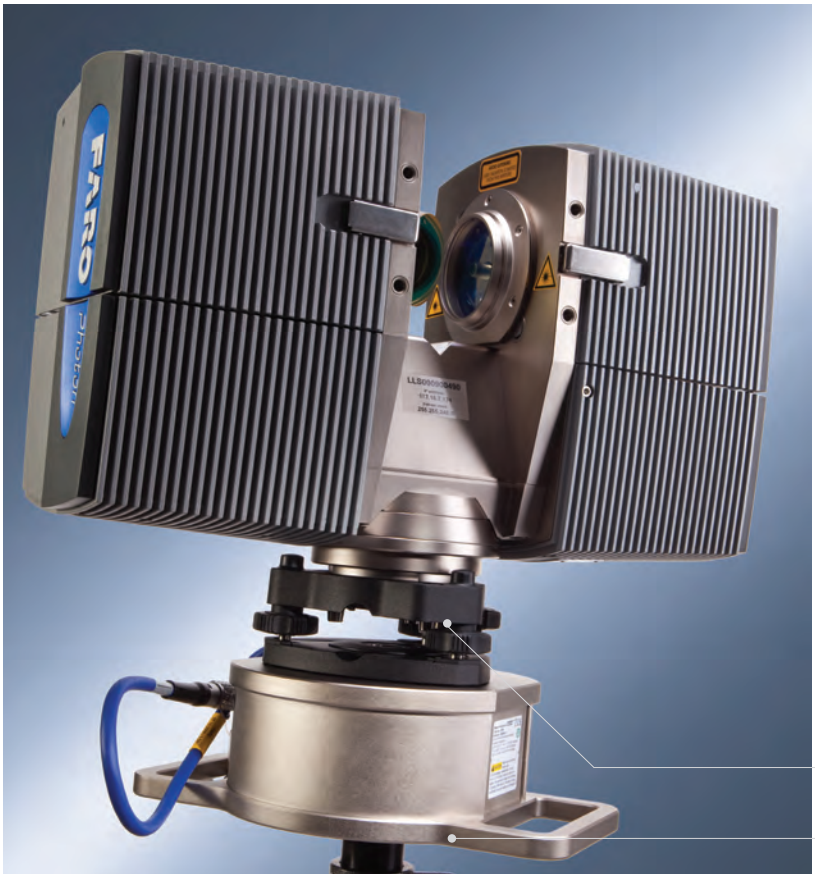
E-2 Accuracy of test results

To determine the accuracy of the scans a benchmark scan was made of the horizontal and leveled position of the mould surface. This position was setup by hand and distances between the target were 150 millimeter. The design of the targets was changed such that black area of the target is larger. Therefore more black points would be scanned and the average of all point would be a more detailed description of the location of the center of the target. However a difference was found in the result for the accuracy compared to the earlier made scans. The measured points of the targets were located in an area around the target. Possibly the laser did not reflect as well as for the earlier used targets. Another printer was used to print the targets. The targets and the scan can be seen in the Figures E-9 and E-10. The root mean square error of the targets located near the edge was calculated to determine the accuracy. The result is: 1.14 mm for the x-direction, 1.33 mm for y-direction and 1.54 for the z-direction.

The used targets are shown in Figure E-11. It was found that white a white surface results in more accurate measurements. Also the target could me smaller since the density of the scans can high. Therefore a suggestion for an alternative target is shown in Figure E-12. Instead of the black points, white points should be filtered. Small white lines were added so that the target can be positioned accurately.

www.faroasia.com

FARO Laser Scanner Photon 120/20



Longest Range 3D Phase-Shift Laser Scanner
Produce virtual images comprised of millions of 3D measurement points collected within an unprecedented range - up to 120m (395ft.)[1]

High Speed Survey and Inspection
Scan at rates of up to 976,000 points-per-second

Speed Control
Balance speed and scan quality according to application

High Accuracy
± 2mm ranging error[2] at 25 m

Best-in-Class Field-of-View
360º horizontal and 320º vertical - the largest field-of-view on the market

Modular Design
Removable sealed modules for convenient system upgrade and maintenance

Wireless Operability
Independent web server; data recording on 80GB internal hard disk; control via iPod® touch or most wireless PDAs

Universal Quick Mount
For mounting on a surveyor tripod

Compact Power base (option)
Provides 6 hours of operation per charge

The Photon 120: Large Scale Scanning at its Fastest

A high-speed 3D scanner for full-detail survey and documentation! Utilizing non-contact laser technology, the FARO Photon generates highly detailed 3D replicas of complex environments and geometries in a matter of minutes. Photon recreates the real world and defines it within a virtual space. The resulting image is a collection of millions of 3D measurements, providing an accurate digital representation of as-built or as-is conditions. Capable of scanning at the blistering rate of 976,000 points-per-second and a maximum reach of 120m, the Photon 120 offers the most efficient method for documenting conditions in three dimensions.

Document with Confidence

Never again miss critical dimensions. With Photon, digitally capture all the required documentation for engineering, procurement, construction, and investigation - in complete detail. Replace cumbersome data collection via tape measures, laser range finders, digital cameras, and total stations that involve additional effort and risk. Photon, also available in a 20m model, is the ultimate digital documentation instrument - the only limit to what you can do is your imagination.

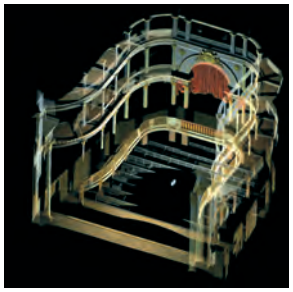
Additional Features

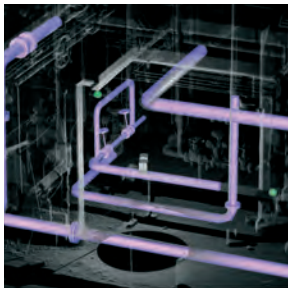
- Camera option for photo-realistic high-resolution color scans
- Mobile scanning interface for scanning along roads, rails, and tunnels with optional integration software
- Optimized for exceptional image quality in outdoor conditions
- Automatic target recognition, naming, and registration
- Crisp object definition

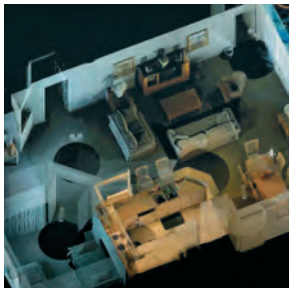
www.faro-LaserScanner-asia.com

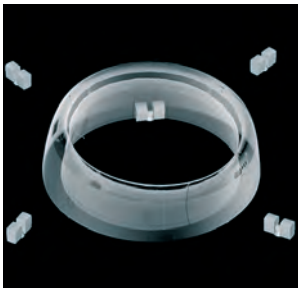
FARO Laser Scanner Photon 120/20

Applications


Commercial


Industrial


Residential


Manufacturing

Specifications

Ranging unit

Unambiguity interval: 153.49m (503.58ft)
Range[3] (Ph: 0.6m - 120m indoor or outdoor with low ambient light on 90% reflective surface
Range[3] (Photon 20): 0.6m - 20m on >2% matte reflective surface
Measurement speed: 122,000 / 244,000 / 488,000 / 976,000 points/sec
Ranging error[2]: ±2mm at 10m and 25m, each at 90% and 10% reflectivity
Ranging noise[4]: standard deviation
@10m - raw data: 0.8mm @ 90% refl. | 1.4mm rms @ 10% refl.
@10m - noise compressed[5]: 0.4mm @ 90% refl. | 0.7mm @ 10% refl.
@25m - raw data: 1.0mm @ 90% refl. | 2.7mm @ 10% refl.
@25m - noise compressed[5]: 0.5mm @ 90% refl. | 1.35mm @ 10% refl.

Deflection unit

Vertical field of view: 320°
Horizontal field of view: 360°
Vertical step size: 0.009° (40,000 3D pixel on 360°)
Horizontal step size: 0.009° (40,000 3D pixel on 360°)
Max. vertical scan speed: 2,880 rpm

Laser (Optical transmitter)

Laser power (cw Ø): 20 mW (Laser class 3R)
Wavelength: 785 nm
Beam divergence: Typical 0.16 mrad (0.009°)
Beam diameter at exit: 3.3 mm, circular

Handling of data

Internal PC: Intel Celeron-M 600MHz, 512 MB RAM, 80 GB HD
Data storage: Local: on internal hard disk drive (for most resolutions)
Remote: via Ethernet on external PC or laptop
Scanner control: via Ethernet or WLAN by PC or PDA, on local network, internet or independent operation

[1] Range specification applies to the Photon 120.
[2] Depends on ambient light, which can act as a source of noise. Bright ambient light (e.g. sunshine) may shorten the actual range of the scanner to lesser distances. In low ambient light, the range can be more than 120m for normal incidence on high-reflective surfaces.
*[3] **Ranging error** is defined as the maximum error in the distance measured by the scanner from its origin point to a point on a planar target.*
*[4] **Ranging noise** is defined as standard deviation of values about the best-fit plane.*
[5] A noise-compression algorithm may be activated to average points in sets of four or sixteen, thereby compressing raw data noise by a factor of 2 or 4.
[6] Based on vendor specification.
Subject to change without prior notice.
*More information available at **www.faro-LaserTracker-asia.com**.*

General

Power supply voltage: 24 V DC (Battery pack or AC converter)
Power consumption: ~60 W
Ambient temperature: 5° - 40° C
Humidity: Non condensing
Inclination sensor: Accuracy 0.02°; Resolution 0.001°; Range ±15°
Weight: 14.5 kg (31.97lb)

Size (LxWxH): 410mm x 160mm x 280mm
Maintenance calibration: Once a year
Exchange modules: Distance sensor / mirror axis / PC
Georeferencing: Yes
Cable connector: Located in scanner mount
Parallax-free color overlay (with color camera option)

GSA Contract Holder

ISO 9001 BUREAU VERITAS Certification

To learn more, visit www.faro-LaserScanner-asia.com

FARO Singapore Pte Ltd (Asia Pacific Headquarter)
China • India • Japan • Malaysia • Thailand • Vietnam • Indonesia • Philippines • South Korea • Australia
3 Changi South Street 2, Xilin Districentre Tower B, Singapore 486548
Tel: +65 6511 1350 Fax: +65 6543 0111
Email: salesap@faro.com

FARO Business Technologies India Pvt Ltd
E-12, B-1 Extension, Mohan Cooperative Industrial Estate, Mathura Road,
New Delhi-110044, India
Tel: +91 11 4646 5656 Fax: +91 11 4646 5660
Email : enquiry-india@faro.com

Global Head Quarters: USA - Lake Mary, Florida 32746 | Europe Head Quarters: Germany - Lingwiesenstr. 11/2 · 70825 Korntal-Münchingen

04REF101-031.pdf

Appendix F

Matlab sheet for processing scanned data

```
%% 2013 Peter Eigenraam and Daan Eigenraam
% Get target points from pointclouds based on gratscale value.
% Prepare pointcloud .pts files with indices from 01 to max 99.

% Clear
clc;clear;close all;
format long;

% Enter filename of scan, filename of targets, number of targets and
% threshold value for black and white.
filename_prefix_scan = 'scan_20130523014';
filename_prefix_surface = 'scan_20130523014_surface';
filename_prefix_surface_mid = 'scan_20130523014_surface_mid';
filename_prefix_target = 'target_';
file_extention = '.pts';
N = 68;
threshold_black = 120;
threshold_white = 180;

%% Filter black points from total scan.

% Read points from file.
%m_t = read_points([filename_prefix_scan, file_extention]);

%Filter out wanted points and put data in matrix P_t
%[x_t,y_t,z_t,l_t,r_t,g_t,b_t] = threshold_filter(m_t, threshold);

% Create a .pts file for the points in the targets.
%fileId = fopen([filename_prefix_scan, '_targets', file_extention], 'a');
%fprintf(fileId, '%d\n', length(x_t));
```

```
31 %fprintf(fileId, '%f %f %f %d %d %d %d\n', [x_t,y_t,z_t,l_t,r_t,g_t,b_t
       ]');
32 %fclose(fileId);
33
34 %% Filter white points from surface scan.
35
36 %Read points of files based on a generated string for file name
37 %Results are put into m where each cell is a vector
38 %m_s = read_points([filename_prefix_surface, file_extention]);
39
40 %Filter out wanted points and put data in matrix P_t
41 %[x_s,y_s,z_s,l_s,r_s,g_s,b_s] = threshold_filter(m_s, threshold_white,
       1);
42
43 % Create a .txt file for the resulting points.
44 %fileId = fopen([filename_prefix_surface, '.txt'], 'a');
45 %fprintf(fileId, '%f %f %f\n', [x_s,y_s,z_s]');
46 %fclose(fileId);
47
48 %% Filter white points from middle surface scan.
49
50 %Read points of files based on a generated string for file name
51 %Results are put into m where each cell is a vector
52 m_s_mid = read_points([filename_prefix_surface_mid, file_extention]);
53
54 %Filter out wanted points and put data in matrix P_t
55 [x_s_mid,y_s_mid,z_s_mid,l_s_mid,r_s_mid,g_s_mid,b_s_mid] =
       threshold_filter(m_s_mid, threshold_white, 1);
56
57 % Create a .txt file for the resulting points.
58 fileId = fopen([filename_prefix_surface_mid, '.txt'], 'a');
59 fprintf(fileId, '%f %f %f\n', [x_s_mid,y_s_mid,z_s_mid]');
60 fclose(fileId);
61
62 %% Filter black points from target scans.
63
64 % Define matrix for coordinates in x, y and z.
65 MP = zeros(N,3);
66
67 % Retreive and process data from files.
68 for i=1:N
69     %For files with index below 10.
70     if(i < 10)
71         %Read points of files based on a generated string for file name
72         %Results are put into m where each cell is a vector
73         m = read_points([filename_prefix_target, '0', int2str(i),
               file_extention]);
74
75         %Filter out wanted points and put data in matrix P
76         [x,y,z,l,r,g,b] = threshold_filter(m, threshold_black, 0);
77         P = [x,y,z];
78
79         %Calculate the average
```

```
80          MP(i,:) = mean(P);
81
82          % Create a .pts file for the points in the target.
83          %fileId = fopen([filename_prefix_target, '0', int2str(i),'
                _reduced', file_extention], 'a');
84          %fprintf(fileId, '%d\n', length(x));
85          %fprintf(fileId, '%f %f %f %d %d %d %d\n', [x,y,z,l,r,g,b]');
86          %fclose(fileId);
87
88          %For other files
89      else
90          %Read points of files based on a generated string for file name
91          %Results are put into m where each cell is a vector
92          m = read_points([filename_prefix_target, int2str(i),
                file_extention]);
93
94          %Filter out wanted points and put data in matrix P
95          [x,y,z,l,r,g,b] = threshold_filter(m, threshold_black, 0);
96          P = [x,y,z];
97          R = [x,y,z,l,r,g,b];
98
99          %Calculate the average
100         MP(i,:) = mean(P);
101
102         % Create a .pts file for the points in the target.
103         %fileId = fopen([filename_prefix_target, int2str(i),'_reduced',
                file_extention], 'a');
104         %fprintf(fileId, '%d\n', length(x));
105         %fprintf(fileId, '%f %f %f %d %d %d %d\n', [x,y,z,l,r,g,b]');
106         %fclose(fileId);
107     end
108 end
109
110 % Create .txt file of resulting target points.
111 fileId = fopen([filename_prefix_scan, '_points.txt'], 'a');
112 fprintf(fileId, '%f %f %f\n', [MP(:,1),MP(:,2),MP(:,3)]');
113 fclose(fileId);
```

```
1  function [ m ] = read_points( filename )
2  %READ_POINTS Summary of this function goes here
3  %   Detailed explanation goes here
4      fileID = fopen(filename);
5      m = textscan(fileID, '%f %f %f %f %f %f %f');
6      fclose(fileID);
7  end
```

```
1  function [ x, y, z, l, r, g, b ] = threshold_filter( m, threshold, hl )
2  %THRESHOLD_FILTER
3  %   This function determines which points are used
4
5  if hl == 0;
6      % Find indices of r values higher than threshold
7      I = find(m{5} < threshold);
```

```
8
9       % Select only the values that make the threshold
10      x = m{1}(I);
11      y = m{2}(I);
12      z = m{3}(I);
13      l = m{4}(I);
14      r = m{5}(I);
15      g = m{6}(I);
16      b = m{7}(I);
17
18  else
19      % Find indices of r values lower than threshold
20      I = find(m{5} > threshold);
21
22      % Select only the values that make the threshold
23      x = m{1}(I);
24      y = m{2}(I);
25      z = m{3}(I);
26      l = m{4}(I);
27      r = m{5}(I);
28      g = m{6}(I);
29      b = m{7}(I);
30  end
31
32  end
```

Appendix G

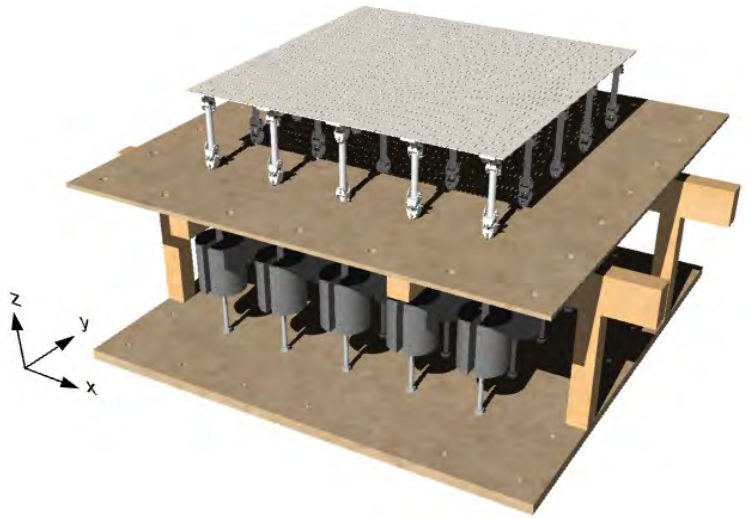
Test data

Mould setup data

Do not modify

Number of support in x-direction [-]:	5
Number of support in y-direction [-]:	5
Number of strips between supports:	5
Distance between supports [mm]:	150
Reference height [mm]:	150
Length connection top [mm]:	18,3
Length connection bottom [mm]:	127,7
Length connection bottom 2 [mm]:	145,7
Youngs modulus [N/mm^2]:	3851
Moment of innertia single strip [mm^4]:	10
Thickness concrete [mm]:	0
Density concrete [kg/m^3]:	2400

Flexible mould



Compensation for support length deviation (z-direction) [mm]

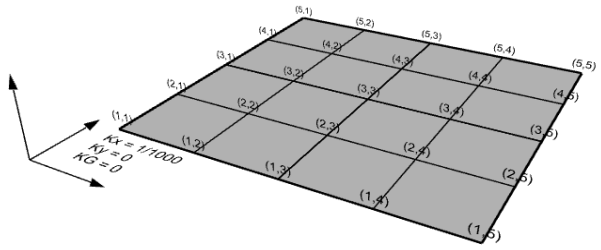
	1	2	3	4	5
5	6	6	7	9	7
4	9	8	4	6	7
3	5	5	0	5	2
2	5	7	6	7	9
1	5	4	4	5	5

Scan and results benchmark

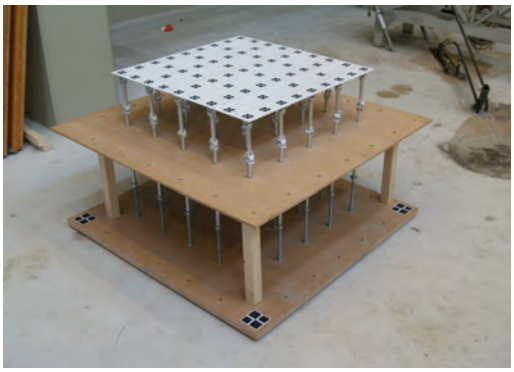
Do not modify
All measurements are relative to the middle support (3,3).

Min. radius of curvature [1/mm]:	inf
Max. radius of curvature [1/mm]:	inf

Surface



Scan



Location x-direction [mm]

	1		2		3		4		5
5	-300,00		-150,00		0,00		150,00		300,00
4	-300,00								300,00
3	-300,00				Ref.				300,00
2	-300,00								300,00
1	-300,00		-150,00		0,00		150,00		300,00

Measured location target x-direction [mm]

	1		2		3		4		5
5	-300,53		-149,65		0,59		150,99		300,15
4	-300,32								300,09
3	-300,98				Ref.				299,95
2	-301,47								298,60
1	-301,44		-151,92		-1,45		149,76		297,59

Difference location x-direction [mm]

	1		2		3		4		5
5	*0,53*		*-0,35*		*-0,59*		*-0,99*		*-0,15*
4	*0,32*								*-0,09*
3	*0,98*				Ref.				*0,05*
2	*1,47*								*1,40*
1	*1,44*		*1,92*		*1,45*		*0,25*		*2,41*

RMSE location (x-direction) [mm]: **1,14**

Location y-direction [mm]

	1		2		3		4		5
5	300,00		300,00		300,00		300,00		300,00
4	150,00								150,00
3	0,00				Ref.				0,00
2	-150,00								-150,00
1	-300,00		-300,00		-300,00		-300,00		-300,00

Measured location target y-direction [mm]

	1		2		3		4		5
5	298,82		300,05		300,27		299,93		298,09
4	150,76								148,81
3	-0,47				Ref.				-1,46
2	-150,86								-152,68
1	-300,50		-301,36		-300,97		-300,55		-302,78

Difference location y-direction [mm]

	1		2		3		4		5
5	*1,18*		*-0,05*		*-0,27*		*0,07*		*1,92*
4	*-0,75*								*1,19*
3	*0,47*				Ref.				*1,46*
2	*0,86*								*2,68*
1	*0,50*		*1,36*		*0,97*		*0,55*		*2,78*

RMSE location (y-direction) [mm]: **1,33**

Location z-direction [mm]

	1		2		3		4		5
5	0,00		0,00		0,00		0,00		0,00
4	0,00								0,00
3	0,00				Ref.				0,00
2	0,00								0,00
1	0,00		0,00		0,00		0,00		0,00

Measured location target y-direction [mm]

	1		2		3		4		5
5	1,60		0,74		1,14		*-1,72*		3,18
4	-0,89								2,01
3	0,43				Ref.				1,67
2	0,88								2,12
1	-0,92		0,07		0,27		-0,99		2,34

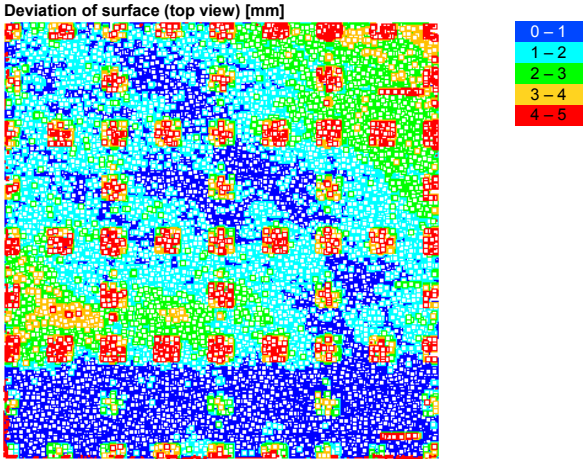
Difference location y-direction [mm]

	1		2		3		4		5
5	*-1,60*		*-0,74*		*-1,14*		*-1,72*		*-3,18*
4	*0,89*								*-2,01*
3	*-0,43*				Ref.				*-1,67*
2	*-0,88*								*-2,12*
1	*0,92*		*-0,07*		*-0,27*		*0,99*		*-2,34*

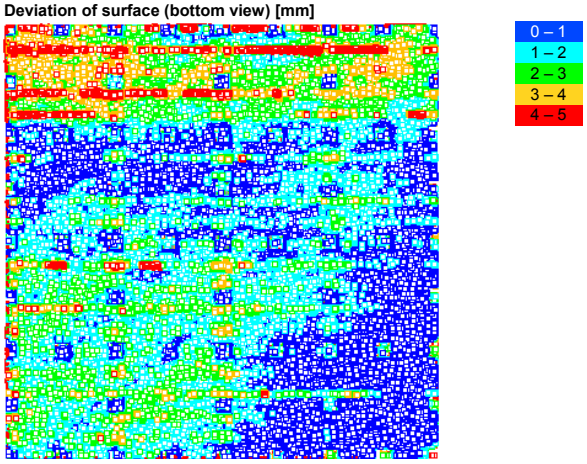
RMSE location (y-direction) [mm]: **1,54**

RMSE total [mm]: **2,33**

Deviation of surface (top view) [mm]



Deviation of surface (bottom view) [mm]



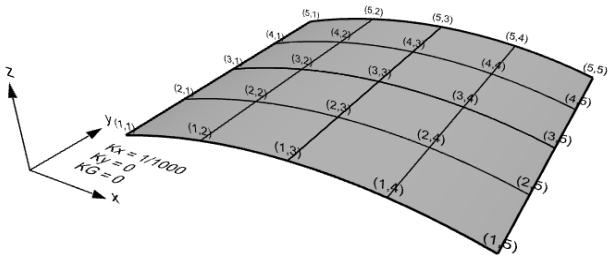
Number of points [–]:	**1.469.011**
Mean distance [mm]:	**0,73**
Median distance [mm]:	**0,59**
Standard deviation [mm]:	**0,59**

Setup shape 1

Do not modify

Distance between supports [mm]:	150
Reference height [mm]:	150
Load [N/m^2]:	-0
Load top edge strips [N/mm]:	0,00
Load top middle strips [N/mm]:	0,00
Youngs modulus [N/mm^2]:	3851
Moment of innertia top edge strip [mm^4]:	25
Moment of innertia top middle strip [mm^4]:	50
Moment of innertia lower strip [mm^4]:	50
Length connection top [mm]:	18,3
Length connection bottom [mm]:	127,7
Length connection bottom 2 [mm]:	145,7
Min. radius of curvature [1/mm]:	1000
Max. radius of curvature [1/mm]:	Inf

Surface



Support relative intersection heights [mm]

	1	2	3	4	5
5	0,00	34,75	46,06	34,75	0,00
4	0,00	34,75	46,06	34,75	0,00
3	0,00	34,75	46,06	34,75	0,00
2	0,00	34,75	46,06	34,75	0,00
1	0,00	34,75	46,06	34,75	0,00

Horizontal support displacement in x-direction [mm]

	1	2	3	4	5
5	4,63	0,60	0,00	-0,60	-4,63
4	4,63	0,60	0,00	-0,60	-4,63
3	4,63	0,60	0,00	-0,60	-4,63
2	4,63	0,60	0,00	-0,60	-4,63
1	4,63	0,60	0,00	-0,60	-4,63

Compensation supports in x-direction [mm]

	1	2	3	4	5
5	2,19	0,39	0,00	0,39	2,19
4	2,19	0,39	0,00	0,39	2,19
3	2,15	0,39	0,00	0,39	2,15
2	2,19	0,39	0,00	0,39	2,19
1	2,19	0,39	0,00	0,39	2,19

Support reactions for strips in x-direction [N]

	1	2	3	4	5
5	-0,87	1,21	-0,67	1,21	-0,87
4	-1,74	2,41	-1,35	2,41	-1,74
3	-1,74	2,41	-1,35	2,41	-1,74
2	-1,74	2,41	-1,35	2,41	-1,74
1	-0,87	1,21	-0,67	1,21	-0,87

Loading (qrep) at supports of strips in y-direction [N/mm]

	1	2	3	4	5
5	*0,0116*	*-0,0161*	*0,0090*	*-0,0161*	*0,0116*
4	*0,0116*	*-0,0161*	*0,0090*	*-0,0161*	*0,0116*
3	*0,0116*	*-0,0161*	*0,0090*	*-0,0161*	*0,0116*
2	*0,0116*	*-0,0161*	*0,0090*	*-0,0161*	*0,0116*
1	*0,0116*	*-0,0161*	*0,0090*	*-0,0161*	*0,0116*

Horizontal support displacement in y-direction [mm]

	1	2	3	4	5
5	0,00	0,00	0,00	0,00	0,00
4	0,00	0,00	0,00	0,00	0,00
3	0,00	0,00	0,00	0,00	0,00
2	0,00	0,00	0,00	0,00	0,00
1	0,00	0,00	0,00	0,00	0,00

Compensation supports in y-direction [mm]

	1	2	3	4	5
5	0,00	0,00	0,00	0,00	0,00
4	0,00	0,00	0,00	0,00	0,00
3	0,00	0,00	0,00	0,00	0,00
2	0,00	0,00	0,00	0,00	0,00
1	0,00	0,00	0,00	0,00	0,00

Support reactions for strips in y-direction [N]

	1	2	3	4	5
5	-0,68	0,95	-0,53	0,95	-0,68
4	-1,99	2,76	-1,54	2,76	-1,99
3	-1,62	2,24	-1,25	2,24	-1,62
2	-1,99	2,76	-1,54	2,76	-1,99
1	-0,68	0,95	-0,53	0,95	-0,68

Total compensation [mm]

	1	2	3	4	5
5	*2,19*	*0,39*	*0,00*	*0,39*	*2,19*
4	*2,19*	*0,39*	*0,00*	*0,39*	*2,19*
3	*2,15*	*0,39*	*0,00*	*0,39*	*2,15*
2	*2,19*	*0,39*	*0,00*	*0,39*	*2,19*
1	*2,19*	*0,39*	*0,00*	*0,39*	*2,19*

Setup height supports (including support length compensation) [mm]

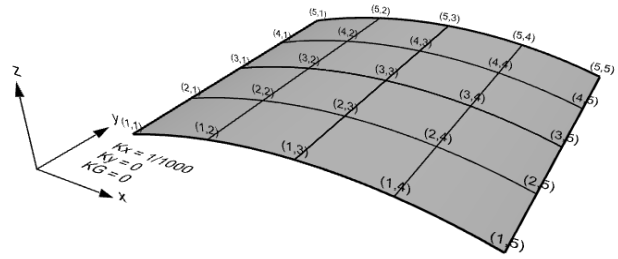
	1	2	3	4	5
5	*112,13*	*145,08*	*157,00*	*148,08*	*113,13*
4	*115,13*	*147,08*	*154,00*	*145,08*	*113,13*
3	*111,09*	*144,08*	*150,00*	*144,08*	*108,09*
2	*111,13*	*146,08*	*156,00*	*146,08*	*115,13*
1	*111,13*	*143,08*	*154,00*	*144,08*	*111,13*

Scan and results shape 1

Do not modify
All measurements are relative to the middle support (3,3).

Min. radius of curvature [1/mm]:	1000
Max. radius of curvature [1/mm]:	Inf

Surface



Scan

Predicted target displacement (x-direction) [mm]

	1	2	3	4	5
5	4,63				-4,63
4	4,63				-4,63
3	4,63		Ref.		-4,63
2	4,63				-4,63
1	4,63				-4,63

Measured target displacement (x-direction) [mm]

	1	2	3	4	5
5	3,66				-6,69
4	3,56				-6,28
3	3,83		Ref.		-6,29
2	4,39				-6,08
1	4,35				-5,85

Difference displacement (x-direction) [mm]

	1	2	3	4	5
5	*0,97*				*2,06*
4	*1,07*				*1,65*
3	*0,80*		Ref.		*1,66*
2	*0,23*				*1,45*
1	*0,27*				*1,22*

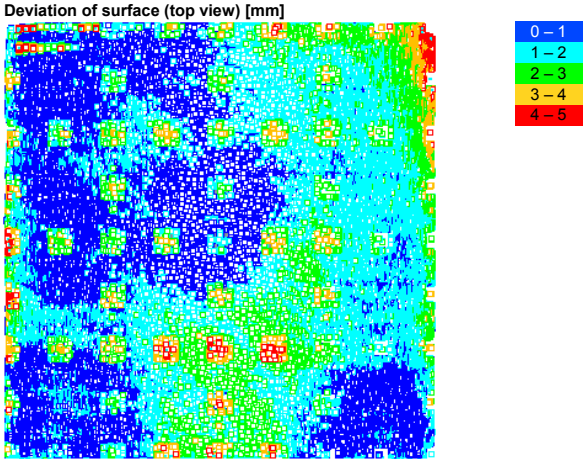
Ratio displacement difference vs prediction (x-direction) [%]

	1	2	3	4	5
5	*74%*				*69%*
4	*70%*				*74%*
3	*79%*		Ref.		*74%*
2	*95%*				*76%*
1	*94%*				*79%*

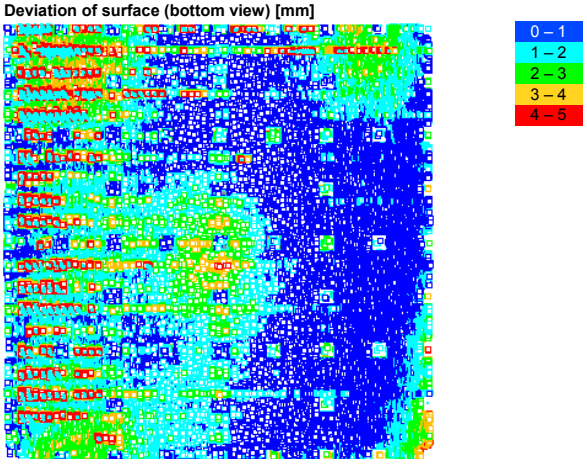
RMSE displacement (x-direction) [mm]: **1,27**

Average absolute accuracy of predicted displacement [mm]: **78%**

Deviation of surface (top view) [mm]

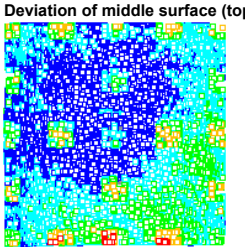


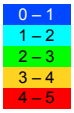
Deviation of surface (bottom view) [mm]



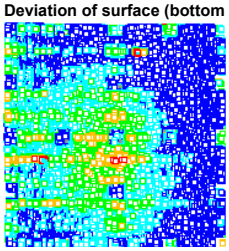
Number of points [–]:	**1.432.488**
Mean distance [mm]:	**0,91**
Median distance [mm]:	**0,75**
Standard deviation [mm]:	**0,73**

Deviation of middle surface (top view) [mm]





Deviation of surface (bottom view) [mm]



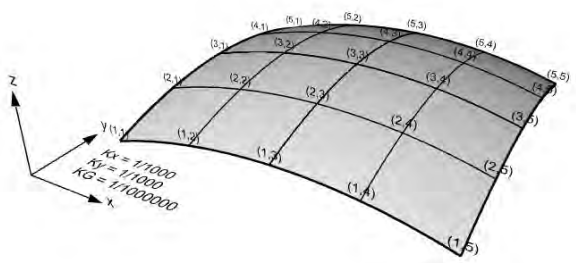
Number of points [–]:	**364.384**
Mean distance [mm]:	**0,78**
Median distance [mm]:	**0,67**
Standard deviation [mm]:	**0,58**

Setup element 2

Do not modify

Distance between supports [mm]:	150
Reference height [mm]:	150
Load [N/m^2]:	-0
Load top edge strips [N/mm]:	0,00
Load top middle strips [N/mm]:	0,00
Youngs modulus [N/mm^2]:	3851
Moment of innertia top edge strip [mm^4]:	25
Moment of innertia top middle strip [mm^4]:	50
Moment of innertia lower strip [mm^4]:	50
Length connection top [mm]:	18,3
Length connection bottom [mm]:	127,7
Length connection bottom 2 [mm]:	145,7
Min. radius of curvature [1/mm]:	1000
Max. radius of curvature [1/mm]:	Inf

Surface



Support relative intersection heights [mm]

	1	2	3	4	5
5	0,00	36,53	48,40	36,53	0,00
4	36,53	71,70	83,15	71,70	36,53
3	48,40	83,15	94,46	83,15	48,40
2	36,53	71,70	83,15	71,70	36,53
1	0,00	36,53	48,40	36,53	0,00

Horizontal support displacement in x-direction [mm]

	1	2	3	4	5
5	5,11	0,66	0,00	-0,66	-5,11
4	4,69	0,61	0,00	-0,61	-4,69
3	4,63	0,60	0,00	-0,60	-4,63
2	4,69	0,61	0,00	-0,61	-4,69
1	5,11	0,66	0,00	-0,66	-5,11

Compensation supports in x-direction [mm]

	1	2	3	4	5
5	2,50	0,44	0,00	0,44	2,50
4	2,23	0,40	0,00	0,40	2,23
3	2,15	0,39	0,00	0,39	2,15
2	2,23	0,40	0,00	0,40	2,23
1	2,50	0,44	0,00	0,44	2,50

Support reactions for strips in x-direction [N]

	1	2	3	4	5
5	-0,92	1,27	-0,72	1,27	-0,92
4	-1,74	2,39	-1,30	2,39	-1,74
3	-1,74	2,41	-1,35	2,41	-1,74
2	-1,74	2,39	-1,30	2,39	-1,74
1	-0,92	1,27	-0,72	1,27	-0,92

Loading (qrep) at supports of strips in y-direction [N/mm]

	1	2	3	4	5
5	*0,0122*	*-0,0170*	*0,0095*	*-0,0170*	*0,0122*
4	*0,0116*	*-0,0159*	*0,0086*	*-0,0159*	*0,0116*
3	*0,0116*	*-0,0161*	*0,0090*	*-0,0161*	*0,0116*
2	*0,0116*	*-0,0159*	*0,0086*	*-0,0159*	*0,0116*
1	*0,0122*	*-0,0170*	*0,0095*	*-0,0170*	*0,0122*

Horizontal support displacement in y-direction [mm]

	1	2	3	4	5
5	-5,12	-4,72	-4,64	-4,72	-5,12
4	-0,66	-0,61	-0,60	-0,61	-0,66
3	0,00	0,00	0,00	0,00	0,00
2	0,66	0,61	0,60	0,61	0,66
1	5,12	4,72	4,64	4,72	5,12

Compensation supports in y-direction [mm]

	1	2	3	4	5
5	2,57	2,18	2,19	2,18	2,57
4	0,43	0,41	0,38	0,41	0,43
3	0,00	0,00	0,00	0,00	0,00
2	0,43	0,41	0,38	0,41	0,43
1	2,57	2,18	2,19	2,18	2,57

Support reactions for strips in y-direction [N]

	1	2	3	4	5
5	-2,54	-0,77	-2,29	-0,77	-2,54
4	0,54	5,21	0,90	5,21	0,54
3	-3,04	0,86	-2,58	0,86	-3,04
2	0,54	5,21	0,90	5,21	0,54
1	-2,54	-0,77	-2,29	-0,77	-2,54

Total compensation [mm]

	1	2	3	4	5
5	*5,07*	*2,62*	*2,19*	*2,62*	*5,07*
4	*2,66*	*0,81*	*0,38*	*0,81*	*2,66*
3	*2,15*	*0,39*	*0,00*	*0,39*	*2,15*
2	*2,66*	*0,81*	*0,38*	*0,81*	*2,66*
1	*5,07*	*2,62*	*2,19*	*2,62*	*5,07*

Setup height supports (including support length compensation) [mm]

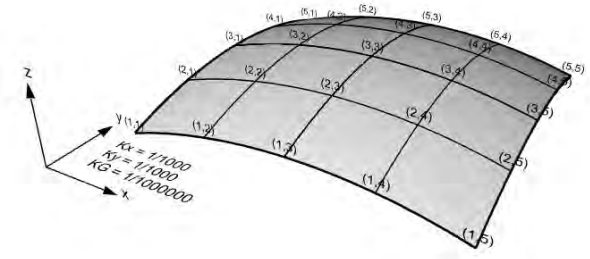
	1	2	3	4	5
5	*66,61*	*100,69*	*113,13*	*103,69*	*67,61*
4	*103,73*	*136,05*	*143,07*	*134,05*	*101,73*
3	*111,09*	*144,08*	*150,00*	*144,08*	*108,09*
2	*99,73*	*135,05*	*145,07*	*135,05*	*103,73*
1	*65,61*	*98,69*	*110,13*	*99,69*	*65,61*

Scan and results shape 2

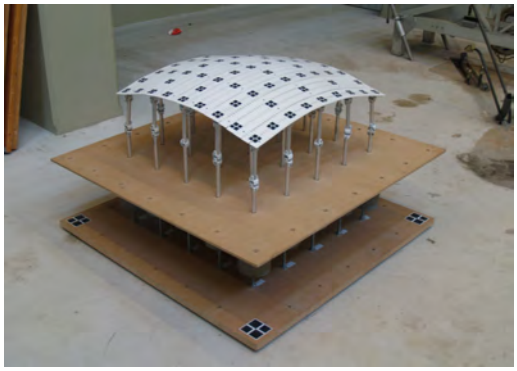
Do not modify
All measurements are relative to the middle support (3,3).

Min. radius of curvature [1/mm]:	1000
Max. radius of curvature [1/mm]:	Inf

Surface



Scan



Predicted target displacement x-direction [mm]

	1	2	3	4	5
5	5,11				-5,11
4	4,69				-4,69
3	4,63		Ref.		-4,63
2	4,69				-4,69
1	5,11				-5,11

Measured target displacement x-direction [mm]

	1	2	3	4	5
5	4,82				-9,71
4	3,51				-6,58
3	3,60		Ref.		-5,44
2	5,87				-1,87
1	10,40				0,90

Difference displacement x-direction [mm]

	1	2	3	4	5
5	*0,29*				*4,60*
4	*1,18*				*1,89*
3	*1,03*		Ref.		*0,81*
2	*-1,18*				*-2,82*
1	*-5,29*				*-6,01*

Ratio displacement difference vs prediction (x-direction) [%]

	1	2	3	4	5
5	*94%*				*53%*
4	*66%*				*71%*
3	*71%*		Ref.		*85%*
2	*80%*				*-51%*
1	*49%*				*-569%*

RMSE displacement (x-direction) [mm]: **3,18**

Average absolute accuracy of predicted displacement [mm]: **119%**

Predicted target displacement y-direction [mm]

	1		2		3		4		5
5	-5,12		-4,72		-4,64		-4,72		-5,12
4									
3					Ref.				
2									
1	5,12		4,72		4,64		4,72		5,12

Measured target displacement y-direction [mm]

	1		2		3		4		5
5	-4,70		-4,20		-4,21		-4,91		-8,59
4									
3					Ref.				
2									
1	3,15		2,64		3,70		3,72		3,97

Difference displacement y-direction [mm]

	1		2		3		4		5
5	*-0,42*		*-0,52*		*-0,43*		*0,19*		*3,47*
4									
3					Ref.				
2									
1	*1,98*		*2,08*		*0,94*		*1,01*		*1,16*

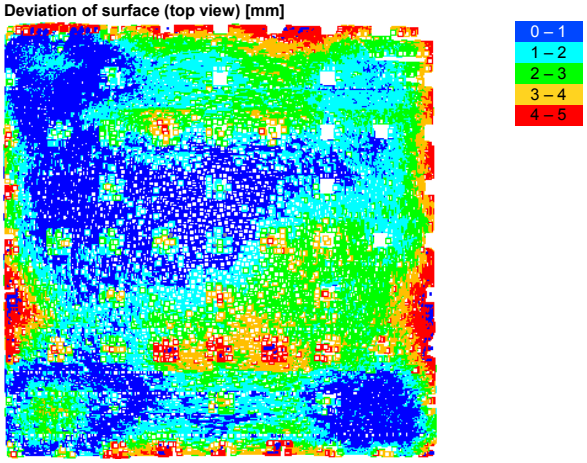
Ratio displacement difference vs prediction (y-direction) [%]

	1		2		3		4		5
5	*92%*		*89%*		*91%*		*96%*		*32%*
4									
3					Ref.				
2									
1	*61%*		*56%*		*80%*		*79%*		*77%*

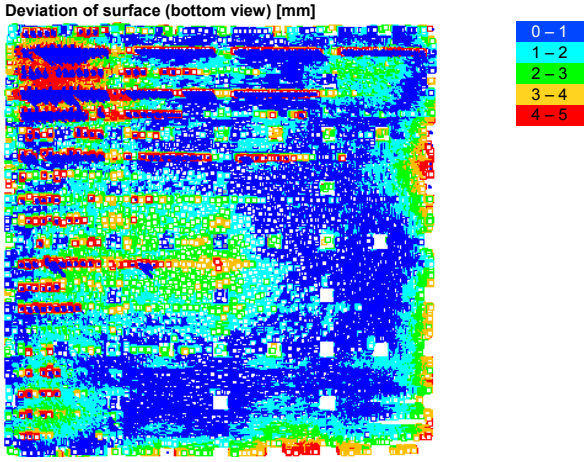
RMSE displacement (y-direction) [mm]: **1,56**

Average absolute accuracy of predicted displacement [mm]: **75%**

Deviation of surface (top view) [mm]

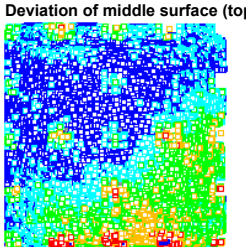


Deviation of surface (bottom view) [mm]

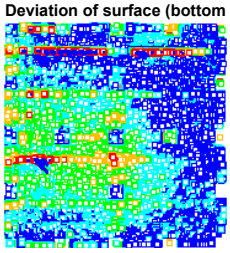


Number of points [–]:	**1.366.750**
Mean distance [mm]:	**1,25**
Median distance [mm]:	**1,04**
Standard deviation [mm]:	**0,98**

Deviation of middle surface (top view) [mm]



Deviation of surface (bottom view) [mm]



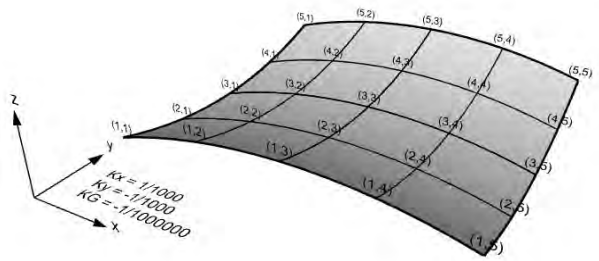
Number of points [–]:	**361.596**
Mean distance [mm]:	**0,96**
Median distance [mm]:	**0,82**
Standard deviation [mm]:	**0,72**

Setup element 3

Do not modify

Distance between supports [mm]:	150
Reference height [mm]:	150
Load [N/m^2]:	-0
Load top edge strips [N/mm]:	0,00
Load top middle strips [N/mm]:	0,00
Youngs modulus [N/mm^2]:	3851
Moment of innertia top edge strip [mm^4]:	25
Moment of innertia top middle strip [mm^4]:	50
Moment of innertia lower strip [mm^4]:	50
Length connection top [mm]:	18,3
Length connection bottom [mm]:	127,7
Length connection bottom 2 [mm]:	145,7
Min. radius of curvature [1/mm]:	1000
Max. radius of curvature [1/mm]:	1000

Surface



Support relative intersection heights [mm]

	1	2	3	4	5
5	46,06	80,81	92,12	80,81	46,06
4	11,31	46,06	57,37	46,06	11,31
3	0,00	34,75	46,06	34,75	0,00
2	11,31	46,06	57,37	46,06	11,31
1	46,06	80,81	92,12	80,81	46,06

Horizontal support displacement in x-direction [mm]

	1	2	3	4	5
5	4,63	0,60	0,00	-0,60	-4,63
4	4,63	0,60	0,00	-0,60	-4,63
3	4,63	0,60	0,00	-0,60	-4,63
2	4,63	0,60	0,00	-0,60	-4,63
1	4,63	0,60	0,00	-0,60	-4,63

Compensation supports in x-direction [mm]

	1	2	3	4	5
5	2,19	0,39	0,00	0,39	2,19
4	2,19	0,39	0,00	0,39	2,19
3	2,15	0,39	0,00	0,39	2,15
2	2,19	0,39	0,00	0,39	2,19
1	2,19	0,39	0,00	0,39	2,19

Support reactions for strips in x-direction [N]

	1	2	3	4	5
5	-0,87	1,21	-0,67	1,21	-0,87
4	-1,74	2,41	-1,35	2,41	-1,74
3	-1,74	2,41	-1,35	2,41	-1,74
2	-1,74	2,41	-1,35	2,41	-1,74
1	-0,87	1,21	-0,67	1,21	-0,87

Loading (qrep) at supports of strips in y-direction [N/mm]

	1	2	3	4	5
5	0,0116	-0,0161	0,0090	-0,0161	0,0116
4	0,0116	-0,0161	0,0090	-0,0161	0,0116
3	0,0116	-0,0161	0,0090	-0,0161	0,0116
2	0,0116	-0,0161	0,0090	-0,0161	0,0116
1	0,0116	-0,0161	0,0090	-0,0161	0,0116

Horizontal support displacement in y-direction [mm]

	1	2	3	4	5
5	-4,62	-4,65	-4,62	-4,65	-4,62
4	-0,59	-0,60	-0,59	-0,60	-0,59
3	0,00	0,00	0,00	0,00	0,00
2	0,59	0,60	0,59	0,60	0,59
1	4,62	4,65	4,62	4,65	4,62

Compensation supports in y-direction [mm]

	1	2	3	4	5
5	-0,57	-0,58	-0,57	-0,58	-0,57
4	0,17	0,16	0,17	0,16	0,17
3	0,00	0,00	0,00	0,00	0,00
2	0,17	0,16	0,17	0,16	0,17
1	-0,57	-0,58	-0,57	-0,58	-0,57

Support reactions for strips in y-direction [N]

	1	2	3	4	5
5	1,06	2,69	1,21	2,69	1,06
4	-4,40	0,35	-3,96	0,35	-4,40
3	-0,27	3,59	0,09	3,59	-0,27
2	-4,40	0,35	-3,96	0,35	-4,40
1	1,06	2,69	1,21	2,69	1,06

Total compensation [mm]

	1	2	3	4	5
5	1,62	-0,18	-0,57	-0,18	1,62
4	2,37	0,56	0,17	0,56	2,37
3	2,15	0,39	0,00	0,39	2,15
2	2,37	0,56	0,17	0,56	2,37
1	1,62	-0,18	-0,57	-0,18	1,62

Setup relative height supports (including support length compensation) [mm]

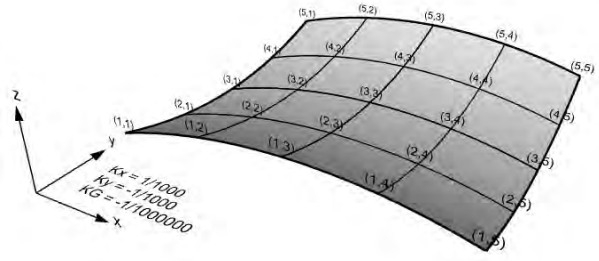
	1	2	3	4	5
5	*157,62*	*190,57*	*202,49*	*193,57*	*158,62*
4	*126,62*	*158,56*	*165,48*	*156,56*	*124,62*
3	*111,09*	*144,08*	*150,00*	*144,08*	*108,09*
2	*122,62*	*157,56*	*167,48*	*157,56*	*126,62*
1	*156,62*	*188,57*	*199,49*	*189,57*	*156,62*

Scan and results shape 3

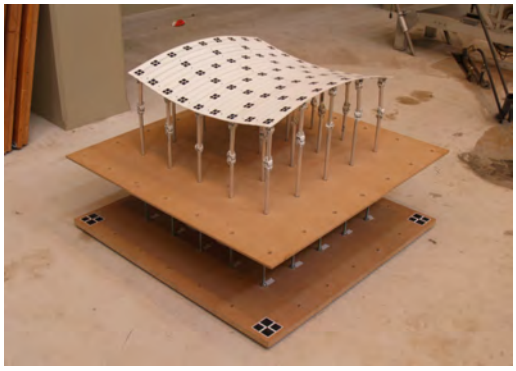
Do not modify
All measurements are relative to the middle support (3,3).

Min. radius of curvature [1/mm]:	1000
Max. radius of curvature [1/mm]:	1000

Surface



Scan



Predicted target displacement x-direction [mm]

	1	2	3	4	5
5	4,63				-4,63
4	4,63				-4,63
3	4,63		Ref.		-4,63
2	4,63				-4,63
1	4,63				-4,63

Measured target displacement x-direction [mm]

	1	2	3	4	5
5	3,81				-5,82
4	3,34				-5,47
3	3,67		Ref.		-5,34
2	3,25				-6,98
1	1,71				-8,93

Difference displacement x-direction [mm]

	1	2	3	4	5
5	*0,82*				*1,19*
4	*1,29*				*0,84*
3	*0,96*		Ref.		*0,71*
2	*1,38*				*2,35*
1	*2,91*				*4,30*

Ratio displacement difference vs prediction (x-direction) [%]

	1	2	3	4	5
5	*82%*				*74%*
4	*72%*				*82%*
3	*79%*		Ref.		*85%*
2	*70%*				*49%*
1	*37%*				*7%*

RMSE displacement (x-direction) [mm]: **2,01**

Average absolute accuracy of predicted displacement [mm]: **64%**

Predicted target displacement y-direction [mm]

	1	2	3	4	5
5	-4,62	-4,65	-4,62	-4,65	-4,62
4					
3			Ref.		
2					
1	4,62	4,65	4,62	4,65	4,62

Measured target displacement y-direction [mm]

	1	2	3	4	5
5	-5,77	-5,55	-5,18	-4,45	-4,40
4					
3			Ref.		
2					
1	2,51	4,61	5,54	6,28	3,04

Difference displacement y-direction [mm]

	1	2	3	4	5
5	*1,16*	*0,90*	*0,56*	*-0,20*	*-0,21*
4					
3			Ref.		
2					
1	*2,10*	*0,04*	*-0,92*	*-1,63*	*1,57*

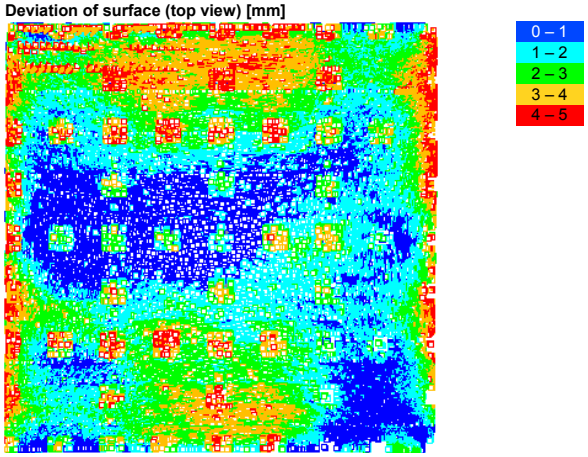
Ratio displacement difference vs prediction (y-direction) [%]

	1	2	3	4	5
5	*75%*	*81%*	*88%*	*96%*	*95%*
4					
3			Ref.		
2					
1	*54%*	*99%*	*80%*	*65%*	*66%*

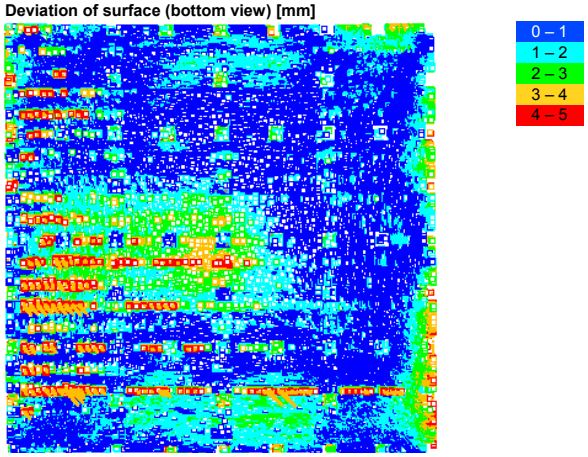
RMSE displacement (y-direction) [mm]: **1,14**

Average absolute accuracy of predicted displacement [mm]: **80%**

Deviation of surface (top view) [mm]

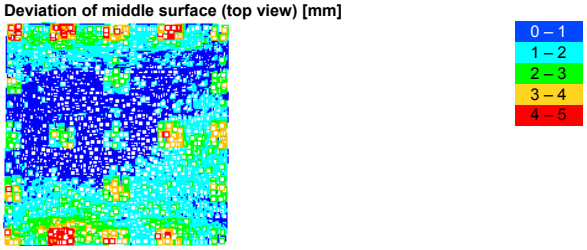


Deviation of surface (bottom view) [mm]

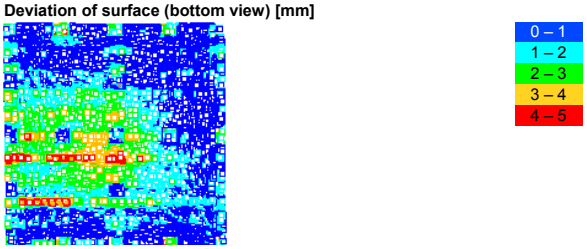


Number of points [–]:	**1.429.664**
Mean distance [mm]:	**1,22**
Median distance [mm]:	**1,05**
Standard deviation [mm]:	**0,90**

Deviation of middle surface (top view) [mm]



Deviation of surface (bottom view) [mm]



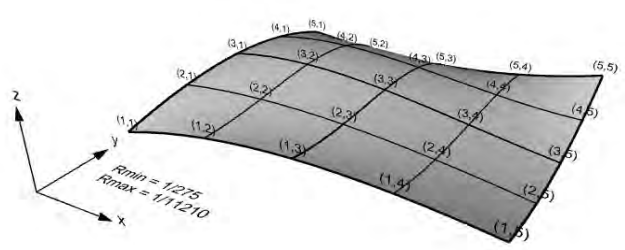
Number of points [–]:	**364.889**
Mean distance [mm]:	**0,86**
Median distance [mm]:	**0,72**
Standard deviation [mm]:	**0,67**

Setup element 4

Do not modify

Distance between supports [mm]:	150
Reference height [mm]:	150
Load [N/m^2]:	-0
Load top edge strips [N/mm]:	0,00
Load top middle strips [N/mm]:	0,00
Youngs modulus [N/mm^2]:	3851
Moment of innertia top edge strip [mm^4]:	25
Moment of innertia top middle strip [mm^4]:	50
Moment of innertia lower strip [mm^4]:	50
Length connection top [mm]:	18,3
Length connection bottom [mm]:	127,7
Length connection bottom 2 [mm]:	145,7
Min. radius of curvature [1/mm]:	600
Max. radius of curvature [1/mm]:	100000

Surface



Support relative intersection heights [mm]

	1	2	3	4	5
5	19,12	3,25	0,87	14,62	47,12
4	59,50	69,85	55,42	33,23	20,30
3	75,87	86,14	70,45	41,66	12,58
2	70,87	79,83	72,51	52,94	25,08
1	47,12	78,81	88,28	80,13	58,82

Horizontal support displacement in x-direction [mm]

	1	2	3	4	5
5	0,94	0,09	0,00	-0,73	-4,25
4	1,27	0,82	0,00	-1,64	-2,21
3	1,42	0,98	0,00	-2,74	-5,54
2	0,55	0,25	0,00	-1,30	-3,87
1	3,81	0,44	0,00	-0,29	-1,81

Compensation supports in x-direction [mm]

	1	2	3	4	5
5	0,03	0,04	-0,01	0,14	-0,49
4	0,29	-0,01	-0,21	0,39	0,24
3	0,30	-0,01	-0,27	1,04	1,66
2	0,13	0,00	-0,09	0,53	1,32
1	1,72	0,29	0,00	0,15	0,61

Support reactions for strips in x-direction [N]

	1	2	3	4	5
5	0,48	-0,56	0,39	-1,01	0,70
4	-2,03	3,67	-0,38	-2,15	0,89
3	-2,06	3,48	-0,65	-0,91	0,14
2	-1,24	1,89	-0,61	0,52	-0,56
1	-0,84	1,26	-0,44	0,48	-0,46

Loading (qrep) at supports of strips in y-direction [N/mm]

	1	2	3	4	5
5	-0,0064	0,0075	-0,0052	0,0135	-0,0094
4	0,0135	-0,0245	0,0025	0,0144	-0,0059
3	0,0137	-0,0232	0,0043	0,0061	-0,0009
2	0,0083	-0,0126	0,0041	-0,0035	0,0037
1	0,0112	-0,0168	0,0059	-0,0064	0,0061

Horizontal support displacement in y-direction [mm]

	1	2	3	4	5
5	-6,48	-15,81	-10,88	-1,43	-2,75
4	-1,08	-1,42	-1,09	-0,25	-0,32
3	0,00	0,00	0,00	0,00	0,00
2	0,21	0,14	0,03	0,47	0,66
1	2,12	0,16	0,89	2,96	4,46

Compensation supports in y-direction [mm]

	1	2	3	4	5
5	3,31	12,37	7,17	0,48	-0,17
4	0,67	1,30	0,88	0,10	0,11
3	-0,01	0,00	0,00	-0,02	0,00
2	0,14	0,02	0,03	0,10	0,16
1	0,84	0,00	0,04	-0,21	-0,53

Support reactions for strips in y-direction [N]

	1	2	3	4	5
5	-1,68	-4,04	-3,02	-1,73	1,90
4	0,92	10,00	5,62	-0,34	-0,74
3	-3,01	2,24	-1,23	-0,72	1,10
2	0,05	-0,12	-4,06	-1,98	-2,78
1	-1,96	1,65	1,01	1,68	1,23

Total compensation [mm]

	1	2	3	4	5
5	3,34	12,41	7,16	0,62	-0,65
4	0,96	1,29	0,67	0,49	0,36
3	0,29	-0,01	-0,27	1,02	1,66
2	0,27	0,02	-0,06	0,63	1,48
1	2,56	0,29	0,04	-0,06	0,08

Setup relative height supports (including support length compensation) [mm]

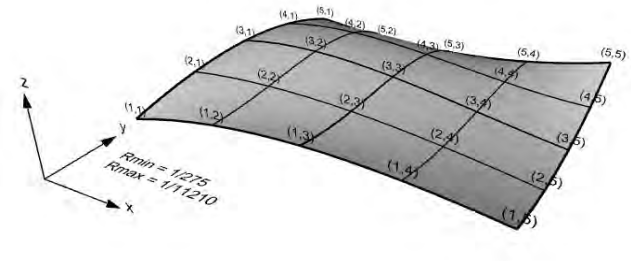
	1	2	3	4	5
5	*108,28*	*101,48*	*94,84*	*104,05*	*133,29*
4	*149,28*	*158,96*	*139,91*	*119,54*	*107,47*
3	*160,98*	*170,95*	*150,00*	*127,50*	*96,05*
2	*155,96*	*166,67*	*158,27*	*140,39*	*115,38*
1	*134,50*	*162,92*	*172,14*	*164,89*	*143,71*

Scan and results shape 4

Do not modify
All measurements are relative to the middle support (3,3).

Min. radius of curvature [1/mm]:	600
Max. radius of curvature [1/mm]:	100000

Surface



Scan

Predicted target displacement x-direction [mm]

	1	2	3	4	5
5	0,94				-4,25
4	1,27				-2,21
3	1,42		Ref.		-5,54
2	0,55				-3,87
1	3,81				-1,81

Measured target displacement x-direction [mm]

	1	2	3	4	5
5	-5,91				-6,39
4	-1,23				-3,54
3	0,93		Ref.		-5,85
2	1,93				-4,21
1	5,99				-0,86

Difference displacement x-direction [mm]

	1	2	3	4	5
5	*6,85*				*2,13*
4	*2,50*				*1,32*
3	*0,50*		Ref.		*0,31*
2	*-1,37*				*0,34*
1	*-2,18*				*-0,96*

Ratio displacement difference vs prediction (x-direction) [%]

	1	2	3	4	5
5	*-627%*				*50%*
4	*-97%*				*40%*
3	*65%*		Ref.		*94%*
2	*-148%*				*91%*
1	*43%*				*47%*

RMSE displacement (x-direction) [mm]: **2,60**

Average absolute accuracy of predicted displacement [mm]: **130%**

Predicted target displacement y-direction [mm]

	1	2	3	4	5
5	-6,48	-15,81	-10,88	-1,43	-2,75
4					
3			Ref.		
2					
1	2,12	0,16	0,89	2,96	4,46

Measured target displacement y-direction [mm]

	1	2	3	4	5
5	-9,53	-14,02	-9,22	-1,83	-2,91
4					
3			Ref.		
2					
1	0,13	0,08	0,97	2,38	2,41

Difference displacement y-direction [mm]

	1	2	3	4	5
5	*3,05*	*-1,79*	*-1,67*	*0,40*	*0,16*
4					
3			Ref.		
2					
1	*1,99*	*0,08*	*-0,08*	*0,58*	*2,05*

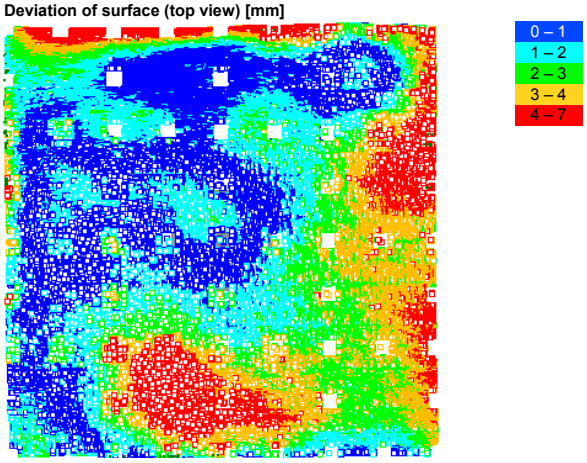
Ratio displacement difference vs prediction (y-direction) [%]

	1	2	3	4	5
5	*53%*	*89%*	*85%*	*72%*	*94%*
4					
3			Ref.		
2					
1	*6%*	*49%*	*91%*	*80%*	*54%*

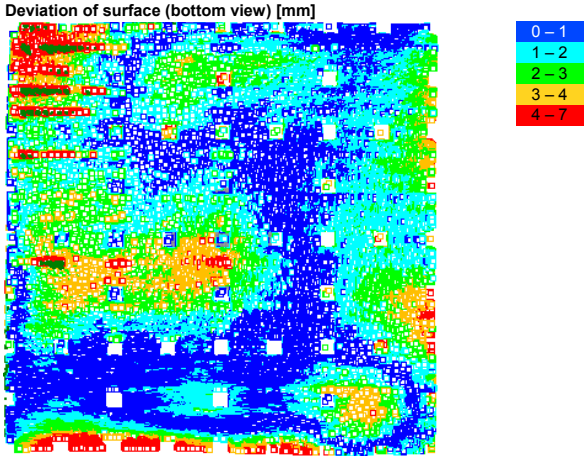
RMSE displacement (y-direction) [mm]: **1,55**

Average absolute accuracy of predicted displacement [mm]: **67%**

Deviation of surface (top view) [mm]

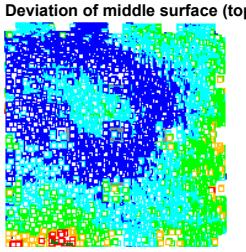


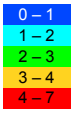
Deviation of surface (bottom view) [mm]



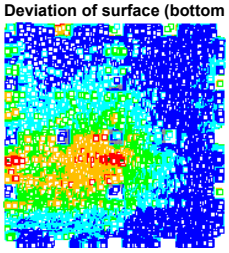
Number of points [–]:	**2.477.448**
Mean distance [mm]:	**1,63**
Median distance [mm]:	**1,44**
Standard deviation [mm]:	**1,15**

Deviation of middle surface (top view) [mm]





Deviation of surface (bottom view) [mm]



Number of points [–]:	**308.567**
Mean distance [mm]:	**1,11**
Median distance [mm]:	**0,98**
Standard deviation [mm]:	**0,78**

Appendix H

Geometrical definitions (In development)

Here the geometric definitions and assumptions for positive directions will be stated that are used in this thesis. Definitions for lines and surfaces are treated separately. For strain and curvature the difference can be seen by the used of double indices.

H-1 Coordinate systems

Geometric quantities and shapes are described in a global Cartesian coordinate system. Directions of the axis are chosen according to the right hand rule as in Figure H-1. To describe local properties of surfaces a local coordinates is chosen where the z-direction is perpendicular to the surface and where the x- and y-direction are tangent to the surface. In Figure H-2 the axis of the global coordinate system are indicated with $\bar{x}$, $\bar{y}$ and $\bar{z}$. The axis of the local coordinate system are indicated with x, y and z.

H-2 Geometric quantities for lines

Geometric quantities describe how objects move, rotate and deform. Here the positive direction of geometric quantities are defined. In Figure H-3 all positive directions are shown. Displacements are indicated with w;

$$w_x$$

$$w_y$$

$$w_z$$

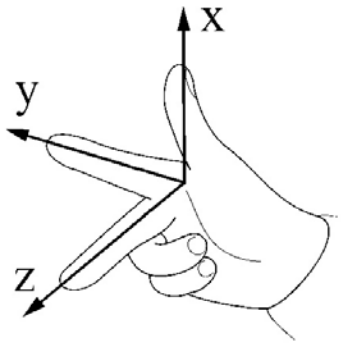


Figure H-1: Right hand rule for coordinate systems

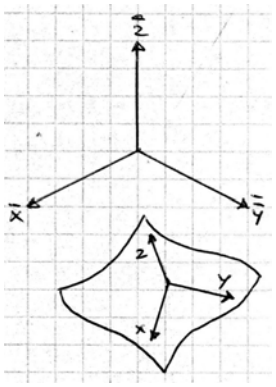


Figure H-2: Global and local coordinate system

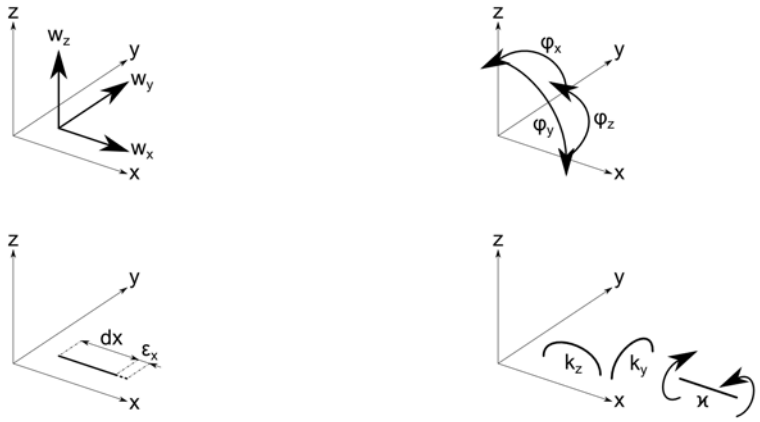


Figure H-3: Positive displacements (w), strain (ε), rotation (φ) and curvature (k) for lines.

Rotations are a change of location measured at a distance perpendicular to the displacement. Rotations indicated with the angle φ;

$$\varphi_x = \frac{dw_z}{dy} \text{ (used for torsion)} \tag{H-1}$$

$$\varphi_y = -\frac{dw_z}{dx} \tag{H-2}$$

$$\varphi_z = \frac{dw_y}{dx} \tag{H-3}$$

Strain is the change in length of a line where l_0 is the original length and l the length after straining. The strain is indicated with ε;

$$\begin{aligned} \varepsilon &= \frac{l - l_0}{l_0} \\ \varepsilon_x &= \frac{dw_x}{dx} \end{aligned} \tag{H-4}$$

Curvature is the change of rotation of a cross section. Curvature is indicated with k;

$$k_y = -\frac{d\varphi_z}{dx} \tag{H-5}$$

$$k_z = \frac{d\varphi_y}{dx} \tag{H-6}$$

(H-7)

Torsion is the change of the rotation of a cross section along an axis. Torsion is indicated with $\varkappa$;

$$\varkappa = \frac{d\varphi_x}{dx} \tag{H-8}$$

H-3 Geometric quantities for surfaces

Here the positive direction of geometric quantities for surfaces are defined. In Figure H-4 all positive directions are shown. Displacements are indicated with w;

w_x

w_y

w_z

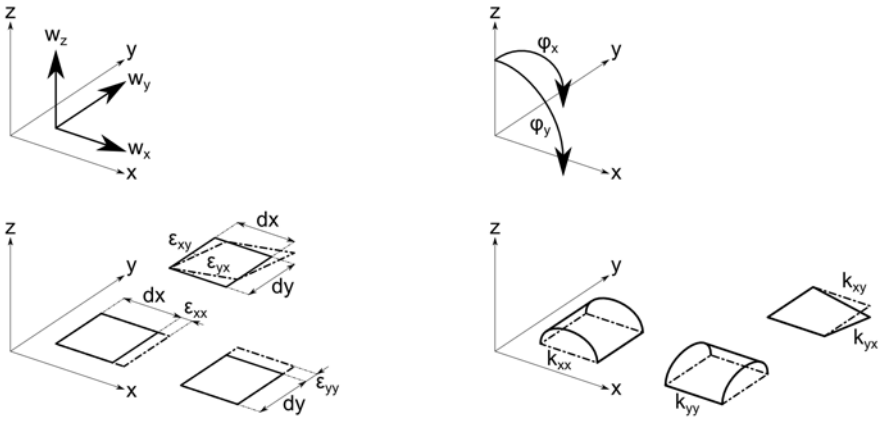


Figure H-4: Positive displacements (w), strain (ε), rotation (φ) and curvature (k) for surfaces.

Rotations are a change of location measured at a distance perpendicular to the displacement. Rotations indicated with the angle φ;

$$\varphi_x = -\frac{\partial w_z}{\partial y} \tag{H-9}$$

$$\varphi_y = -\frac{\partial w_z}{\partial x} \tag{H-10}$$

$$\tag{H-11}$$

Strain is the change in length of a surface where l_0 is the original length and l the length after straining. The strain is indicated with ε;

$$\varepsilon = \frac{l - l_0}{l_0}$$

$$\varepsilon_{xx} = \frac{\partial w_x}{\partial x} \tag{H-12}$$

$$\varepsilon_{yy} = \frac{\partial w_y}{\partial y} \tag{H-13}$$

$$\varepsilon_{xy} = \frac{\partial w_x}{\partial y} \tag{H-14}$$

$$\varepsilon_{yx} = \frac{\partial w_y}{\partial x} \tag{H-15}$$

$$\tag{H-16}$$

The shear deformation is the deformation that deforms the surface into a diamond shape and is expressed as the angle γ. The total shear is the sum of contributions in x- and y-direction;

$$\gamma_{xy} = \frac{\partial w_x}{dy} + \frac{\partial w_y}{dx}$$

Curvature is indicated with k. Curvature is the change of rotation.

$$k_{xx} = -\frac{\partial \varphi_x}{\partial x} \tag{H-17}$$

$$k_{yy} = -\frac{\partial \varphi_y}{\partial y} \tag{H-18}$$

$$\tag{H-19}$$

Twisting deformation changes the surface in to the shape of a hyperbolic paraboloid. Twisting is defined as the change of angle perpendicular to its direction. There are two contributions to twisting.

$$k_{xy} = \frac{\partial \varphi_x}{\partial y} \tag{H-20}$$

$$k_{yx} = -\frac{\partial \varphi_y}{\partial x} \tag{H-21}$$

The total twist is the sum of twist of contributions in x- and y-direction. This is indicated with ρ.

$$\rho_{xy} = \frac{\partial \varphi_x}{\partial y} - \frac{\partial \varphi_y}{\partial x} \tag{H-22}$$

The principle directions of the curvature are the directions in which there is no twist. The curvature is a second order tensor. Therefore the principle directions can be determined by the circle of Mohr.

$$\begin{aligned} k_{1,2} &= \frac{k_{xx} + k_{yy}}{2} \pm \sqrt{\left(\frac{k_{xx} - k_{yy}}{2}\right)^2 + k_{xy}^2} \\ &= A \pm B \,(\text{for later use}) \end{aligned} \tag{H-23}$$

The directions of the tangent planes can be found with;

$$\alpha_1 = \frac{1}{2} \arctan\left(\frac{2k_{xy}}{k_{xx} - k_{yy}}\right) \tag{H-24}$$

$$\alpha_2 = \frac{1}{2} \arctan\left(\frac{2k_{xy}}{k_{xx} - k_{yy}}\right) + \frac{1}{2}\pi \tag{H-25}$$

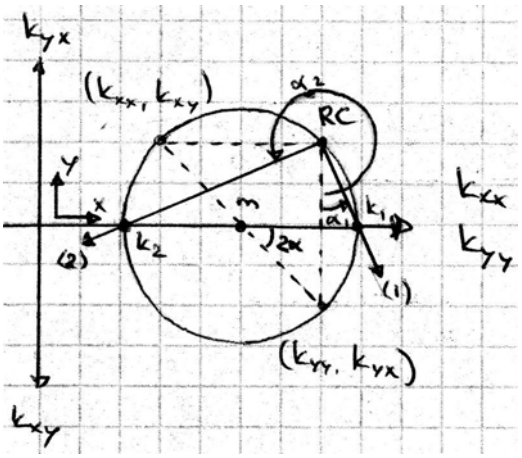


Figure H-5: Circle of Mohr for curvature

H-4 Gaussian curvature

Carl Friedrich Gauss (1777-1855) was a German scientist who became famous for his work in mathematics. In his paper 'General investigations of curved surfaces' he described extensively the definition of the measure of curvature Gauss [1827] (translated version Gauss et al. [1902]). Now known as Gaussian curvature. Essential, but often not mentioned, is the notion that surfaces can be described , simultaneously, by intrinsic and extrinsic properties features. Intrinsic properties depend only on measurements of length in two dimensions. Extrinsic properties depend measurements in three-dimensional space. Gaussian curvature is the mathematical connection between these two descriptions Calladine [1983].

$$\begin{aligned} G &= k_1 \cdot k_2 \\ &= (A+B)\cdot(A-B) \text{ (note: A and B from H-23)} \\ &= A^2 - B^2 \\ &= \left(\frac{k_{xx}+k_{yy}}{2}\right)^2 - \left[\left(\frac{k_{xx}-k_{yy}}{2}\right)^2 + k_{xy}^2\right] \\ &= k_{xx}\cdot k_{yy} - k_{xy}^2 \end{aligned} \tag{H-26}$$

Substitution of Equation H-17, H-18 and H-20 in Equation H-26 results in;

$$G = \frac{\partial^2 u_z}{\partial x^2}\cdot\frac{\partial^2 u_z}{\partial y^2} - \left(\frac{\partial^2 u_z}{\partial x \partial y}\right)^2 \tag{H-27}$$

This definition of Gaussian curvature is the best known definition and is described in many books, but there are more definitions. And they can be very useful according to Calladine [1983] when analyzing shell structures. The definition in Equation H-27 is described in three dimension in Euclidean space. Here different definitions of curvature of lines and surfaces will be given to illustrate what curvature represents;

$$k = \frac{\text{angle of embrace}}{\text{length of arc}} \quad \text{(H-28)}$$

$$k = \frac{1}{\text{radius of curvature}} \quad \text{(H-29)}$$

$$k = \frac{\text{arc length cd of circular image}}{\text{arc length CD of curve}} \quad \text{(H-30)}$$

$$k_G = \frac{\text{area of spherical image}}{\text{area region of surface}} \quad \text{(H-31)}$$

$$k_G = \frac{1}{R^1}\frac{1}{R^2} \quad \text{(H-32)}$$

$$k_G = \frac{\text{angular defect at a vertex}}{\text{area associated with the vertex}} \quad \text{(H-33)}$$

Appendix I

Strain and change of Gaussian curvature (In development)

An equation will be derived that describes the relation between straining of a surface due to a change in the Gaussian curvature. First basic equations will be described then the straining of respectively a bar and a surface in second order theory will de derived. Finally the relation between strain and Gaussian curvature is derived based on compatibility equations.

I-1 First and second order theory

Theory of mechanics makes use of mathematical models to describe in this case the strain. To derive these mathematical models assumptions are made. Commonly used is the first order theory. Here it is assumed that strain in a curtain direction is only due displacements in that direction. Displacements in other directions are considered very small and are therefore neglected. However when describing kinematic properties of surfaces it can be shown that assumptions of first order theory are insufficient.

The name second order theory does not refer to displacements in the directions of the bar or surface. Like in the first order theory these terms are considered to be very small and are therefore neglected. The essential difference compared to thirst order theory is an additional contribution to strain of displacement perpendicular to the bar or surface.

I-2 Strain of a bar

Here the strain of a bar is derived. First order theory takes into account displacements u in x-direction as in Figure I-1. Theory and figures are adapted from [Blaauwendraad and Kok, 1973, chap. 4.3].

According to the second order theory a second displacement w perpendicular to the bar is also considered. This can be seen in Figure I-2. In this case the total displacements u and w

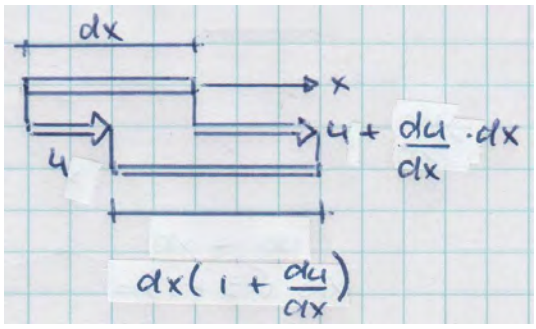


Figure I-1: Strain of a bar in first order theory

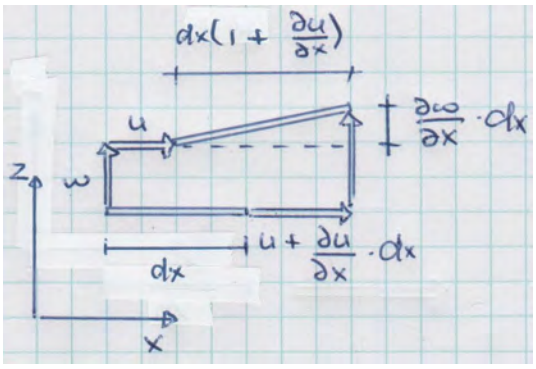


Figure I-2: Strain of a bar in second order theory

in x- and z-direction of the bar are described by two functions of the separate displacements $\overline{u}$ and $\overline{w}$;

$$u(x,z) = \overline{u}(x) - y\frac{d\overline{w}(x)}{dx} \tag{I-1}$$

$$w(x,z) = \overline{w}(x) \tag{I-2}$$

The strain is an measure of extension expressed as the ratio of extension and initial length;

$$\varepsilon = \frac{l - l_0}{l_0} \tag{I-3}$$

The initial length l_0 and length after deformation l of the bar in Figure I-2 can be expressed as;

$$l_0 = dx \tag{I-4}$$

$$l = dx\sqrt{\left(1 + \frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial w}{\partial x}\right)^2} \tag{I-5}$$

Substituting Equation I-4 and I-5 into Equation I-3 results in;

$$\begin{aligned} \varepsilon &= \frac{l - l_0}{l_0} \\ &= \frac{dx\sqrt{\left(1 + \frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial w}{\partial x}\right)^2} - dx}{dx} \\ &= \sqrt{\left(1 + \frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial w}{\partial x}\right)^2} - 1 \end{aligned} \tag{I-6}$$

The root in this function makes evaluating a laborious job. Therefore I-6 is estimated using a second order Taylor expansion of two variables. Higher order Taylor expansions include higher order derivatives of the original functions. These are assumed to be very small and are therefore not taken onto account. The result of the Taylor expansion is;

$$\varepsilon = \frac{\partial u}{\partial x} + \frac{1}{2}\left(\frac{\partial w}{\partial x}\right)^2 \tag{I-7}$$

Now substitution of Equation I-1 and I-2 in I-7 results in;

$$\varepsilon(x,z) = \frac{d\overline{u}}{dx} + \frac{1}{2}\left(\frac{d\overline{w}}{dx}\right)^2 - z\frac{d^2\overline{w}}{dx^2} \tag{I-8}$$

The last part of Equation I-8 described the strain over the thickness of the cross section. In this part $\frac{d^2\overline{w}}{dx^2}$ can be recognized as the curvature of the bar;

$$\kappa = -\frac{d^2\overline{w}}{dx^2} \tag{I-9}$$

For a cross section the first two terms are constant and therefore the last term becomes zero. The final expression of the strain is now;

$$\varepsilon_{xx} = \frac{d\overline{u}}{dx} + \frac{1}{2}\left(\frac{d\overline{w}}{dx}\right)^2 \tag{I-10}$$

Here the difference compared to the first order theory becomes clear. In first order theory the expression of the strain is $\varepsilon_{xx} = \frac{d\overline{u}}{dx}$. By deriving this equation it shown how the additional considered displacement in second order theory influences the strain.

I-3 Strain of a surface

To describe the strain of a surface a fiber in curtain direction is imagined. Further derivations will consider this fiber as a bar, but now an additional displacement is added. In Figure I-4 the total displacements u, v and w of the fiber in x-direction can be described by three functions of the separate displacements $\overline{u}$, $\overline{v}$ and $\overline{w}$;

$$u(x,y,z) = \overline{u}(x,y,z) - z\frac{\partial \overline{w}(x,y)}{\partial x} \tag{I-11}$$

$$v(x,y,z) = \overline{v}(x,y,z) - z\frac{\partial \overline{w}(x,y)}{\partial y} \tag{I-12}$$

$$z(x,y,z) = \overline{w}(x,y,z) \tag{I-13}$$

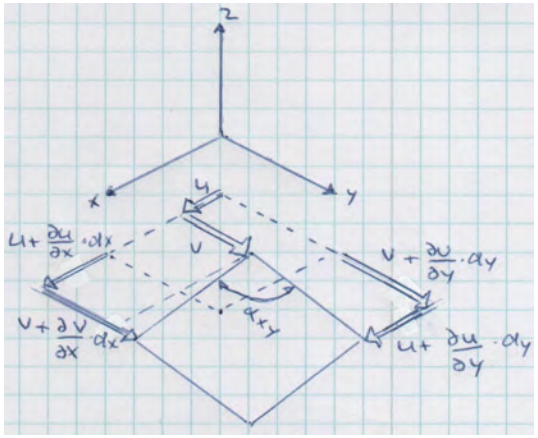


Figure I-3: Strain of a surface in first order theory

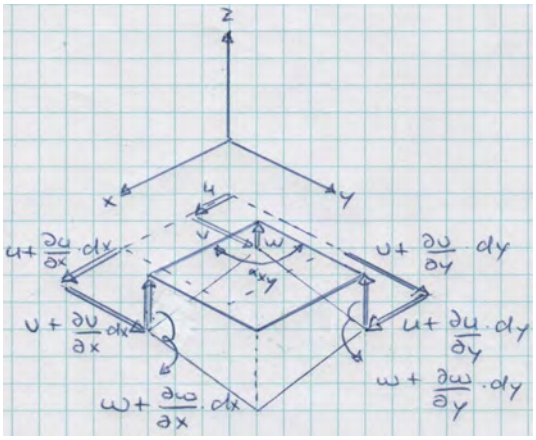


Figure I-4: Strain of a surface in second order theory

To determine the strain in x-direction the length of a piece dx after displacement and deformation is described as;

$$l = dx\sqrt{\left(1+\frac{\partial u}{\partial x}\right)^2+\left(\frac{\partial v}{\partial x}\right)^2+\left(\frac{\partial w}{\partial x}\right)^2} \tag{I-14}$$

Using I-3 and I-4 the expression for the strain in x-direction becomes;

$$\varepsilon_{xx} = \sqrt{\left(1+\frac{\partial u}{\partial x}\right)^2+\left(\frac{\partial v}{\partial x}\right)^2+\left(\frac{\partial w}{\partial x}\right)^2}-1 \tag{I-15}$$

This function can be estimated using a second order Taylor expansion of three variables. This results in;

$$\varepsilon_{xx} = \frac{\partial u}{\partial x}+\frac{1}{2}\left(\frac{\partial v}{\partial x}\right)^2+\frac{1}{2}\left(\frac{\partial w}{\partial x}\right)^2 \tag{I-16}$$

Substituting I-11, I-12 and I-13 results in;

$$\varepsilon_{xx} = \frac{d\overline{u}}{dx}+\frac{1}{2}\left(\frac{d\overline{v}}{dx}\right)^2+\frac{1}{2}\left(\frac{d\overline{w}}{dx}\right)^2-z\frac{d^2\overline{w}}{dx^2} \tag{I-17}$$

In a similar way the strain in y-direction can be determined. This results in;

$$\varepsilon_{yy} = \frac{d\overline{v}}{dy}+\frac{1}{2}\left(\frac{d\overline{u}}{dy}\right)^2+\frac{1}{2}\left(\frac{d\overline{w}}{dy}\right)^2-z\frac{d^2\overline{w}}{dy^2} \tag{I-18}$$

In Equation I-17 and I-18 the last part can be recognized as the curvature in x- and y-direction. This part of the equation takes into account the bending of the surface.

$$\kappa_{xx} = -\frac{d^2\overline{w}}{dx^2} \tag{I-19}$$

$$\kappa_{yy} = -\frac{d^2\overline{w}}{dy^2} \tag{I-20}$$

Assumptions of the second order theory allow to neglect the 2nd term in I-17 and I-18. The first and third terms are constant for a cross section and therefore the last term becomes zero. The equations become;

$$\varepsilon_{xx} = \frac{d\overline{u}}{dx} + \frac{1}{2}\left(\frac{d\overline{w}}{dx}\right)^2 \tag{I-21}$$

$$\varepsilon_{yy} = \frac{d\overline{v}}{dy} + \frac{1}{2}\left(\frac{d\overline{w}}{dy}\right)^2 \tag{I-22}$$

[γ_{xy} can be derived according to Blaauwendraad or Timoshenco. I have not yet been able to derived the equation for $cos\alpha_{xy}$ that Blaauwendraad uses. He used cos where other mathematic books use sin]

$$\gamma_{xy} = \frac{d\overline{u}}{dy} + \frac{d\overline{v}}{dx} + \frac{d\overline{w}}{dx}\frac{d\overline{w}}{dy} \tag{I-23}$$

I-4 Gaussian curvature

The functions for the strain derived in the previous section cannot be arbitrary functions when they describe the strain in the surface after deformation. The surface is assumed to remain intact after deformation. It is possible to find a combination of strain functions that do not satisfy this assumption. Compatibility equations are the boundary conditions that require the strain functions to satisfy the assumption. This is illustrated with an example of a flat surface where the strain function are chosen in such a way the compatibility equations are not met.

If the strain function are described by; $\varepsilon_{xx} = \varepsilon_{yy} = 0$ and $\gamma_{xy} = cxy$. Where c is a constant. The corresponding deformation is in Figure I-5. It is clear that the surface has not remained intact and the compatibility requirements are therefore not met. The surface in this example is required to remain in plane. But it can also be seen that if the surface is allowed to displace in perpendicular direction it would be possible to create a deformed curved surface. From that it becomes clear that the curvature has changed due to straining.

For first order theory the compatibility equation can be found by differentiating twice the first terms of Equation I-21, I-22 and I-23 [Koiter, 1985, chap. 15]. By choosing a linear combination of these derivative that eliminate their contributions the following compatibility equation appears;

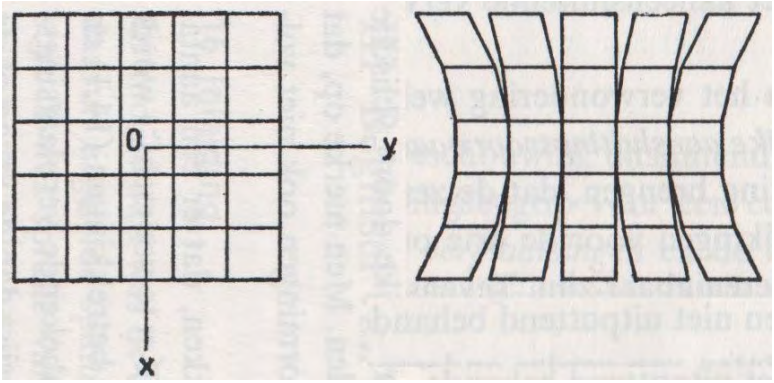


Figure I-5: Combination of strain functions that does not meet the compatibility requirements.

$$\begin{aligned}
\frac{\partial^2 \varepsilon_{xx}}{\partial y^2} &= \frac{\partial^3 \overline{u}}{\partial x \partial y^2} \\
\frac{\partial^2 \varepsilon_{y}}{\partial x^2} &= \frac{\partial^3 \overline{v}}{\partial y \partial x^2} \\
\frac{\partial^2 \gamma_{xy}}{\partial x \partial y} &= \frac{\partial^3 \overline{u}}{\partial x \partial y^2} + \frac{\partial^3 \overline{v}}{\partial x^2 \partial y} \\
-\frac{\partial^2 \varepsilon_{xx}}{\partial y^2} - \frac{\partial^2 \varepsilon_{y}}{\partial x^2} + \frac{\partial^2 \gamma_{xy}}{\partial x \partial y} &= 0
\end{aligned} \tag{I-24}$$

For second order theory the derivation of the compatibility equations can be derived in the same way [[Calladine, 1983, chap. 6]]. But it is more extensive since all terms in Equation I-21, I-22 and I-23 must be taken into account. The following derivatives are determined;

The first and second derivative of ε_{xx} to y are;

$$\begin{aligned}
\varepsilon_{xx} &= \frac{\partial u}{\partial x} + \frac{1}{2}\left(\frac{\partial z}{\partial x}\right)^2 \\
\frac{\partial \varepsilon_{xx}}{\partial y} &= \frac{\partial^2 u}{\partial x \partial y} + \frac{\partial z}{\partial x}\frac{\partial^2 z}{\partial x \partial y} \\
\frac{\partial^2 \varepsilon_{xx}}{\partial y^2} &= \frac{\partial^3 u}{\partial x \partial y^2} + \frac{\partial z}{\partial x}\frac{\partial^3 z}{\partial x \partial y^2} + \left(\frac{\partial^2 z}{\partial x \partial y}\right)^2 \\
&= C + D + E
\end{aligned} \tag{I-25}$$

The derivatives of ε_{yy} to x are;

$$\begin{aligned}
\varepsilon_{yy} &= \frac{\partial v}{\partial y} + \frac{1}{2}\left(\frac{\partial z}{\partial y}\right)^2 \\
\frac{\partial \varepsilon_{yy}}{\partial x} &= \frac{\partial^2 v}{\partial x \partial y} + \frac{\partial z}{\partial y}\frac{\partial^2 z}{\partial x \partial y} \\
\frac{\partial^2 \varepsilon_{yy}}{\partial x^2} &= \frac{\partial^3 v}{\partial x^2 \partial y} + \frac{\partial z}{\partial y}\frac{\partial^3 z}{\partial x^2 \partial y} + \left(\frac{\partial^2 z}{\partial x \partial y}\right)^2 \\
&= F + G + E
\end{aligned} \tag{I-26}$$

The derivatives of γ_{xy} to x and then to y are;

$$\begin{aligned}
\gamma_{xy} &= \frac{\partial u}{\partial y} + \frac{\partial v}{\partial x} + \frac{\partial z}{\partial x}\frac{\partial z}{\partial y} \\
\frac{\partial \gamma_{xy}}{\partial x} &= \frac{\partial^2 u}{\partial x \partial y} + \frac{\partial^2 v}{\partial x^2} + \frac{\partial z}{\partial x}\frac{\partial^2 z}{\partial x \partial y} + \frac{\partial^2 z}{\partial x^2}\frac{\partial z}{\partial y} \\
\frac{\partial^2 \gamma_{xy}}{\partial x \partial y} &= \frac{\partial^3 u}{\partial x \partial y^2} + \frac{\partial^3 v}{\partial x^2 \partial y} + \frac{\partial z}{\partial x}\frac{\partial^3 z}{\partial x \partial y^2} + \left(\frac{\partial^2 z}{\partial x \partial y}\right)^2 + \frac{\partial^2 z}{\partial x^2}\frac{\partial^2 z}{\partial y^2} + \frac{\partial^3 z}{\partial x^2 \partial y}\frac{\partial z}{\partial y} \\
&= C + F + D + E + H + G
\end{aligned} \tag{I-27}$$

If now a linear combination is taken as in Equation I-24 if follows that;

$$\begin{aligned}
-\frac{\partial^2 \varepsilon_{xx}}{\partial y^2} - \frac{\partial^2 \varepsilon_{yy}}{\partial x^2} + \frac{\partial^2 \gamma_{xy}}{\partial x \partial y} &= \\
-(C + D + E) - (F + G + E) + (C + F + D + E + H + G) &= \\
H - E &= \frac{\partial^2 z}{\partial x^2}\frac{\partial^2 z}{\partial y^2} - \left(\frac{\partial^2 z}{\partial x \partial y}\right)^2
\end{aligned} \tag{I-28}$$

This last expression can be recognized as the Gaussian curvature. Now the compatibility equation in second order theory can be expressed. This equation describes the relation between straining of a surface and the change of Gaussian curvature;

$$-\frac{\partial^2 \varepsilon_{xx}}{\partial y^2} - \frac{\partial^2 \varepsilon_{yy}}{\partial x^2} + \frac{\partial^2 \gamma_{xy}}{\partial x \partial y} = k_G \tag{I-29}$$

Appendix J

Visual Studio C# code for parametric Grasshopper component (In development)

This code was written the programming language C#. The code was written for a Grasshopper components that provide data for calculations in Matlab. The component is shown in Figure J-1.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using Grasshopper.Kernel;
using Rhino.Geometry;
using System.Drawing;
using FlexMouldComponents.Properties;

namespace FlexMouldComponents
{
    //------------------------------------------------------------------------
    //Create support points from surface component
    //Develloped by Peter Eigenraam
    //20130328
    //20130410 Fixed index bug for creating matlab data of support points.
    //20130413 Changed input and output data. Removed EI. Added E, NS and T.
    //20130418 Changed units of loading from N/mm^2 to N/m^2.
    //------------------------------------------------------------------------

    //------------------------------------------------
    //SETUP OF CODE:
    //Create parameters for geometry.
    //Create parameters for calculations.
    //Create parameters for results.
    //Create parameters for presenting in Rhino.
    //Constructor.
    //Register input- and output parameters.
    //Clear lists.
    //Retreive input parameters.
    //Add data to list of Matlab data.
    //Create point for supports.
    //Create points of shape.
    //Assign data to output variables.
    //Create custom preview for presentation.
    //Methods for retreiving data from input parameters.
    //Manage Guid.
    //Manage icon.
    //------------------------------------------------

    public class CreateSupportPointsSurfaceComponent : GH_Component
    {
        #region Create parameters for geometry.

        //Create parameters for geometry.
        Rhino.Geometry.Surface geo_s = null;

        //Create list for grid points for supports.
        //List<Rhino.Geometry.Point3d> geo_grid = new List<Rhino.Geometry.Point3d>();

        #endregion

        #region Create parameters for calculations.

        //Create parameters for calculation.
        int cal_sx = new int(); //Number of supports in x-direction.
        int cal_sy = new int(); //Number of supports in y_direction
        double cal_d = new double(); //Distance between supports.
        double cal_q = new double(); //Distance between supports.

        #endregion

        #region Create parameters for results.

        //Create list of points for the supports.
        List<Rhino.Geometry.Point3d> res_supp = new List<Rhino.Geometry.Point3d>();

        //Create parameters for calculation.
        int res_sx = new int(); //Number of supports in x-direction.
        int res_sy = new int(); //Number of supports in y-direction.
        double res_d = new double(); //Distance between supports.
        double res_q = new double(); //Distributed load.
        double res_e = new double(); //Youngs modulus of material of strips.
        int res_n = new int(); //Number of strips between supports.
        double res_t = new double(); //Thickness of strips.
```

```
    double res_l1 = new double(); //Length of top part of support.
    double res_l2 = new double(); //Length of top part of support.

    #endregion

    #region Create parameters for presenting in Rhino.

    //Create list of messages and matlab data.
    //Clear list at the start SolveInstance.
    List<string> pre_messages = new List<string>();
    List<string> pre_matlab = new List<string>();

    #endregion

    #region Constructor.

    public CreateSupportPointsSurfaceComponent()
        : base("Support points and matlab data from surface", "Supports", "This component creates support points and matlab data from a surface. Draw a surface and place it near the origin of the coordinate system on the positive sides of the X- and Y-direction. (v20130418)", "FlexMould", "Multiple supports")
    { }

    #endregion

    #region Register input- and output parameters.

    protected override void RegisterInputParams(GH_Component.GH_InputParamManager pManager)
    {
        pManager.AddSurfaceParameter("Surface", "S", "Surface of the intended shape.",GH_ParamAccess.item);
        pManager.AddIntegerParameter("Supports X-direction", "Sx", "Number of supports of the strip in X-direction.",GH_ParamAccess.item);
        pManager.AddIntegerParameter("Supports Y-direction", "Sy", "Number of supports of the strip in Y-direction.", GH_ParamAccess.item);
        pManager.AddNumberParameter("Distance", "D", "Distance between supports [mm].", GH_ParamAccess.item);
        pManager.AddNumberParameter("Load", "q", "Equally distributed force [N/m^2].", GH_ParamAccess.item);
        pManager.AddNumberParameter("Youngs modulus", "E", "Youngs modulus of material of strips [N/mm^2].", GH_ParamAccess.item);
        pManager.AddIntegerParameter("Strips", "N", "Number of strips between supports [-].", GH_ParamAccess.item);
        pManager.AddNumberParameter("Thickness", "T", "Thickness of strips [mm].", GH_ParamAccess.item);
        pManager.AddNumberParameter("Length", "L1", "Length of top part of support [mm].", GH_ParamAccess.item);
        pManager.AddNumberParameter("Length", "L2", "Length of bottom part of support [mm].", GH_ParamAccess.item);
    }

    protected override void RegisterOutputParams(GH_Component.GH_OutputParamManager pManager)
    {
        pManager.AddTextParameter("Messages", "M", "Messages.",GH_ParamAccess.list);
        pManager.AddTextParameter("Matlab", "Matlab", "Input for calculations in Matlab.", GH_ParamAccess.list);
    }

    #endregion

    protected override void SolveInstance(IGH_DataAccess DA)
    {
        #region Clear lists.

        //Clear list of support points.
        res_supp.Clear();

        //Clear list of messages.
        pre_messages.Clear();

        //Clear list of matlab data.
        pre_matlab.Clear();

        #endregion

        #region Retreive input parameters.
```

```
//Retreive geometry.
if (!DA.GetData(0, ref geo_s))
{
    return;
}

//Retreive integer for number of supports in X- and Y-direction.
RetreiveInteger(DA, 1, ref cal_sx);
RetreiveInteger(DA, 2, ref cal_sy);

//Retreive double for support distance.
RetrieveDouble(DA, 3, ref cal_d);

//Retreive double for load.
RetrieveDouble(DA, 4, ref cal_q);

//Retreive double Youns modulus.
RetrieveDouble(DA, 5, ref res_e);

//Retreive integer number of strips between supports.
RetreiveInteger(DA, 6, ref res_n);

//Retreive double thickness of strips.
RetrieveDouble(DA, 7, ref res_t);

//Retreive double for length of top part of support.
RetrieveDouble(DA, 8, ref res_l1);

//Retreive double for length of bottom part of support.
RetrieveDouble(DA, 9, ref res_l2);

#endregion

#region Add data to list of Matlab data.

//Number of supports in x-direction.
res_sx = cal_sx;
pre_matlab.Add(res_sx.ToString());

//Number of supports in y-direction.
res_sy = cal_sy;
pre_matlab.Add(res_sy.ToString());

//Support distance.
res_d = cal_d;
pre_matlab.Add(res_d.ToString());

//Loading.
res_q = cal_q / Math.Pow(10, 6);
pre_matlab.Add(res_q.ToString());

//Youns mudulus of material of strips.
pre_matlab.Add(res_e.ToString());

//Number of strips between supports.
pre_matlab.Add(res_n.ToString());

//Thickness of strips.
pre_matlab.Add(res_t.ToString());

//Length of top part of support.
pre_matlab.Add(res_l1.ToString());

//Length of top bottom of support.
pre_matlab.Add(res_l2.ToString());

#endregion

#region Create points for supports.

//Add points to list of support points.
for (int i = 0; i < cal_sy; i++)
{
    for (int j = 0; j < cal_sx; j++)
    {
```

```
                //Create curve to intersect with surface.
                Rhino.Geometry.Point3d i_point_1 = new Rhino.Geometry.Point3d((double)j * cal_d, (double)↙
i * cal_d, 0);
                Rhino.Geometry.Point3d i_point_2 = new Rhino.Geometry.Point3d((double)j * cal_d, (double)↙
i * cal_d, geo_s.GetBoundingBox(true).Max.Z + (double)1);
                Rhino.Geometry.Curve i_curve = new Rhino.Geometry.Line(i_point_1,i_point_2).ToNurbsCurve ↙
();

                //Find intersection events of curve and surface.
                var intersection = Rhino.Geometry.Intersect.Intersection.CurveSurface(i_curve,geo_s,0.001↙
,0.0001);

                //Add support points to list.
                res_supp.Add(intersection[0].PointA);
            }
        }

        //Create messages and matlab data.
        pre_messages.Add("There are " + res_supp.Count.ToString() + " supports.");
        pre_messages.Add(" ");

        for (int i = 0; i < cal_sy; i++)
        {
            for (int j = 0; j < cal_sx; j++)
            {
                int supp_nr_x = j + 1;
                int supp_nr_y = i + 1;
                int index = i * cal_sx + j;
                pre_messages.Add("P[" + supp_nr_x.ToString() + "," + supp_nr_y.ToString() + "] = (" +    ↙
String.Format("{0:F2}", res_supp[index].X) + ", " + String.Format("{0:F2}", res_supp[index].Y) + ", " +  ↙
String.Format("{0:F2}", res_supp[index].Z) + ") [mm].");
                pre_matlab.Add(String.Format("{0:F2}", res_supp[index].X) + " " + String.Format("{0:F2}",↙
 res_supp[index].Y) + " " + String.Format("{0:F2}", res_supp[index].Z));
            }
        }

        #endregion

        #region Create points of shape.

        //Add points of shape above strips in x-direction to list of matlab data.
        for (int i = 0; i < cal_sy; i++)
        {
            for (int j = 0; j <= (cal_sx - 1) * 10; j++)
            {
                //Create curve to intersect with surface.
                Rhino.Geometry.Point3d i_point_1 = new Rhino.Geometry.Point3d((double)j * cal_d / 10,    ↙
(double)i * cal_d, 0);
                Rhino.Geometry.Point3d i_point_2 = new Rhino.Geometry.Point3d((double)j * cal_d / 10,    ↙
(double)i * cal_d, geo_s.GetBoundingBox(true).Max.Z + (double)1);
                Rhino.Geometry.Curve i_curve = new Rhino.Geometry.Line(i_point_1, i_point_2).ToNurbsCurve↙
();

                //Find intersection events of curve and surface.
                var intersection = Rhino.Geometry.Intersect.Intersection.CurveSurface(i_curve, geo_s, 0. ↙
001, 0.0001);

                //Add support points to lists.
                pre_matlab.Add(String.Format("{0:F2}", intersection[0].PointA.X) + " " + String.Format(" ↙
{0:F2}", intersection[0].PointA.Y) + " " + String.Format("{0:F2}", intersection[0].PointA.Z));
            }
        }

        //Add points of shape above strips in y-direction to list of matlab data.
        for (int i = 0; i < cal_sx; i++)
        {
            for (int j = 0; j <= (cal_sy - 1) * 10; j++)
            {
                //Create curve to intersect with surface.
                Rhino.Geometry.Point3d i_point_1 = new Rhino.Geometry.Point3d((double)i * cal_d, (double)↙
j * cal_d / 10, 0);
                Rhino.Geometry.Point3d i_point_2 = new Rhino.Geometry.Point3d((double)i * cal_d, (double)↙
j * cal_d / 10, geo_s.GetBoundingBox(true).Max.Z + (double)1);
                Rhino.Geometry.Curve i_curve = new Rhino.Geometry.Line(i_point_1, i_point_2).ToNurbsCurve↙
();
```

```
                //Find intersection events of curve and surface.
                var intersection = Rhino.Geometry.Intersect.Intersection.CurveSurface(i_curve, geo_s, 0. ↙
001, 0.0001);

                //Add support points to lists.
                pre_matlab.Add(String.Format("{0:F2}", intersection[0].PointA.X) + " " + String.Format(" ↙
{0:F2}", intersection[0].PointA.Y) + " " + String.Format("{0:F2}", intersection[0].PointA.Z));
            }
        }

        #endregion

        #region Assign data to output variables.

        //Assign messages to output parameter.
        DA.SetDataList(0, pre_messages);

        //Assign matlab data to output parameter.
        DA.SetDataList(1, pre_matlab);

        #endregion
    }

    #region Create custom preview for presentation

    public override void DrawViewportWires(IGH_PreviewArgs args)
    {
        //Create preview of support points. Call the base for drawviewport.
        base.DrawViewportWires(args);

        //Display support points.
        args.Display.DrawPoints(res_supp, Rhino.Display.PointStyle.X, 10, Color.Black);
    }

    #endregion

    #region Methods for retreiving data from input parameters.

    /// <summary>
    /// Retreive double from input of component.
    /// </summary>
    /// <param name="DA">Use "DA".</param>
    /// <param name="i">Index of input parameter (zero based).</param>
    /// <param name="d">Double to store data in.</param>
    private static void RetrieveDouble(IGH_DataAccess DA, int i, ref double result)
    {
        if (!DA.GetData(i, ref result))
        {
            return;
        }

        if (!Rhino.RhinoMath.IsValidDouble(result))
        {
            return;
        }
    }

    /// <summary>
    /// Retreive integer from input of component.
    /// </summary>
    /// <param name="DA">Use "DA"</param>
    /// <param name="i">Index of input parameter (zero based).</param>
    /// <param name="result">Integer to store data in.</param>
    private static void RetreiveInteger(IGH_DataAccess DA, int i, ref int result)
    {
        if (!DA.GetData(i, ref result))
        {
            return;
        }
    }

    #endregion

    #region Manage Guid.
```

```
        public override Guid ComponentGuid
        {
            get
            {
                return new Guid("{96A2E018-2DF2-46ab-AC0F-3633BEDDF660}");
            }
        }

        #endregion

        #region Manage icon.

        protected override Bitmap Icon
        {
            get
            {
                return Resources.CreateSupportPointsSurface;
            }
        }

        #endregion
    }

}
```

Figure J-1: Custom Grasshopper component determines the initial support heights according to the distance between the support. The component also provides other information for structural calculation. This information can be read by other software. For example Matlab.

Appendix K

Parametric Matlab script (In development)

```
%% Calculation of strip displacements and reaction forces of the flexible
    mould.
% Peter Eigenraam March-April 2013
% Import support data file from Grasshopper

%% Outline of calculations
% This calculation script consists of three parts:
%   1. Importing data
%   2. Structural and kinematic calculations
%   3. Presenting results

% Clear
clc;clear;close all;
format long

% Enter filename.
filename = 'data.txt';

%% PART 1 IMPORTING DATA
% Calculations start here.

% Read file data.
data = textscan(fopen(filename), '%f %f %f');
fclose(fopen(filename));

% Set variables.
sx = data{1,1}(1); % Number of supports in x-direction.
sy = data{1,1}(2); % Number of supports in y-direction.
d = data{1,1}(3); % Distance between supports.
q = data{1,1}(4);% Loading [N/mm^2].
e = data{1,1}(5); % Youngs mudulus [N/mm^2]
```

```
31 n = data{1,1}(6); % Number of strips between supports [-].
32 t = data{1,1}(7); % Thickness of strip [mm].
33 l1 = data{1,1}(8); % Length of top part of support [mm].
34 l2 = data{1,1}(9); % Length of bottom part of support [mm].
35 ni = 9; % Number of input variables.
36
37 % Check the number of supports in two directions. If not an odd integer
       create a warning.
38 % If number of supports in x-direction is an even number.
39 if mod(sx,2)==0
40     disp('WARNING:')
41     disp('Number of supports in x-direction is an even number. This should
           ')
42     disp('be odd since the middle support is assumed to be fixed.')
43        disp(' ')
44     break
45 end
46
47 % If number of supports in y-direction is an even number.
48 if mod(sy,2)==0
49     disp('WARNING:')
50     disp('Number of supports in y-direction is an even number. This should
           ')
51     disp('be odd since the middle support is assumed to be fixed.')
52     disp(' ')
53     break
54 end
55
56 % Total number of support points.
57 st = sx * sy;
58
59 % If q is zero replace with very small number.
60 if q==0
61      q = 1e-16;
62 end
63
64 % Width of strips.
65 w = d / n - 1;
66
67 % Reduction factor for width of strip due to notches.
68 rw = 0.5;
69
70 % Factor for increase in stiffness of strips in y-direction.
71 fsy = 2;
72
73 % Number of unknown constants. There are four constants for each strip/
       equation.
74 nc = ((sx - 1) * sy + (sy - 1) * sx) * 4;
75
76 % Number of unknown constants for strips in x- and y-direction.
77 ncx = sy * (sx - 1) * 4;
78 ncy = sx * (sy - 1) * 4;
79
```

```
80  % Number of devision to visualize deflection.
81  dev = 10;
82
83  % Number of data points in x- and y-direction for shape.
84  ndx = sy * ((sx - 1) * dev + 1);
85  ndy = sx * ((sy - 1) * dev + 1);
86
87  % Explain which calculations are done:
88  disp('CALCULATIONS:');
89  disp('For the flexible mould the deflection of the strips in two
        directions');
90  disp('is calculated. Only strips above supports are considered and to be'
        );
91  disp('seperate so there is no interaction.');
92  %disp(' ');
93  %disp('ASSUMPTIONS:');
94  %disp(' - Only single bending occurs');
95  %disp(' - Small deflections');
96  disp(' ');
97  disp('!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!');
98  disp('!! Note that the assumptions are not fullfilled !!');
99  disp('!! Calculations are indicative                  !!');
100 disp('!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!');
101 disp(' ');
102
103 % Display basic parameters.
104 disp('INPUT FOR CALCULATIONS:');
105 disp(['Number of supports in x-direction: ', num2str(sx), ' [-]']);
106 disp(['Number of supports in y-direction: ', num2str(sy), ' [-]']);
107 disp(['Distance between supports: ', num2str(d), ' [mm]']);
108 disp(['Loading of surface: ', num2str(q), ' [N/mm^2]']);
109 disp(['Youngs modulus: ', num2str(e), ' [N/mm^2]']);
110 disp(['Number of strips between supports: ', num2str(n), ' [-]']);
111 disp(['Width of strips: ', num2str(w), ' [mm]']);
112 disp(['Thickness of strips: ', num2str(t), ' [mm]']);
113 disp(['Length of top part of support: ', num2str(l1), ' [mm]']);
114 disp(['Length of bottom part of support: ', num2str(l2), ' [mm]']);
115 disp(['Number of unknowns to solve: ', num2str(nc), ' [-]']);
116 disp(['Number of devision per strips for plots: ', num2str(dev), ' [-]'])
        ;
117 disp(' ');
118
119 %% PART TWO Structural and kinematic calculations
120
121 % Fill matrices for system of equations for strips in x-direction.
122
123 % Create matrices to solve.
124 AX = zeros(ncx, ncx + 1);
125
126 % Create loop to fill the matrix for strips in x-direction.
127 for i = 1 : sy
128     if i == 1
129         % Bending stiffness of representative strip.
```

```
130         % Reduction is 0.5.
131         ei = e * 0.5 * n * (1/12) * rw * w * t^3;
132         %ei = 1; % Used for checking the script.
133
134         % First support.
135         % Set variables.
136         i_temp = 1 + (i - 1) * (sx - 1) * 4; % Index of first entry per
                loop.
137         var_temp = data{1,1}(ni + 1 + (i - 1) * sx); % x-coordinate of
                first support.
138
139         % Deflection is known.
140         AX(i_temp, i_temp) = var_temp^3;
141         AX(i_temp, i_temp + 1) = var_temp^2;
142         AX(i_temp, i_temp + 2) = var_temp;
143         AX(i_temp, i_temp + 3) = 1;
144
145         % Moment is known.
146         AX(i_temp + 1, i_temp) = -(ei * 6 * var_temp);
147         AX(i_temp + 1, i_temp + 1) = -(ei * 2);
148         AX(i_temp + 1, i_temp + 2) = 0;
149         AX(i_temp + 1, i_temp + 3) = 0;
150
151         % Middle supports.
152         for j = 1 : sx - 2
153             % Set variables.
154             i_temp = 3 + (i - 1) * (sx - 1) * 4 + (j - 1) * 4; % Index of
                    entry per loop.
155             var_temp = data{1,1}(ni + 1 + (i - 1) * sx + j); % x-
                    coordinate of support.
156
157             % Deflection is known.
158             AX(i_temp, i_temp - 2) = var_temp^3;
159             AX(i_temp, i_temp - 1) = var_temp^2;
160             AX(i_temp, i_temp) = var_temp;
161             AX(i_temp, i_temp + 1) = 1;
162             AX(i_temp, i_temp + 2) = 0;
163             AX(i_temp, i_temp  + 3) = 0;
164             AX(i_temp, i_temp + 4) = 0;
165             AX(i_temp, i_temp + 5) = 0;
166
167             % Deflection left and right of the support are equal.
168             AX(i_temp + 1, i_temp - 2)= var_temp^3;
169             AX(i_temp + 1, i_temp - 1) = var_temp^2;
170             AX(i_temp + 1, i_temp) = var_temp;
171             AX(i_temp + 1, i_temp + 1) = 1;
172             AX(i_temp + 1, i_temp + 2) = -(var_temp^3);
173             AX(i_temp + 1, i_temp + 3) = -(var_temp^2);
174             AX(i_temp + 1, i_temp + 4) = -(var_temp);
175             AX(i_temp + 1, i_temp + 5)= -1;
176
177             % Rotation left and right of the support are equal.
178             AX(i_temp + 2, i_temp - 2)= -(3 * var_temp^2);
```

```
179             AX(i_temp + 2, i_temp - 1) = -(2 * var_temp);
180             AX(i_temp + 2, i_temp) = -1;
181             AX(i_temp + 2, i_temp + 1) = 0;
182             AX(i_temp + 2, i_temp + 2) = 3 * var_temp^2;
183             AX(i_temp + 2, i_temp + 3) = 2* var_temp;
184             AX(i_temp + 2, i_temp + 4) = 1;
185             AX(i_temp + 2, i_temp + 5)= 0;
186
187             % Moment left and right of the support is equal.
188             AX(i_temp + 3, i_temp - 2)= -(ei * 6 * var_temp);
189             AX(i_temp + 3, i_temp - 1) = -(ei * 2);
190             AX(i_temp + 3, i_temp) = 0;
191             AX(i_temp + 3, i_temp + 1) = 0;
192             AX(i_temp + 3, i_temp + 2) = ei * 6 * var_temp;
193             AX(i_temp + 3, i_temp + 3) = ei * 2;
194             AX(i_temp + 3, i_temp + 4) = 0;
195             AX(i_temp + 3, i_temp + 5)= 0;
196         end
197
198         % Last support.
199         % Set variables.
200         i_temp = i * (sx - 1) * 4 - 1; % Index of first entry per loop.
201         var_temp = data{1,1}(ni + i * sx); % x-coordinate of last support
                .
202
203         % Deflection is known.
204         AX(i_temp, i_temp - 2) = var_temp^3;
205         AX(i_temp, i_temp - 1) = var_temp^2;
206         AX(i_temp, i_temp) = var_temp;
207         AX(i_temp, i_temp + 1) = 1;
208
209         % Moment is known.
210         AX(i_temp + 1, i_temp - 2) = -(ei * 6 * var_temp);
211         AX(i_temp + 1, i_temp - 1) = -(ei * 2);
212         AX(i_temp + 1, i_temp) = 0;
213         AX(i_temp + 1, i_temp + 1) = 0;
214
215     elseif i < sy
216         % Bending stiffness of representative strip.
217         ei = e * n * (1/12) * rw * w * t^3;
218         %ei = 1; % Used for checking the script.
219
220         % First support.
221         % Set variables.
222         i_temp = 1 + (i - 1) * (sx - 1) * 4; % Index of first entry per
                loop.
223         var_temp = data{1,1}(ni + 1 + (i - 1) * sx); % x-coordinate of
                first support.
224
225         % Deflection is known.
226         AX(i_temp, i_temp) = var_temp^3;
227         AX(i_temp, i_temp + 1) = var_temp^2;
228         AX(i_temp, i_temp + 2) = var_temp;
```

```
229         AX(i_temp, i_temp + 3) = 1;
230
231         % Moment is known.
232         AX(i_temp + 1, i_temp) = -(ei * 6 * var_temp);
233         AX(i_temp + 1, i_temp + 1) = -(ei * 2);
234         AX(i_temp + 1, i_temp + 2) = 0;
235         AX(i_temp + 1, i_temp + 3) = 0;
236
237         % Middle supports.
238         for j = 1 : sx - 2
239             % Set variables.
240             i_temp = 3 + (i - 1) * (sx - 1) * 4 + (j - 1) * 4; % Index of
                    entry per loop.
241             var_temp = data{1,1}(ni + 1 + (i - 1) * sx + j); % x-
                   coordinate of support.
242
243             % Deflection is known.
244             AX(i_temp, i_temp - 2) = var_temp^3;
245             AX(i_temp, i_temp - 1) = var_temp^2;
246             AX(i_temp, i_temp) = var_temp;
247             AX(i_temp, i_temp + 1) = 1;
248             AX(i_temp, i_temp + 2) = 0;
249             AX(i_temp, i_temp  + 3) = 0;
250             AX(i_temp, i_temp + 4) = 0;
251             AX(i_temp, i_temp + 5) = 0;
252
253             % Deflection left and right of the support are equal.
254             AX(i_temp + 1, i_temp - 2)= var_temp^3;
255             AX(i_temp + 1, i_temp - 1) = var_temp^2;
256             AX(i_temp + 1, i_temp) = var_temp;
257             AX(i_temp + 1, i_temp + 1) = 1;
258             AX(i_temp + 1, i_temp + 2) = -(var_temp^3);
259             AX(i_temp + 1, i_temp + 3) = -(var_temp^2);
260             AX(i_temp + 1, i_temp + 4) = -(var_temp);
261             AX(i_temp + 1, i_temp + 5)= -1;
262
263             % Rotation left and right of the support are equal.
264             AX(i_temp + 2, i_temp - 2)= -(3 * var_temp^2);
265             AX(i_temp + 2, i_temp - 1) = -(2 * var_temp);
266             AX(i_temp + 2, i_temp) = -1;
267             AX(i_temp + 2, i_temp + 1) = 0;
268             AX(i_temp + 2, i_temp + 2) = 3 * var_temp^2;
269             AX(i_temp + 2, i_temp + 3) = 2* var_temp;
270             AX(i_temp + 2, i_temp + 4) = 1;
271             AX(i_temp + 2, i_temp + 5)= 0;
272
273             % Moment left and right of the support is equal.
274             AX(i_temp + 3, i_temp - 2)= -(ei * 6 * var_temp);
275             AX(i_temp + 3, i_temp - 1) = -(ei * 2);
276             AX(i_temp + 3, i_temp) = 0;
277             AX(i_temp + 3, i_temp + 1) = 0;
278             AX(i_temp + 3, i_temp + 2) = ei * 6 * var_temp;
279             AX(i_temp + 3, i_temp + 3) = ei * 2;
```

```
280             AX(i_temp + 3, i_temp + 4) = 0;
281             AX(i_temp + 3, i_temp + 5)= 0;
282         end
283
284         % Last support.
285         % Set variables.
286         i_temp = i * (sx - 1) * 4 - 1; % Index of first entry per loop.
287         var_temp = data{1,1}(ni + i * sx); % x-coordinate of last support
                .
288
289         % Deflection is known.
290         AX(i_temp, i_temp - 2) = var_temp^3;
291         AX(i_temp, i_temp - 1) = var_temp^2;
292         AX(i_temp, i_temp) = var_temp;
293         AX(i_temp, i_temp + 1) = 1;
294
295         % Moment is known.
296         AX(i_temp + 1, i_temp - 2) = -(ei * 6 * var_temp);
297         AX(i_temp + 1, i_temp - 1) = -(ei * 2);
298         AX(i_temp + 1, i_temp) = 0;
299         AX(i_temp + 1, i_temp + 1) = 0;
300
301     else
302         % Bending stiffness of representative strip.
303         % Reduction is 0.5.
304         ei = e * 0.5 * n * (1/12) * rw * w * t^3;
305         %ei = 1; % Used for checking the script.
306
307         % First support.
308         % Set variables.
309         i_temp = 1 + (i - 1) * (sx - 1) * 4; % Index of first entry per
                loop.
310         var_temp = data{1,1}(ni + 1 + (i - 1) * sx); % x-coordinate of
                first support.
311
312         % Deflection is known.
313         AX(i_temp, i_temp) = var_temp^3;
314         AX(i_temp, i_temp + 1) = var_temp^2;
315         AX(i_temp, i_temp + 2) = var_temp;
316         AX(i_temp, i_temp + 3) = 1;
317
318         % Moment is known.
319         AX(i_temp + 1, i_temp) = -(ei * 6 * var_temp);
320         AX(i_temp + 1, i_temp + 1) = -(ei * 2);
321         AX(i_temp + 1, i_temp + 2) = 0;
322         AX(i_temp + 1, i_temp + 3) = 0;
323
324         % Middle supports.
325         for j = 1 : sx - 2
326             % Set variables.
327             i_temp = 3 + (i - 1) * (sx - 1) * 4 + (j - 1) * 4; % Index of
                    entry per loop.
```

```
328             var_temp = data{1,1}(ni + 1 + (i - 1) * sx + j); % x-
                    coordinate of support.
329
330             % Deflection is known.
331             AX(i_temp, i_temp - 2) = var_temp^3;
332             AX(i_temp, i_temp - 1) = var_temp^2;
333             AX(i_temp, i_temp) = var_temp;
334             AX(i_temp, i_temp + 1) = 1;
335             AX(i_temp, i_temp + 2) = 0;
336             AX(i_temp, i_temp  + 3) = 0;
337             AX(i_temp, i_temp + 4) = 0;
338             AX(i_temp, i_temp + 5) = 0;
339
340             % Deflection left and right of the support are equal.
341             AX(i_temp + 1, i_temp - 2)= var_temp^3;
342             AX(i_temp + 1, i_temp - 1) = var_temp^2;
343             AX(i_temp + 1, i_temp) = var_temp;
344             AX(i_temp + 1, i_temp + 1) = 1;
345             AX(i_temp + 1, i_temp + 2) = -(var_temp^3);
346             AX(i_temp + 1, i_temp + 3) = -(var_temp^2);
347             AX(i_temp + 1, i_temp + 4) = -(var_temp);
348             AX(i_temp + 1, i_temp + 5)= -1;
349
350             % Rotation left and right of the support are equal.
351             AX(i_temp + 2, i_temp - 2)= -(3 * var_temp^2);
352             AX(i_temp + 2, i_temp - 1) = -(2 * var_temp);
353             AX(i_temp + 2, i_temp) = -1;
354             AX(i_temp + 2, i_temp + 1) = 0;
355             AX(i_temp + 2, i_temp + 2) = 3 * var_temp^2;
356             AX(i_temp + 2, i_temp + 3) = 2* var_temp;
357             AX(i_temp + 2, i_temp + 4) = 1;
358             AX(i_temp + 2, i_temp + 5)= 0;
359
360             % Moment left and right of the support is equal.
361             AX(i_temp + 3, i_temp - 2)= -(ei * 6 * var_temp);
362             AX(i_temp + 3, i_temp - 1) = -(ei * 2);
363             AX(i_temp + 3, i_temp) = 0;
364             AX(i_temp + 3, i_temp + 1) = 0;
365             AX(i_temp + 3, i_temp + 2) = ei * 6 * var_temp;
366             AX(i_temp + 3, i_temp + 3) = ei * 2;
367             AX(i_temp + 3, i_temp + 4) = 0;
368             AX(i_temp + 3, i_temp + 5)= 0;
369         end
370
371         % Last support.
372         % Set variables.
373         i_temp = i * (sx - 1) * 4 - 1; % Index of first entry per loop.
374         var_temp = data{1,1}(ni + i * sx); % x-coordinate of last support
                .
375
376         % Deflection is known.
377         AX(i_temp, i_temp - 2) = var_temp^3;
378         AX(i_temp, i_temp - 1) = var_temp^2;
```

```
379         AX(i_temp, i_temp) = var_temp;
380         AX(i_temp, i_temp + 1) = 1;
381
382         % Moment is known.
383         AX(i_temp + 1, i_temp - 2) = -(ei * 6 * var_temp);
384         AX(i_temp + 1, i_temp - 1) = -(ei * 2);
385         AX(i_temp + 1, i_temp) = 0;
386         AX(i_temp + 1, i_temp + 1) = 0;
387     end
388 end
389
390 % Create loop to fill the matrix for strips in x-direction.
391 for i = 1 : sy
392     if i ==1
393         % Bending stiffness of representative strip.
394         % Reduction is 0.5.
395         ei = e * 0.5 * n * (1/12) * rw * w * t^3;
396         %ei = 1; % Used for checking the script.
397
398         % Representative load on strip.
399         % Reduction is 0.5.
400         qs = 0.5 * q * d;
401         %qs = 1; % Used for checking the script. % Reduction is 0.5.
402
403         % First support.
404         % Set variables.
405         i_temp = 1 + (i - 1) * (sx - 1) * 4; % Index of first entry per
                loop.
406         var_temp = data{1,1}(ni + 1 + (i - 1) * sx); % x-coordinate of
                first support.
407
408         % Deflection is known.
409         AX(i_temp, ncx + 1) = data{1,3}(ni + 1 + (i - 1) * sx) - ((qs *
                var_temp^4) / (24 * ei));
410
411         % Moment is known.
412         AX(i_temp + 1, ncx + 1) = 0 + ((qs * var_temp^2) / 2);
413
414         % Middle supports.
415         for j = 1 : sx - 2
416             % Set variables.
417             i_temp = 3 + (i - 1) * (sx - 1) * 4 + (j - 1) * 4; % Index of
                    entry per loop.
418             var_temp = data{1,1}(ni + 1 + (i - 1) * sx + j); % x-
                    coordinate of support.
419
420             % Deflection is known.
421             AX(i_temp, ncx + 1) = data{1,3}(ni + 1 + (i - 1) * sx + j) -
                    ((qs * var_temp^4) / (24 * ei));
422
423             % Deflection left and right of the support are equal.
424             AX(i_temp + 1, ncx + 1) = 0;
425
```

```
426             % Rotation left and right of the support are equal.
427             AX(i_temp + 2, ncx + 1) = 0;
428 
429             % Moment left and right of the support are equal.
430             AX(i_temp + 3, ncx + 1) = 0;
431         end
432 
433         % Last support.
434         % Set variables.
435         i_temp = i * (sx - 1) * 4 - 1; % Index of first entry per loop.
436         var_temp = data{1,1}(ni + i * sx); % x-coordinate of last support
                .
437 
438         % Deflection is known.
439         AX(i_temp, ncx + 1) = data{1,3}(ni + i * sx) - ((qs * var_temp^4)
                / (24 * ei));
440 
441         % Moment is known.
442         AX(i_temp + 1, ncx + 1) = 0 + ((qs * var_temp^2) / 2);
443 
444     elseif i < sy
445         % Bending stiffness of representative strip.
446         ei = e * n * (1/12) * rw * w * t^3;
447         %ei = 1; % Used for checking the script.
448 
449         % Representative load on strip.
450         qs = q * d;
451         %qs = 1; % Used for checking the script.
452 
453         % First support.
454         % Set variables.
455         i_temp = 1 + (i - 1) * (sx - 1) * 4; % Index of first entry per
                loop.
456         var_temp = data{1,1}(ni + 1 + (i - 1) * sx); % x-coordinate of
                first support.
457 
458         % Deflection is known.
459         AX(i_temp, ncx + 1) = data{1,3}(ni + 1 + (i - 1) * sx) - ((qs *
                var_temp^4) / (24 * ei));
460 
461         % Moment is known.
462         AX(i_temp + 1, ncx + 1) = 0 + ((qs * var_temp^2) / 2);
463 
464         % Middle supports.
465         for j = 1 : sx - 2
466             % Set variables.
467             i_temp = 3 + (i - 1) * (sx - 1) * 4 + (j - 1) * 4; % Index of
                    entry per loop.
468             var_temp = data{1,1}(ni + 1 + (i - 1) * sx + j); % x-
                    coordinate of support.
469 
470             % Deflection is known.
```

```
471             AX(i_temp, ncx + 1) = data{1,3}(ni + 1 + (i - 1) * sx + j) -
                    ((qs * var_temp^4) / (24 * ei));
472
473             % Deflection left and right of the support are equal.
474             AX(i_temp + 1, ncx + 1) = 0;
475
476             % Rotation left and right of the support are equal.
477             AX(i_temp + 2, ncx + 1) = 0;
478
479             % Moment left and right of the support are equal.
480             AX(i_temp + 3, ncx + 1) = 0;
481         end
482
483         % Last support.
484         % Set variables.
485         i_temp = i * (sx - 1) * 4 - 1; % Index of first entry per loop.
486         var_temp = data{1,1}(ni + i * sx); % x-coordinate of last support
                .
487
488         % Deflection is known.
489         AX(i_temp, ncx + 1) = data{1,3}(ni + i * sx) - ((qs * var_temp^4)
                / (24 * ei));
490
491         % Moment is known.
492         AX(i_temp + 1, ncx + 1) = 0 + ((qs * var_temp^2) / 2);
493
494     else
495         % Bending stiffness of representative strip.
496         % Reduction is 0.5.
497         ei = e * 0.5 * n * (1/12) * rw * w * t^3;
498         %ei = 1; % Used for checking the script.
499
500         % Representative load on strip.
501         % Reduction is 0.5.
502         qs = 0.5 * q * d;
503         %qs = 1; % Used for checking the script.
504
505         % First support.
506         % Set variables.
507         i_temp = 1 + (i - 1) * (sx - 1) * 4; % Index of first entry per
                loop.
508         var_temp = data{1,1}(ni + 1 + (i - 1) * sx); % x-coordinate of
                first support.
509
510         % Deflection is known.
511         AX(i_temp, ncx + 1) = data{1,3}(ni + 1 + (i - 1) * sx) - ((qs *
                var_temp^4) / (24 * ei));
512
513         % Moment is known.
514         AX(i_temp + 1, ncx + 1) = 0 + ((qs * var_temp^2) / 2);
515
516         % Middle supports.
517         for j = 1 : sx - 2
```

```
518             % Set variables.
519             i_temp = 3 + (i - 1) * (sx - 1) * 4 + (j - 1) * 4; % Index of
                    entry per loop.
520             var_temp = data{1,1}(ni + 1 + (i - 1) * sx + j); % x-
                   coordinate of support.
521
522             % Deflection is known.
523             AX(i_temp, ncx + 1) = data{1,3}(ni + 1 + (i - 1) * sx + j) -
                    ((qs * var_temp^4) / (24 * ei));
524
525             % Deflection left and right of the support are equal.
526             AX(i_temp + 1, ncx + 1) = 0;
527
528             % Rotation left and right of the support are equal.
529             AX(i_temp + 2, ncx + 1) = 0;
530
531             % Moment left and right of the support are equal.
532             AX(i_temp + 3, ncx + 1) = 0;
533         end
534
535         % Last support.
536         % Set variables.
537         i_temp = i * (sx - 1) * 4 - 1; % Index of first entry per loop.
538         var_temp = data{1,1}(ni + i * sx); % x-coordinate of last support
                .
539
540         % Deflection is known.
541         AX(i_temp, ncx + 1) = data{1,3}(ni + i * sx) - ((qs * var_temp^4)
                / (24 * ei));
542
543         % Moment is known.
544         AX(i_temp + 1, ncx + 1) = 0 + ((qs * var_temp^2) / 2);
545     end
546 end
547
548 % Solve matices. Use sufficient accuracy!!!
549 CX = rref(AX, 1e-16);
550
551 %% Create results for strips in x-direction.
552 % - deflection
553 % - rotation
554 % - curvature
555 % - moment
556
557 % Create matrices for results of strips in x-direction.
558 WX = zeros(ndx, 3);
559 PHIX = zeros(ndx, 3);
560 KX = zeros(ndx, 3);
561 MX = zeros(ndx, 3);
562 VX = zeros(ndx, 3);
563
564 % Results for strips in x-direction.
565 for i = 1 : sy
```

```
566     if i == 1
567         % Bending stiffness of representative strip.
568         ei = e * 0.5 * n * (1/12) * rw * w * t^3;
569         %ei = 1; % Used for checking the script.
570 
571         % Representative load on strip.
572         % Reduction is 0.5.
573         qs = 0.5 * q * d;
574         %qs = 1; % Used for checking the script.
575 
576         % First entry of strip.
577         % Set variables.
578         i_temp = 1 + (i - 1) * (sx - 1) * 4;
579         c1_temp = CX(i_temp , ncx + 1); % first constant.
580         c2_temp = CX(i_temp + 1, ncx + 1); % second constant.
581         c3_temp = CX(i_temp + 2, ncx + 1); % third constant.
582         c4_temp = CX(i_temp + 3, ncx + 1); % fourth constant.
583         var_x_temp = data{1,1}(ni + 1 + ((i - 1) * sx)); % x-coordinate
                of first support.
584         var_y_temp = data{1,2}(ni + 1 + ((i - 1) * sx)); % y-coordinate
                of first support.
585         i_temp = 1 + (i - 1) * ((sx - 1) * dev + 1);
586 
587         % Assign first entries for deflection.
588         WX(i_temp,1) = var_x_temp;
589         WX(i_temp,2) = var_y_temp;
590         WX(i_temp,3) = w_deflection(var_x_temp, ei, qs, c1_temp, c2_temp,
                c3_temp, c4_temp);
591 
592         % Assign first entries for rotation.
593         PHIX(i_temp,1) = var_x_temp;
594         PHIX(i_temp,2) = var_y_temp;
595         PHIX(i_temp,3) = phi_rotation(var_x_temp, ei, qs, c1_temp,
                c2_temp, c3_temp);
596 
597         % Assign first entries for curvature.
598         KX(i_temp,1) = var_x_temp;
599         KX(i_temp,2) = var_y_temp;
600         KX(i_temp,3) = k_curvature(var_x_temp, ei, qs, c1_temp, c2_temp);
601 
602         % Assign first entries for moment.
603         MX(i_temp,1) = var_x_temp;
604         MX(i_temp,2) = var_y_temp;
605         MX(i_temp,3) = m_moment(var_x_temp, ei, qs, c1_temp, c2_temp);
606 
607         % Assign first entries for shear.
608         VX(i_temp,1) = var_x_temp;
609         VX(i_temp,2) = var_y_temp;
610         VX(i_temp,3) = v_shear(var_x_temp, ei, qs, c1_temp);
611 
612         % Other entries of strip.
613         for j = 1 : sx - 1
614             % Set variables.
```

```
615             i_temp = 1 + (i - 1) * (sx - 1) * 4 + (j - 1) * 4;
616             c1_temp = CX(i_temp, ncx + 1); % first constant.
617             c2_temp = CX(i_temp + 1, ncx + 1); % second constant.
618             c3_temp = CX(i_temp + 2, ncx + 1); % third constant.
619             c4_temp = CX(i_temp + 3, ncx + 1); % fourth constant.
620
621             % Assign entries.
622             for k = 1 : dev
623                 % Set variables.
624                 var_x_temp = data{1,1}(ni + j) + k * (d / dev); % x-
                        coordinate.
625                 var_y_temp = data{1,2}(ni + 1 + ((i - 1) * sx)); % y-
                        coordinate.
626                 i_temp = 1 + (i - 1) * ((sx - 1) * dev + 1) + (j - 1) *
                        dev + k;
627
628                 % Assign entries for deflection.
629                 WX(i_temp, 1) = var_x_temp;
630                 WX(i_temp, 2) = var_y_temp;
631                 WX(i_temp, 3) = w_deflection(var_x_temp, ei, qs, c1_temp,
                         c2_temp, c3_temp, c4_temp);
632
633                 % Assign entries for rotation.
634                 PHIX(i_temp, 1) = var_x_temp;
635                 PHIX(i_temp, 2) = var_y_temp;
636                 PHIX(i_temp, 3) = phi_rotation(var_x_temp, ei, qs,
                        c1_temp, c2_temp, c3_temp);
637
638                 % Assign entries for curvature.
639                 KX(i_temp, 1) = var_x_temp;
640                 KX(i_temp, 2) = var_y_temp;
641                 KX(i_temp, 3) = k_curvature(var_x_temp, ei, qs, c1_temp,
                        c2_temp);
642
643                 % Assign entries for moment.
644                 MX(i_temp, 1) = var_x_temp;
645                 MX(i_temp, 2) = var_y_temp;
646                 MX(i_temp, 3) = m_moment(var_x_temp, ei, qs, c1_temp,
                        c2_temp);
647
648                 % Assign entries for shear.
649                 VX(i_temp, 1) = var_x_temp;
650                 VX(i_temp, 2) = var_y_temp;
651                 VX(i_temp, 3) = v_shear(var_x_temp, ei, qs, c1_temp);
652             end
653         end
654
655     elseif i < sy
656         % Bending stiffness of representative strip.
657         ei = e * n * (1/12) * rw * w * t^3;
658         %ei = 1; % Used for checking the script.
659
660         % Representative load on strip.
```

```
661         qs = q * d;
662         %qs = 1; % Used for checking the script.
663 
664         % First entry of strip.
665         % Set variables.
666         i_temp = 1 + (i - 1) * (sx - 1) * 4;
667         c1_temp = CX(i_temp , ncx + 1); % first constant.
668         c2_temp = CX(i_temp + 1, ncx + 1); % second constant.
669         c3_temp = CX(i_temp + 2, ncx + 1); % third constant.
670         c4_temp = CX(i_temp + 3, ncx + 1); % fourth constant.
671         var_x_temp = data{1,1}(ni + 1 + ((i - 1) * sx)); % x-coordinate
                of first support.
672         var_y_temp = data{1,2}(ni + 1 + ((i - 1) * sx)); % y-coordinate
                of first support.
673         i_temp = 1 + (i - 1) * ((sx - 1) * dev + 1);
674 
675         % Assign first entries for deflection.
676         WX(i_temp,1) = var_x_temp;
677         WX(i_temp,2) = var_y_temp;
678         WX(i_temp,3) = w_deflection(var_x_temp, ei, qs, c1_temp, c2_temp,
                c3_temp, c4_temp);
679 
680         % Assign first entries for rotation.
681         PHIX(i_temp,1) = var_x_temp;
682         PHIX(i_temp,2) = var_y_temp;
683         PHIX(i_temp,3) = phi_rotation(var_x_temp, ei, qs, c1_temp,
                c2_temp, c3_temp);
684 
685         % Assign first entries for curvature.
686         KX(i_temp,1) = var_x_temp;
687         KX(i_temp,2) = var_y_temp;
688         KX(i_temp,3) = k_curvature(var_x_temp, ei, qs, c1_temp, c2_temp);
689 
690         % Assign first entries for moment.
691         MX(i_temp,1) = var_x_temp;
692         MX(i_temp,2) = var_y_temp;
693         MX(i_temp,3) = m_moment(var_x_temp, ei, qs, c1_temp, c2_temp);
694 
695         % Assign first entries for shear.
696         VX(i_temp,1) = var_x_temp;
697         VX(i_temp,2) = var_y_temp;
698         VX(i_temp,3) = v_shear(var_x_temp, ei, qs, c1_temp);
699 
700         % Other entries of strip.
701         for j = 1 : sx - 1
702             % Set variables.
703             i_temp = 1 + (i - 1) * (sx - 1) * 4 + (j - 1) * 4;
704             c1_temp = CX(i_temp, ncx + 1); % first constant.
705             c2_temp = CX(i_temp + 1, ncx + 1); % second constant.
706             c3_temp = CX(i_temp + 2, ncx + 1); % third constant.
707             c4_temp = CX(i_temp + 3, ncx + 1); % fourth constant.
708 
709             % Assign entries.
```

```
710             for k = 1 : dev
711                 % Set variables.
712                 var_x_temp = data{1,1}(ni + j) + k * (d / dev); % x-
                        coordinate.
713                 var_y_temp = data{1,2}(ni + 1 + ((i - 1) * sx)); % y-
                        coordinate.
714                 i_temp = 1 + (i - 1) * ((sx - 1) * dev + 1) + (j - 1) *
                        dev + k;
715
716                 % Assign entries for deflection.
717                 WX(i_temp, 1) = var_x_temp;
718                 WX(i_temp, 2) = var_y_temp;
719                 WX(i_temp, 3) = w_deflection(var_x_temp, ei, qs, c1_temp,
                         c2_temp, c3_temp, c4_temp);
720
721                 % Assign entries for rotation.
722                 PHIX(i_temp, 1) = var_x_temp;
723                 PHIX(i_temp, 2) = var_y_temp;
724                 PHIX(i_temp, 3) = phi_rotation(var_x_temp, ei, qs,
                        c1_temp, c2_temp, c3_temp);
725
726                 % Assign entries for curvature.
727                 KX(i_temp, 1) = var_x_temp;
728                 KX(i_temp, 2) = var_y_temp;
729                 KX(i_temp, 3) = k_curvature(var_x_temp, ei, qs, c1_temp,
                        c2_temp);
730
731                 % Assign entries for moment.
732                 MX(i_temp, 1) = var_x_temp;
733                 MX(i_temp, 2) = var_y_temp;
734                 MX(i_temp, 3) = m_moment(var_x_temp, ei, qs, c1_temp,
                        c2_temp);
735
736                 % Assign entries for shear.
737                 VX(i_temp, 1) = var_x_temp;
738                 VX(i_temp, 2) = var_y_temp;
739                 VX(i_temp, 3) = v_shear(var_x_temp, ei, qs, c1_temp);
740             end
741         end
742
743     else
744         % Bending stiffness of representative strip.
745         ei = e * 0.5 * n * (1/12) * rw * w * t^3;
746         %ei = 1; % Used for checking the script.
747
748         % Representative load on strip.
749         % Reduction is 0.5.
750         qs = 0.5 * q * d;
751         %qs = 1; % Used for checking the script.
752
753         % First entry of strip.
754         % Set variables.
755         i_temp = 1 + (i - 1) * (sx - 1) * 4;
```

```
756         c1_temp = CX(i_temp , ncx + 1); % first constant.
757         c2_temp = CX(i_temp + 1, ncx + 1); % second constant.
758         c3_temp = CX(i_temp + 2, ncx + 1); % third constant.
759         c4_temp = CX(i_temp + 3, ncx + 1); % fourth constant.
760         var_x_temp = data{1,1}(ni + 1 + ((i - 1) * sx)); % x-coordinate
                of first support.
761         var_y_temp = data{1,2}(ni + 1 + ((i - 1) * sx)); % y-coordinate
                of first support.
762         i_temp = 1 + (i - 1) * ((sx - 1) * dev + 1);
763
764         % Assign first entries for deflection.
765         WX(i_temp,1) = var_x_temp;
766         WX(i_temp,2) = var_y_temp;
767         WX(i_temp,3) = w_deflection(var_x_temp, ei, qs, c1_temp, c2_temp,
                c3_temp, c4_temp);
768
769         % Assign first entries for rotation.
770         PHIX(i_temp,1) = var_x_temp;
771         PHIX(i_temp,2) = var_y_temp;
772         PHIX(i_temp,3) = phi_rotation(var_x_temp, ei, qs, c1_temp,
                c2_temp, c3_temp);
773
774         % Assign first entries for curvature.
775         KX(i_temp,1) = var_x_temp;
776         KX(i_temp,2) = var_y_temp;
777         KX(i_temp,3) = k_curvature(var_x_temp, ei, qs, c1_temp, c2_temp);
778
779         % Assign first entries for moment.
780         MX(i_temp,1) = var_x_temp;
781         MX(i_temp,2) = var_y_temp;
782         MX(i_temp,3) = m_moment(var_x_temp, ei, qs, c1_temp, c2_temp);
783
784         % Assign first entries for shear.
785         VX(i_temp,1) = var_x_temp;
786         VX(i_temp,2) = var_y_temp;
787         VX(i_temp,3) = v_shear(var_x_temp, ei, qs, c1_temp);
788
789         % Other entries of strip.
790         for j = 1 : sx - 1
791             % Set variables.
792             i_temp = 1 + (i - 1) * (sx - 1) * 4 + (j - 1) * 4;
793             c1_temp = CX(i_temp, ncx + 1); % first constant.
794             c2_temp = CX(i_temp + 1, ncx + 1); % second constant.
795             c3_temp = CX(i_temp + 2, ncx + 1); % third constant.
796             c4_temp = CX(i_temp + 3, ncx + 1); % fourth constant.
797
798             % Assign entries.
799             for k = 1 : dev
800                 % Set variables.
801                 var_x_temp = data{1,1}(ni + j) + k * (d / dev); % x-
                        coordinate.
802                 var_y_temp = data{1,2}(ni + 1 + ((i - 1) * sx)); % y-
                        coordinate.
```

```
803                 i_temp = 1 + (i - 1) * ((sx - 1) * dev + 1) + (j - 1) *
                        dev + k;
804
805                 % Assign entries for deflection.
806                 WX(i_temp, 1) = var_x_temp;
807                 WX(i_temp, 2) = var_y_temp;
808                 WX(i_temp, 3) = w_deflection(var_x_temp, ei, qs, c1_temp,
                        c2_temp, c3_temp, c4_temp);
809
810                 % Assign entries for rotation.
811                 PHIX(i_temp, 1) = var_x_temp;
812                 PHIX(i_temp, 2) = var_y_temp;
813                 PHIX(i_temp, 3) = phi_rotation(var_x_temp, ei, qs,
                        c1_temp, c2_temp, c3_temp);
814
815                 % Assign entries for curvature.
816                 KX(i_temp, 1) = var_x_temp;
817                 KX(i_temp, 2) = var_y_temp;
818                 KX(i_temp, 3) = k_curvature(var_x_temp, ei, qs, c1_temp,
                        c2_temp);
819
820                 % Assign entries for moment.
821                 MX(i_temp, 1) = var_x_temp;
822                 MX(i_temp, 2) = var_y_temp;
823                 MX(i_temp, 3) = m_moment(var_x_temp, ei, qs, c1_temp,
                        c2_temp);
824
825                 % Assign entries for shear.
826                 VX(i_temp, 1) = var_x_temp;
827                 VX(i_temp, 2) = var_y_temp;
828                 VX(i_temp, 3) = v_shear(var_x_temp, ei, qs, c1_temp);
829             end
830         end
831     end
832 end
833
834 %% Create results for support reactions of strips in x-direction.
835
836 % Create matrix for support reactions of strips in x-direction.
837 RX = zeros(st, 3);
838
839 % Create matrix for total of support reactions per strip.
840 FX = zeros(sy, 3);
841
842 % Support reaction for strips in x-direction.
843 for i = 1 : sy
844     if i == 1
845         % Bending stiffness of representative strip.
846         ei = e * 0.5 * n * (1/12) * rw * w * t^3;
847         %ei = 1; % Used for checking the script.
848
849         % Representative load on strip.
850         % Reduction is 0.5.
```

```
851         qs = 0.5 * q * d;
852         %qs = 1; % Used for checking the script.
853
854         % First support.
855         % Set variables.
856         i_temp = 1 + (i - 1) * (sx - 1) * 4;
857         c1_right_temp = CX(i_temp , ncx + 1); % constant of equation
                right of first support.
858         var_x_temp = data{1,1}(ni + 1 + ((i - 1) * sx)); % x-coordinate
                of first support.
859         var_y_temp = data{1,2}(ni + 1 + ((i - 1) * sx)); % y-coordinate
                of first support.
860         i_temp = 1 + (i - 1) * sx;
861
862         % Assign first entries.
863         RX(i_temp,1) = var_x_temp;
864         RX(i_temp,2) = var_y_temp;
865         RX(i_temp,3) = 0 - v_shear(var_x_temp, ei, qs, c1_right_temp);
866
867         % Middle supports.
868         for j = 1 : sx - 2
869             % Set variables.
870             i_temp = 1 + (i - 1) * (sx - 1) * 4 + (j - 1) * 4;
871             c1_left_temp = CX(i_temp, ncx + 1); % constant of equation
                    left of support.
872             c1_right_temp = CX(i_temp + 4, ncx + 1); % constant of
                    equation right of support.
873             var_x_temp = data{1,1}(ni + 1 + j + ((i - 1) * sx)); % x-
                    coordinate of support.
874             var_y_temp = data{1,2}(ni + 1 + j + ((i - 1) * sx)); % y-
                    coordinate of support.
875             i_temp = 1 + j +(i - 1) * sx;
876
877             % Assign first entries.
878             RX(i_temp,1) = var_x_temp;
879             RX(i_temp,2) = var_y_temp;
880             RX(i_temp,3) = v_shear(var_x_temp, ei, qs, c1_left_temp) -
                    v_shear(var_x_temp, ei, qs, c1_right_temp);
881         end
882
883         % Set variables.
884         i_temp = 1 + (sx - 2) * 4 + (i - 1) * (sx - 1) * 4;
885         c1_left_temp = CX(i_temp, ncx + 1); % constant of equation left
                of last support.
886         var_x_temp = data{1,1}(ni + i * sx); % x-coordinate of support.
887         var_y_temp = data{1,2}(ni + i * sx); % y-coordinate of support.
888         i_temp = i * sx;
889
890         % Assign first entries.
891         RX(i_temp,1) = var_x_temp;
892         RX(i_temp,2) = var_y_temp;
893         RX(i_temp,3) = v_shear(var_x_temp, ei, qs, c1_left_temp);
894
```

```
895         % Total of support reactions per strip.
896         s_temp = 1 + (i - 1) * sx;
897         e_temp = sx + (i - 1) * sx;
898         FX(i,2) = RX(s_temp,2);
899         FX(i,3) = sum(RX((s_temp : e_temp),3));
900 
901     elseif i < sy
902         % Bending stiffness of representative strip.
903         ei = e * n * (1/12) * rw * w * t^3;
904         %ei = 1; % Used for checking the script.
905 
906         % Representative load on strip.
907         qs = q * d;
908         %qs = 1; % Used for checking the script.
909 
910         % First support.
911         % Set variables.
912         i_temp = 1 + (i - 1) * (sx - 1) * 4;
913         c1_right_temp = CX(i_temp , ncx + 1); % constant of equation
                right of first support.
914         var_x_temp = data{1,1}(ni + 1 + ((i - 1) * sx)); % x-coordinate
                of first support.
915         var_y_temp = data{1,2}(ni + 1 + ((i - 1) * sx)); % y-coordinate
                of first support.
916         i_temp = 1 + (i - 1) * sx;
917 
918         % Assign first entries.
919         RX(i_temp,1) = var_x_temp;
920         RX(i_temp,2) = var_y_temp;
921         RX(i_temp,3) = 0 - v_shear(var_x_temp, ei, qs, c1_right_temp);
922 
923         % Middle supports.
924         for j = 1 : sx - 2
925             % Set variables.
926             i_temp = 1 + (i - 1) * (sx - 1) * 4 + (j - 1) * 4;
927             c1_left_temp = CX(i_temp, ncx + 1); % constant of equation
                    left of support.
928             c1_right_temp = CX(i_temp + 4, ncx + 1); % constant of
                    equation right of support.
929             var_x_temp = data{1,1}(ni + 1 + j + ((i - 1) * sx)); % x-
                    coordinate of support.
930             var_y_temp = data{1,2}(ni + 1 + j + ((i - 1) * sx)); % y-
                    coordinate of support.
931             i_temp = 1 + j +(i - 1) * sx;
932 
933             % Assign first entries.
934             RX(i_temp,1) = var_x_temp;
935             RX(i_temp,2) = var_y_temp;
936             RX(i_temp,3) = v_shear(var_x_temp, ei, qs, c1_left_temp) -
                    v_shear(var_x_temp, ei, qs, c1_right_temp);
937         end
938 
939         % Set variables.
```

```
940         i_temp = 1 + (sx - 2) * 4 + (i - 1) * (sx - 1) * 4;
941         c1_left_temp = CX(i_temp, ncx + 1); % constant of equation left
                of last support.
942         var_x_temp = data{1,1}(ni + i * sx); % x-coordinate of support.
943         var_y_temp = data{1,2}(ni + i * sx); % y-coordinate of support.
944         i_temp = i * sx;
945
946         % Assign first entries.
947         RX(i_temp,1) = var_x_temp;
948         RX(i_temp,2) = var_y_temp;
949         RX(i_temp,3) = v_shear(var_x_temp, ei, qs, c1_left_temp);
950
951         % Total of support reactions per strip.
952         s_temp = 1 + (i - 1) * sx;
953         e_temp = sx + (i - 1) * sx;
954         FX(i,2) = RX(s_temp,2);
955         FX(i,3) = sum(RX((s_temp : e_temp),3));
956
957     else
958         % Bending stiffness of representative strip.
959         ei = e * 0.5 * n * (1/12) * rw * w * t^3;
960         %ei = 1; % Used for checking the script.
961
962         % Representative load on strip.
963         % Reduction is 0.5.
964         qs = 0.5 * q * d;
965         %qs = 1; % Used for checking the script.
966
967         % First support.
968         % Set variables.
969         i_temp = 1 + (i - 1) * (sx - 1) * 4;
970         c1_right_temp = CX(i_temp , ncx + 1); % constant of equation
                right of first support.
971         var_x_temp = data{1,1}(ni + 1 + ((i - 1) * sx)); % x-coordinate
                of first support.
972         var_y_temp = data{1,2}(ni + 1 + ((i - 1) * sx)); % y-coordinate
                of first support.
973         i_temp = 1 + (i - 1) * sx;
974
975         % Assign first entries.
976         RX(i_temp,1) = var_x_temp;
977         RX(i_temp,2) = var_y_temp;
978         RX(i_temp,3) = 0 - v_shear(var_x_temp, ei, qs, c1_right_temp);
979
980         % Middle supports.
981         for j = 1 : sx - 2
982             % Set variables.
983             i_temp = 1 + (i - 1) * (sx - 1) * 4 + (j - 1) * 4;
984             c1_left_temp = CX(i_temp, ncx + 1); % constant of equation
                    left of support.
985             c1_right_temp = CX(i_temp + 4, ncx + 1); % constant of
                    equation right of support.
```

```
986             var_x_temp = data{1,1}(ni + 1 + j + ((i - 1) * sx)); % x-
                    coordinate of support.
987             var_y_temp = data{1,2}(ni + 1 + j + ((i - 1) * sx)); % y-
                    coordinate of support.
988             i_temp = 1 + j +(i - 1) * sx;
989
990             % Assign first entries.
991             RX(i_temp,1) = var_x_temp;
992             RX(i_temp,2) = var_y_temp;
993             RX(i_temp,3) = v_shear(var_x_temp, ei, qs, c1_left_temp) -
                    v_shear(var_x_temp, ei, qs, c1_right_temp);
994         end
995
996         % Set variables.
997         i_temp = 1 + (sx - 2) * 4 + (i - 1) * (sx - 1) * 4;
998         c1_left_temp = CX(i_temp, ncx + 1); % constant of equation left
                of last support.
999         var_x_temp = data{1,1}(ni + i * sx); % x-coordinate of support.
1000        var_y_temp = data{1,2}(ni + i * sx); % y-coordinate of support.
1001        i_temp = i * sx;
1002
1003        % Assign first entries.
1004        RX(i_temp,1) = var_x_temp;
1005        RX(i_temp,2) = var_y_temp;
1006        RX(i_temp,3) = v_shear(var_x_temp, ei, qs, c1_left_temp);
1007
1008        % Total of support reactions per strip.
1009        s_temp = 1 + (i - 1) * sx;
1010        e_temp = sx + (i - 1) * sx;
1011        FX(i,2) = RX(s_temp,2);
1012        FX(i,3) = sum(RX((s_temp : e_temp),3));
1013    end
1014end
1015
1016%% Create results for loading of strips in y-direction.
1017
1018%Create matrix for loading of strips in y-direction.
1019QY = zeros(sx * (sy - 1), 1);
1020
1021for i = 1 : sx
1022    for j = 1 : sy - 1
1023        if j == 1
1024            i_temp = (i - 1) * (sy - 1) + j;
1025            QY(i_temp, 1) = -((RX(i, 3) + RX(i + sx, 3) * 0.5) / d);
1026        elseif j < sy - 1
1027            i_temp = (i - 1) * (sy - 1) + j;
1028            QY(i_temp, 1) = -((RX(i + (j - 1) * sx, 3) * 0.5 + RX(i + j *
                    sx, 3) * 0.5) / d);
1029        else
1030            i_temp = (i - 1) * (sy - 1) + j;
1031            QY(i_temp, 1) = -((RX(i + (j - 1) * sx, 3) * 0.5 + RX(i + j *
                    sx, 3)) / d);
1032        end
```

```
1033        end
1034    end
1035
1036    %% Fill matrices for system of equations for strips in y-direction.
1037
1038    % Create matrices to solve.
1039    AY = zeros(ncy, ncy + 1);
1040
1041    % Create loop to fill the matrix for strips in y-direction.
1042    for i = 1 : sx
1043        % Bending stiffness of representative strip.
1044        ei = fsy * e * (1/12) * rw * w * t^3;
1045        %ei = 1; % Used for checking the script.
1046
1047        % First support.
1048        % Set variables.
1049        i_temp = 1 + (i - 1) * (sy - 1) * 4; % Index of first entry per loop.
1050        var_temp = data{1,2}(ni + i); % y-coordinate of first support.
1051
1052        % Deflection is known.
1053        AY(i_temp, i_temp) = var_temp^3;
1054        AY(i_temp, i_temp + 1) = var_temp^2;
1055        AY(i_temp, i_temp + 2) = var_temp;
1056        AY(i_temp, i_temp + 3) = 1;
1057
1058        % Moment is known.
1059        AY(i_temp + 1, i_temp) = -(ei * 6 * var_temp);
1060        AY(i_temp + 1, i_temp + 1) = -(ei * 2);
1061        AY(i_temp + 1, i_temp + 2) = 0;
1062        AY(i_temp + 1, i_temp + 3) = 0;
1063
1064        % Middle supports.
1065        for j = 1 : sy - 2
1066            % Set variables.
1067            i_temp = 3 + (i - 1) * (sy - 1) * 4 + (j - 1) * 4; % Index of
                    entry per loop.
1068            var_temp = data{1,2}(ni + i + j * sx); % y-coordinate of support.
1069
1070            % Deflection is known.
1071            AY(i_temp, i_temp - 2) = var_temp^3;
1072            AY(i_temp, i_temp - 1) = var_temp^2;
1073            AY(i_temp, i_temp) = var_temp;
1074            AY(i_temp, i_temp + 1) = 1;
1075            AY(i_temp, i_temp + 2) = 0;
1076            AY(i_temp, i_temp  + 3) = 0;
1077            AY(i_temp, i_temp + 4) = 0;
1078            AY(i_temp, i_temp + 5) = 0;
1079
1080            % Deflection left and right of the support are equal.
1081            AY(i_temp + 1, i_temp - 2)= var_temp^3;
1082            AY(i_temp + 1, i_temp - 1) = var_temp^2;
1083            AY(i_temp + 1, i_temp) = var_temp;
1084            AY(i_temp + 1, i_temp + 1) = 1;
```

```
1085            AY(i_temp + 1, i_temp + 2) = -(var_temp^3);
1086            AY(i_temp + 1, i_temp + 3) = -(var_temp^2);
1087            AY(i_temp + 1, i_temp + 4) = -(var_temp);
1088            AY(i_temp + 1, i_temp + 5)= -1;
1089
1090            % Rotation left and right of the support are equal.
1091            AY(i_temp + 2, i_temp - 2)= -(3 * var_temp^2);
1092            AY(i_temp + 2, i_temp - 1) = -(2 * var_temp);
1093            AY(i_temp + 2, i_temp) = -1;
1094            AY(i_temp + 2, i_temp + 1) = 0;
1095            AY(i_temp + 2, i_temp + 2) = 3 * var_temp^2;
1096            AY(i_temp + 2, i_temp + 3) = 2* var_temp;
1097            AY(i_temp + 2, i_temp + 4) = 1;
1098            AY(i_temp + 2, i_temp + 5)= 0;
1099
1100            % Moment left and right of the support is equal.
1101            AY(i_temp + 3, i_temp - 2)= -(ei * 6 * var_temp);
1102            AY(i_temp + 3, i_temp - 1) = -(ei * 2);
1103            AY(i_temp + 3, i_temp) = 0;
1104            AY(i_temp + 3, i_temp + 1) = 0;
1105            AY(i_temp + 3, i_temp + 2) = ei * 6 * var_temp;
1106            AY(i_temp + 3, i_temp + 3) = ei * 2;
1107            AY(i_temp + 3, i_temp + 4) = 0;
1108            AY(i_temp + 3, i_temp + 5)= 0;
1109
1110        end
1111
1112        % Last support.
1113        % Set variables.
1114        i_temp = i * (sy - 1) * 4 - 1; % Index of first entry per loop.
1115        var_temp = data{1,2}(ni + i + (sy - 1) * sx); % y-coordinate of last
                support.
1116
1117        % Deflection is known.
1118        AY(i_temp, i_temp - 2) = var_temp^3;
1119        AY(i_temp, i_temp - 1) = var_temp^2;
1120        AY(i_temp, i_temp) = var_temp;
1121        AY(i_temp, i_temp + 1) = 1;
1122
1123        % Moment is known.
1124        AY(i_temp + 1, i_temp - 2) = -(ei * 6 * var_temp);
1125        AY(i_temp + 1, i_temp - 1) = -(ei * 2);
1126        AY(i_temp + 1, i_temp) = 0;
1127        AY(i_temp + 1, i_temp + 1) = 0;
1128
1129    end
1130
1131    % Create loop to fill the matrix for strips in y-direction.
1132    for i = 1 : sx
1133        % Bending stiffness of representative strip.
1134        ei = fsy * e * (1/12) * rw * w * t^3;
1135        %ei = 1; % Used for checking the script.
1136
```

```
1137      % Representative load on strip.
1138      qs_right_temp = QY(1 + (i - 1) * (sy - 1), 1);
1139      %qs_right_temp = 1; % Used for checking the script.
1140
1141      % First support.
1142      % Set variables.
1143      i_temp = 1 + (i - 1) * (sy - 1) * 4; % Index of first entry per loop.
1144      var_temp = data{1,2}(ni + i); % y-coordinate of first support.
1145
1146      % Deflection is known.
1147      AY(i_temp, ncy + 1) = data{1,3}(ni + i) - ((qs_right_temp * var_temp
              ^4) / (24 * ei));
1148
1149      % Moment is known.
1150      AY(i_temp + 1, ncy + 1) = 0 + ((qs * var_temp^2) / 2);
1151
1152      % Middle supports.
1153      for j = 1 : sy - 2
1154          % Representative load on strip.
1155          qs_left_temp = QY((i - 1) * (sy - 1) + j, 1);
1156          %qs_left_temp = 1; % Used for checking the script.
1157          qs_right_temp = QY(1 + (i - 1) * (sy - 1) + j, 1);
1158          %qs_right_temp = 1; % Used for checking the script.
1159
1160          % Set variables.
1161          i_temp = 3 + (i - 1) * (sy - 1) * 4 + (j - 1) * 4; % Index of
                  entry per loop.
1162          var_temp = data{1,2}(ni + i + j * sx); % y-coordinate of support.
1163
1164          % Deflection is known.
1165          AY(i_temp, ncy + 1) = data{1,3}(ni + i + j * sx) - ((qs_left_temp
                   * var_temp^4) / (24 * ei));
1166
1167          % Deflection left and right of the support are equal.
1168          AY(i_temp + 1, ncy + 1) = -((qs_left_temp * var_temp^4) / (24 *
                  ei)) + ((qs_right_temp * var_temp^4) / (24 * ei));
1169
1170          % Rotation left and right of the support are equal.
1171          AY(i_temp + 2, ncy + 1) = -((qs_left_temp * var_temp^3) / (6 * ei
                  )) + ((qs_right_temp * var_temp^3) / (6 * ei));
1172
1173          % Moment left and right of the support are equal.
1174          AY(i_temp + 3, ncy + 1) = -((qs_left_temp * var_temp^2) / 2) + ((
                  qs_right_temp * var_temp^2) / 2);
1175      end
1176
1177      % Last support.
1178      % Set variables.
1179      % Representative load on strip.
1180      qs = QY(i * (sy - 1), 1);
1181      %qs = 1; % Used for checking the script.
1182      i_temp = i * (sy - 1) * 4 - 1; % Index of first entry per loop.
```

```
1183      var_temp = data{1,2}(ni + i + sx * (sy - 1)); % y-coordinate of last
              support.
1184
1185      % Deflection is known.
1186      AY(i_temp, ncy + 1) = data{1,3}(ni + i + sx * (sy - 1)) - ((qs *
              var_temp^4) / (24 * ei));
1187
1188      % Moment is known.
1189      AY(i_temp + 1, ncy + 1) = 0 + ((qs * var_temp^2) / 2);
1190
1191  end
1192
1193  % Solve matices. Use sufficient accuracy!!!
1194  CY = rref(AY, 1e-16);
1195
1196  %% Create results for strips in y-direction.
1197  % - deflection
1198  % - rotation
1199  % - curvature
1200  % - moment
1201
1202  % Create matrices for results of strips in y-direction.
1203  WY = zeros(ndy, 3);
1204  PHIY = zeros(ndy, 3);
1205  KY = zeros(ndy, 3);
1206  MY = zeros(ndy, 3);
1207  VY = zeros(ndy, 3);
1208
1209  % Results for strips in y-direction.
1210  for i = 1 : sx
1211      % Bending stiffness of representative strip.
1212      ei = fsy * e * (1/12) * rw * w * t^3;
1213      %ei = 1; % Used for checking the script.
1214
1215      % Representative load on strip.
1216      qs = QY(1 + (i - 1) * (sy - 1), 1);
1217      %qs = 1; % Used for checking the script.
1218
1219      % First entry of strip.
1220      % Set variables.
1221      i_temp = 1 + (i - 1) * (sy - 1) * 4; % Index of first entry per loop.
1222      c1_temp = CY(i_temp , ncy + 1); % first constant.
1223      c2_temp = CY(i_temp + 1, ncy + 1); % second constant.
1224      c3_temp = CY(i_temp + 2, ncy + 1); % third constant.
1225      c4_temp = CY(i_temp + 3, ncy + 1); % fourth constant.
1226      var_x_temp = data{1,1}(ni + i); % x-coordinate of first support.
1227      var_y_temp = data{1,2}(ni + i); % y-coordinate of first support.
1228      i_temp = 1 + (i - 1) * ((sy - 1) * dev + 1);
1229
1230      %Assign first entries for deflection.
1231      WY(i_temp,1) = var_x_temp;
1232      WY(i_temp,2) = var_y_temp;
```

```
1233        WY(i_temp,3) = w_deflection(var_y_temp, ei, qs, c1_temp, c2_temp,
                c3_temp, c4_temp);
1234
1235        % Assign first entries for rotation.
1236        PHIY(i_temp,1) = var_x_temp;
1237        PHIY(i_temp,2) = var_y_temp;
1238        PHIY(i_temp,3) = phi_rotation(var_y_temp, ei, qs, c1_temp, c2_temp,
                c3_temp);
1239
1240        % Assign first entries for curvature.
1241        KY(i_temp,1) = var_x_temp;
1242        KY(i_temp,2) = var_y_temp;
1243        KY(i_temp,3) = k_curvature(var_y_temp, ei, qs, c1_temp, c2_temp);
1244
1245        % Assign first entries for moment.
1246        MY(i_temp,1) = var_x_temp;
1247        MY(i_temp,2) = var_y_temp;
1248        MY(i_temp,3) = m_moment(var_y_temp, ei, qs, c1_temp, c2_temp);
1249
1250        % Assign first entries for shear.
1251        VY(i_temp,1) = var_x_temp;
1252        VY(i_temp,2) = var_y_temp;
1253        VY(i_temp,3) = v_shear(var_y_temp, ei, qs, c1_temp);
1254
1255        % Other entries of strip.
1256        for j = 1 : sy - 1
1257            % Representative load on strip.
1258            qs = QY((i - 1) * (sy - 1) + j, 1);
1259            %qs = 1; % Used for checking the script.
1260
1261            % Set variables.
1262            i_temp = 1 + (i - 1) * (sy - 1) * 4 + (j - 1) * 4;
1263            c1_temp = CY(i_temp, ncy + 1); % first constant.
1264            c2_temp = CY(i_temp + 1, ncy + 1); % second constant.
1265            c3_temp = CY(i_temp + 2, ncy + 1); % third constant.
1266            c4_temp = CY(i_temp + 3, ncy + 1); % fourth constant.
1267
1268            % Assign entries.
1269            for k = 1 : dev
1270                % Set variables.
1271                var_x_temp = data{1,1}(ni + i + (j - 1) * sx); % x-coordinate
                        .
1272                var_y_temp = data{1,2}(ni + i + (j - 1) * sx) + k * (d / dev)
                        ; % y-coordinate.
1273                i_temp = 1 + (i - 1) * ((sy - 1) * dev + 1) + (j - 1) * dev +
                         k;
1274
1275                % Assign entries for deflection.
1276                WY(i_temp, 1) = var_x_temp;
1277                WY(i_temp, 2) = var_y_temp;
1278                WY(i_temp, 3) = w_deflection(var_y_temp, ei, qs, c1_temp,
                        c2_temp, c3_temp, c4_temp);
1279
```

```
1280                % Assign entries for rotation.
1281                PHIY(i_temp, 1) = var_x_temp;
1282                PHIY(i_temp, 2) = var_y_temp;
1283                PHIY(i_temp, 3) = phi_rotation(var_y_temp, ei, qs, c1_temp,
                        c2_temp, c3_temp);
1284
1285                % Assign entries for curvature.
1286                KY(i_temp, 1) = var_x_temp;
1287                KY(i_temp, 2) = var_y_temp;
1288                KY(i_temp, 3) = k_curvature(var_y_temp, ei, qs, c1_temp,
                        c2_temp);
1289
1290                % Assign entries for moment.
1291                MY(i_temp, 1) = var_x_temp;
1292                MY(i_temp, 2) = var_y_temp;
1293                MY(i_temp, 3) = m_moment(var_y_temp, ei, qs, c1_temp, c2_temp
                        );
1294
1295                % Assign entries for shear.
1296                VY(i_temp, 1) = var_x_temp;
1297                VY(i_temp, 2) = var_y_temp;
1298                VY(i_temp, 3) = v_shear(var_y_temp, ei, qs, c1_temp);
1299
1300            end
1301        end
1302    end
1303
1304    %% Create results for support reactions of strips in y-direction.
1305
1306    % Create matrix for support reactions of strips in y-direction.
1307    RY = zeros(st, 3);
1308
1309    % Create matrix for total of support reactions per strip.
1310    FY = zeros(sy, 3);
1311
1312    % Support reaction for strips in y-direction.
1313    for i = 1 : sx
1314        % Bending stiffness of representative strip.
1315        ei = fsy * e * (1/12) * rw * w * t^3;
1316        %ei = 1; % Used for checking the script.
1317
1318        % First support.
1319        % Representative load on strip.
1320        qs_right_temp = QY(1 + (i - 1) * (sy - 1), 1);
1321        %qs_right_temp = 1; % Used for checking the script.
1322
1323        % Set variables.
1324        i_temp = 1 + (i - 1) * (sy - 1) * 4;
1325        c1_right_temp = CY(i_temp , ncy + 1); % Constant of equation right of
                first support.
1326        var_x_temp = data{1,1}(ni + i); % x-coordinate of first support.
1327        var_y_temp = data{1,2}(ni + i); % y-coordinate of first support.
1328        i_temp = 1 + (i - 1) * sy;
```

```
1329
1330        % Assign first entries.
1331        RY(i_temp,1) = var_x_temp;
1332        RY(i_temp,2) = var_y_temp;
1333        RY(i_temp,3) = 0 - v_shear(var_y_temp, ei, qs_right_temp,
                c1_right_temp);
1334
1335        % Middle supports.
1336        for j = 1 : sy - 2
1337            % Representative load on strip.
1338            qs_left_temp = QY((i - 1) * (sy - 1) + j, 1);
1339            %qs_left_temp = 1; % Used for checking the script.
1340            qs_right_temp = QY(1 + (i - 1) * (sy - 1) + j, 1);
1341            %qs_right_temp = 1; % Used for checking the script.
1342
1343            % Set variables.
1344            i_temp = 1 + (i - 1) * (sy - 1) * 4 + (j - 1) * 4;
1345            c1_left_temp = CY(i_temp, ncy + 1); % Constant of equation left
                    of support.
1346            c1_right_temp = CY(i_temp + 4, ncy + 1); % Constant of equation
                    right of support.
1347            var_x_temp = data{1,1}(ni + i + (j - 1) * sx); % x-coordinate of
                    support.
1348            var_y_temp = data{1,2}(ni + 1 + j + ((i - 1) * sx)); % y-
                    coordinate of support.
1349            i_temp = 1 + (i - 1) * sy + j;
1350
1351            % Assign first entries.
1352            RY(i_temp,1) = var_x_temp;
1353            RY(i_temp,2) = var_y_temp;
1354            RY(i_temp,3) = v_shear(var_y_temp, ei, qs_left_temp, c1_left_temp
                    ) - v_shear(var_y_temp, ei, qs_right_temp, c1_right_temp);
1355        end
1356
1357        % Representative load on strip.
1358        qs_left_temp = QY((i - 1) * (sy - 1) + j, 1);
1359        %qs_left_temp = 1; % Used for checking the script.
1360
1361        % Set variables.
1362        i_temp = 1 + (sy - 2) * 4 + (i - 1) * (sy - 1) * 4;
1363        c1_left_temp = CY(i_temp, ncy + 1); % Constant of equation left of
                last support.
1364        var_x_temp = data{1,1}(ni + i + sx * (sy - 1)); % x-coordinate of
                support.
1365        var_y_temp = data{1,2}(ni + i + sx * (sy - 1)); % y-coordinate of
                support.
1366        i_temp = i * sy;
1367
1368        % Assign first entries.
1369        RY(i_temp,1) = var_x_temp;
1370        RY(i_temp,2) = var_y_temp;
1371        RY(i_temp,3) = v_shear(var_y_temp, ei, qs_left_temp, c1_left_temp);
1372
```

```
1373      % Total of support reactions per strip.
1374      s_temp = 1 + (i - 1) * sy;
1375      e_temp = sy + (i - 1) * sy;
1376      FY(i,2) = RY(s_temp,2);
1377      FY(i,3) = sum(RY((s_temp : e_temp),3));
1378  end
1379
1380  %% Create results for shape of strips in x-direction.
1381
1382  % Create matrices for shape of strips in x-direction.
1383  SHX = zeros(ndx, 3);
1384  SHY = zeros(ndy, 3);
1385
1386  % Position of data points of strips in x-direction.
1387  for i = 1 : ndx
1388      SHX(i, 1) = data{1,1}(ni + st + i);
1389      SHX(i, 2) = data{1,2}(ni + st + i);
1390      SHX(i, 3) = data{1,3}(ni + st + i);
1391  end
1392
1393  % Position of data points of strips in y-direction.
1394  for i = 1 : ndy
1395      SHY(i, 1) = data{1,1}(ni + st + i + ndx);
1396      SHY(i, 2) = data{1,2}(ni + st + i + ndx);
1397      SHY(i, 3) = data{1,3}(ni + st + i + ndx);
1398  end
1399
1400  %% Create results for deviation of strips from shape in x- and y-
         direction.
1401
1402  % Create matrices for deviation of strips from shape in x- and y-
         direction.
1403  DX = zeros(ndx, 3);
1404  DY = zeros(ndy, 3);
1405
1406  % Calculate deviation of strips in x-direction.
1407  for i = 1 : ndx
1408      DX(i, 1) = WX(i, 1);
1409      DX(i, 2) = WX(i, 2);
1410      DX(i, 3) = SHX(i, 3) - WX(i, 3);
1411  end
1412
1413  % Calculate deviation of strips in y-direction.
1414  for i = 1 : ndy
1415      DY(i, 1) = WY(i, 1);
1416      DY(i, 2) = WY(i, 2);
1417      DY(i, 3) = SHY(i, 3) - WY(i, 3);
1418  end
1419
1420  %% Create support points.
1421
1422  % Create matrix for support points.
1423  S = zeros(st, 3);
```

```
1424
1425 % Position of supports.
1426 for i = 1 : st
1427     S(i,1) = data{1,1}(ni + i);
1428     S(i,2) = data{1,2}(ni + i);
1429     S(i,3) = data{1,3}(ni + i);
1430 end
1431
1432 %% Create results for length of strips in x- and y-direction.
1433
1434 % Create matrices for length of strips in x- and y-direction.
1435 LX = zeros(sy, (sx - 1));
1436 LY = zeros(sy, (sy - 1));
1437
1438 % Length of strips in x-direction.
1439 for i = 1 : sy
1440     if i ==1
1441         % Bending stiffness of representative strip.
1442         ei = e * 0.5 * n * (1/12) * rw * w * t^3;
1443         %ei = 1; % Used for checking the script.
1444
1445         % Representative load on strip.
1446         % Reduction is 0.5.
1447         qs = 0.5 * q * d;
1448         %qs = 1; % Used for checking the script.
1449
1450         % Calculate length of strips between two supports.
1451         for j = 1 : sx - 1
1452             % Set variables.
1453             i_temp = 1 + (i - 1) * (sx - 1) * 4 + (j - 1) * 4;
1454             c1_temp = CX(i_temp, ncx + 1); % first constant.
1455             c2_temp = CX(i_temp + 1, ncx + 1); % second constant.
1456             c3_temp = CX(i_temp + 2, ncx + 1); % third constant.
1457
1458             % Assign parameter for length.
1459             l_temp = 0;
1460
1461             % Calculate length of strips between two supports.
1462             for k = 1 : dev
1463                 % Set variables.
1464                 var_xs_temp = data{1,1}(ni + ((i - 1) * sx) + j) + ((k -
                        1) * d / dev);
1465                 var_xe_temp = data{1,1}(ni + ((i - 1) * sx) + j) + (k * d
                         / dev);
1466                 s_temp = sqrt(1 + (phi_rotation(var_xs_temp, ei, qs,
                        c1_temp, c2_temp, c3_temp))^2);
1467                 e_temp = sqrt(1 + (phi_rotation(var_xe_temp, ei, qs,
                        c1_temp, c2_temp, c3_temp))^2);
1468                 trapezium = (d / dev) * (s_temp + e_temp) / 2;
1469
1470                 % Assign length.
1471                 l_temp = l_temp + trapezium;
1472             end
```

```
1473
1474            % Assign length.
1475            LX(i,j) = l_temp;
1476        end
1477
1478    elseif i < sy
1479        % Bending stiffness of representative strip.
1480        ei = e * n * (1/12) * rw * w * t^3;
1481        %ei = 1; % Used for checking the script.
1482
1483        % Representative load on strip.
1484        qs = q * d;
1485        %qs = 1; % Used for checking the script.
1486
1487        % Calculate length of strips between two supports.
1488        for j = 1 : sx - 1
1489            % Set variables.
1490            i_temp = 1 + (i - 1) * (sx - 1) * 4 + (j - 1) * 4;
1491            c1_temp = CX(i_temp, ncx + 1); % first constant.
1492            c2_temp = CX(i_temp + 1, ncx + 1); % second constant.
1493            c3_temp = CX(i_temp + 2, ncx + 1); % third constant.
1494
1495            % Assign parameter for length.
1496            l_temp = 0;
1497
1498            % Calculate length of strips between two supports.
1499            for k = 1 : dev
1500                % Set variables.
1501                var_xs_temp = data{1,1}(ni + ((i - 1) * sx) + j) + ((k -
                        1) * d / dev);
1502                var_xe_temp = data{1,1}(ni + ((i - 1) * sx) + j) + (k * d
                         / dev);
1503                s_temp = sqrt(1 + (phi_rotation(var_xs_temp, ei, qs,
                        c1_temp, c2_temp, c3_temp))^2);
1504                e_temp = sqrt(1 + (phi_rotation(var_xe_temp, ei, qs,
                        c1_temp, c2_temp, c3_temp))^2);
1505                trapezium = (d / dev) * (s_temp + e_temp) / 2;
1506
1507                % Assign length.
1508                l_temp = l_temp + trapezium;
1509            end
1510
1511            % Assign length.
1512            LX(i,j) = l_temp;
1513        end
1514
1515    else
1516        % Bending stiffness of representative strip.
1517        ei = e * 0.5 * n * (1/12) * rw * w * t^3;
1518        %ei = 1; % Used for checking the script.
1519
1520        % Representative load on strip.
1521        % Reduction is 0.5.
```

```
1522            qs = 0.5 * q * d;
1523            %qs = 1; % Used for checking the script.
1524
1525            % Calculate length of strips between two supports.
1526            for j = 1 : sx - 1
1527                % Set variables.
1528                i_temp = 1 + (i - 1) * (sx - 1) * 4 + (j - 1) * 4;
1529                c1_temp = CX(i_temp, ncx + 1); % first constant.
1530                c2_temp = CX(i_temp + 1, ncx + 1); % second constant.
1531                c3_temp = CX(i_temp + 2, ncx + 1); % third constant.
1532
1533                % Assign parameter for length.
1534                l_temp = 0;
1535
1536                % Calculate length of strips between two supports.
1537                for k = 1 : dev
1538                    % Set variables.
1539                    var_xs_temp = data{1,1}(ni + ((i - 1) * sx) + j) + ((k -
                        1) * d / dev);
1540                    var_xe_temp = data{1,1}(ni + ((i - 1) * sx) + j) + (k * d
                         / dev);
1541                    s_temp = sqrt(1 + (phi_rotation(var_xs_temp, ei, qs,
                        c1_temp, c2_temp, c3_temp))^2);
1542                    e_temp = sqrt(1 + (phi_rotation(var_xe_temp, ei, qs,
                        c1_temp, c2_temp, c3_temp))^2);
1543                    trapezium = (d / dev) * (s_temp + e_temp) / 2;
1544
1545                    % Assign length.
1546                    l_temp = l_temp + trapezium;
1547                end
1548
1549                % Assign length.
1550                LX(i,j) = l_temp;
1551            end
1552        end
1553    end
1554
1555    % Length of strips in y-direction.
1556    for i = 1 : sx
1557        % Bending stiffness of representative strip.
1558        ei = fsy * e * (1/12) * rw * w * t^3;
1559        %ei = 1; % Used for checking the script.
1560
1561        % Calculate length of strips between two supports.
1562        for j = 1 : sy - 1
1563            % Set variables.
1564            qs = QY((i - 1) * (sy - 1) + j, 1);
1565            %qs = 1; % Used for checking the script.
1566            i_temp = 1 + (i - 1) * (sy - 1) * 4 + (j - 1) * 4;
1567            c1_temp = CY(i_temp, ncy + 1); % first constant.
1568            c2_temp = CY(i_temp + 1, ncy + 1); % second constant.
1569            c3_temp = CY(i_temp + 2, ncy + 1); % third constant.
1570
```

```
1571          % Assign parameter for length.
1572          l_temp = 0;
1573
1574          % Calculate length of strips between two supports.
1575          for k = 1 : dev
1576              % Set variables.
1577              var_ys_temp = data{1,2}(ni + i + (j - 1) * sx) + ((k - 1) * d
                      / dev);
1578              var_ye_temp = data{1,2}(ni + i + (j - 1) * sx) + (k * d / dev
                  );
1579              s_temp = sqrt(1 + (phi_rotation(var_ys_temp, ei, qs, c1_temp,
                      c2_temp, c3_temp))^2);
1580              e_temp = sqrt(1 + (phi_rotation(var_ye_temp, ei, qs, c1_temp,
                      c2_temp, c3_temp))^2);
1581              trapezium = (d / dev) * (s_temp + e_temp) / 2;
1582
1583              % Assign length.
1584              l_temp = l_temp + trapezium;
1585          end
1586
1587          % Assign length.
1588          LY(i,j) = l_temp;
1589      end
1590  end
1591
1592
1593  %% Create results for horizontal displacement of supports in x- and y-
         direction.
1594
1595  % Create matrices for horizontal displacement of supports in x- and y-
         direction.
1596  SDX = zeros(st, 3);
1597  SDY = zeros(st, 3);
1598
1599  % Horizontal displacement in x-direction.
1600  for i = 1 : sy
1601      % For every support on a strip in x-direction.
1602      for j = 1 : sx
1603          % Index for middle node.
1604          i_m = (sx + 1) / 2;
1605
1606          % Fill x- and y-position of support.
1607          SDX(j + (i - 1) * sx, 1) = data{1,1}(ni + j + (i - 1) * sx);
1608          SDX(j + (i - 1) * sx, 2) = data{1,2}(ni + j + (i - 1) * sx);
1609
1610          % Support x-position smaller then x-position middle node.
1611          if (j < i_m)
1612              for k = 1 : i_m - j
1613                  SDX(j + (i - 1) * sx, 3) = SDX(j + (i - 1) * sx, 3) + (LX
                          (i, i_m - k) - d);
1614              end
1615          end
1616
```

```
1617          % Support x-position larger then x-position middle node.
1618          if (j > i_m)
1619              for k = 1 : j - i_m
1620                  SDX(j + (i - 1) * sx, 3) = (SDX(j + (i - 1) * sx, 3) - (
                          LX(i, i_m + k - 1) - d));
1621              end
1622          end
1623      end
1624  end
1625
1626  % Horizontal displacement in y-direction.
1627  for i = 1 : sx
1628      % For every support on a strip in y-direction.
1629      for j = 1 : sy
1630          % Index for middle node.
1631          i_m = (sy + 1) / 2;
1632
1633          % Fill x- and y-position of support.
1634          SDY(j + (i - 1) * sy, 1) = data{1,1}(ni + i + (j - 1) * sx);
1635          SDY(j + (i - 1) * sy, 2) = data{1,2}(ni + i + (j - 1) * sx);
1636
1637          % Support y-position smaller then y-position middle node.
1638          if (j < i_m)
1639              for k = 1 : i_m - j
1640                  SDY(j + (i - 1) * sy, 3) = SDY(j + (i - 1) * sy, 3) + (LY
                          (i, i_m - k) - d);
1641              end
1642          end
1643
1644          % Support y-position larger then x-position middle node.
1645          if (j > i_m)
1646              for k = 1 : j - i_m
1647                  SDY(j + (i - 1) * sy, 3) = (SDY(j + (i - 1) * sy, 3) - (
                          LY(i, i_m + k - 1) - d));
1648              end
1649          end
1650      end
1651  end
1652
1653  %% Create results for angle at supports in x- and y-direction.
1654
1655  % Create matrices for angle at supports in x- and y-direction.
1656  SAX = zeros(st, 3);
1657  SAY = zeros(st, 3);
1658
1659  % Angle at supports in x-direction.
1660  for i = 1 : sy
1661      if i == 1
1662          % Bending stiffness of representative strip.
1663          ei = e * 0.5 * n * (1/12) * rw * w * t^3;
1664          %ei = 1; % Used for checking the script.
1665
1666          % Representative load on strip.
```

```
1667          % Reduction is 0.5.
1668          qs = 0.5 * q * d;
1669          %qs = 1; % Used for checking the script.
1670
1671          % First entries.
1672          % Set variables.
1673          i_temp = 1 + (i - 1) * (sx - 1) * 4;
1674          c1_temp = CX(i_temp, ncx + 1); % first constant.
1675          c2_temp = CX(i_temp + 1, ncx + 1); % second constant.
1676          c3_temp = CX(i_temp + 2, ncx + 1); % third constant.
1677          var_x_temp = data{1,1}(ni + 1 + (i - 1) * sx); % x-coordinate of
                 first support.
1678          var_y_temp = data{1,2}(ni + 1 + (i - 1) * sx); % y-coordinate of
                 first support.
1679          i_temp = 1 + (i - 1) * sx;
1680
1681          % Assign first entries.
1682          SAX(i_temp,1) = var_x_temp;
1683          SAX(i_temp,2) = var_y_temp;
1684          SAX(i_temp,3) = phi_rotation(var_x_temp, ei, qs, c1_temp, c2_temp
                 , c3_temp);
1685
1686          % Other entries.
1687          for j = 1 : sx - 1
1688              % Set variables.
1689              i_temp = 1 + (i - 1) * (sx - 1) * 4 + (j - 1) * 4;
1690              c1_temp = CX(i_temp, ncx + 1); % first constant.
1691              c2_temp = CX(i_temp + 1, ncx + 1); % second constant.
1692              c3_temp = CX(i_temp + 2, ncx + 1); % third constant.
1693              var_x_temp = data{1,1}(ni + 1 + j + (i - 1) * sx); % x-
                     coordinate of support.
1694              var_y_temp = data{1,2}(ni + 1 + j + (i - 1) * sx); % y-
                     coordinate of support.
1695              i_temp = 1 + j + (i - 1) * sx;
1696
1697              % Assign entries.
1698              SAX(i_temp, 1) = var_x_temp;
1699              SAX(i_temp, 2) = var_y_temp;
1700              SAX(i_temp, 3) = phi_rotation(var_x_temp, ei, qs, c1_temp,
                     c2_temp, c3_temp);
1701          end
1702
1703      elseif i < sy
1704          % Bending stiffness of representative strip.
1705          ei = e * n * (1/12) * rw * w * t^3;
1706          %ei = 1; % Used for checking the script.
1707
1708          % Representative load on strip.
1709          qs = q * d;
1710          %qs = 1; % Used for checking the script.
1711
1712          % First entries.
1713          % Set variables.
```

```
1714            i_temp = 1 + (i - 1) * (sx - 1) * 4;
1715            c1_temp = CX(i_temp, ncx + 1); % first constant.
1716            c2_temp = CX(i_temp + 1, ncx + 1); % second constant.
1717            c3_temp = CX(i_temp + 2, ncx + 1); % third constant.
1718            var_x_temp = data{1,1}(ni + 1 + (i - 1) * sx); % x-coordinate of
                    first support.
1719            var_y_temp = data{1,2}(ni + 1 + (i - 1) * sx); % y-coordinate of
                    first support.
1720            i_temp = 1 + (i - 1) * sx;
1721
1722            % Assign first entries.
1723            SAX(i_temp,1) = var_x_temp;
1724            SAX(i_temp,2) = var_y_temp;
1725            SAX(i_temp,3) = phi_rotation(var_x_temp, ei, qs, c1_temp, c2_temp
                    , c3_temp);
1726
1727            % Other entries.
1728            for j = 1 : sx - 1
1729                % Set variables.
1730                i_temp = 1 + (i - 1) * (sx - 1) * 4 + (j - 1) * 4;
1731                c1_temp = CX(i_temp, ncx + 1); % first constant.
1732                c2_temp = CX(i_temp + 1, ncx + 1); % second constant.
1733                c3_temp = CX(i_temp + 2, ncx + 1); % third constant.
1734                var_x_temp = data{1,1}(ni + 1 + j + (i - 1) * sx); % x-
                        coordinate of support.
1735                var_y_temp = data{1,2}(ni + 1 + j + (i - 1) * sx); % y-
                        coordinate of support.
1736                i_temp = 1 + j + (i - 1) * sx;
1737
1738                % Assign entries.
1739                SAX(i_temp, 1) = var_x_temp;
1740                SAX(i_temp, 2) = var_y_temp;
1741                SAX(i_temp, 3) = phi_rotation(var_x_temp, ei, qs, c1_temp,
                        c2_temp, c3_temp);
1742            end
1743
1744        else
1745            % Bending stiffness of representative strip.
1746            ei = e * 0.5 * n * (1/12) * rw * w * t^3;
1747            %ei = 1; % Used for checking the script.
1748
1749            % Representative load on strip.
1750            % Reduction is 0.5.
1751            qs = 0.5 * q * d;
1752            %qs = 1; % Used for checking the script.
1753
1754            % First entries.
1755            % Set variables.
1756            i_temp = 1 + (i - 1) * (sx - 1) * 4;
1757            c1_temp = CX(i_temp, ncx + 1); % first constant.
1758            c2_temp = CX(i_temp + 1, ncx + 1); % second constant.
1759            c3_temp = CX(i_temp + 2, ncx + 1); % third constant.
```

```
1760            var_x_temp = data{1,1}(ni + 1 + (i - 1) * sx); % x-coordinate of
                    first support.
1761            var_y_temp = data{1,2}(ni + 1 + (i - 1) * sx); % y-coordinate of
                    first support.
1762            i_temp = 1 + (i - 1) * sx;
1763
1764            % Assign first entries.
1765            SAX(i_temp,1) = var_x_temp;
1766            SAX(i_temp,2) = var_y_temp;
1767            SAX(i_temp,3) = phi_rotation(var_x_temp, ei, qs, c1_temp, c2_temp
                    , c3_temp);
1768
1769            % Other entries.
1770            for j = 1 : sx - 1
1771                % Set variables.
1772                i_temp = 1 + (i - 1) * (sx - 1) * 4 + (j - 1) * 4;
1773                c1_temp = CX(i_temp, ncx + 1); % first constant.
1774                c2_temp = CX(i_temp + 1, ncx + 1); % second constant.
1775                c3_temp = CX(i_temp + 2, ncx + 1); % third constant.
1776                var_x_temp = data{1,1}(ni + 1 + j + (i - 1) * sx); % x-
                        coordinate of support.
1777                var_y_temp = data{1,2}(ni + 1 + j + (i - 1) * sx); % y-
                        coordinate of support.
1778                i_temp = 1 + j + (i - 1) * sx;
1779
1780                % Assign entries.
1781                SAX(i_temp, 1) = var_x_temp;
1782                SAX(i_temp, 2) = var_y_temp;
1783                SAX(i_temp, 3) = phi_rotation(var_x_temp, ei, qs, c1_temp,
                        c2_temp, c3_temp);
1784            end
1785        end
1786    end
1787
1788    % Angle at supports in y-direction.
1789    for i = 1 : sx
1790        % For the first support.
1791        % Bending stiffness of representative strip.
1792        ei = fsy * e * (1/12) * rw * w * t^3;
1793        %ei = 1; % Used for checking the script.
1794
1795        % Representative load on strip.
1796        qs = QY(1 + (i - 1) * (sy - 1), 1);
1797        %qs = 1; % Used for checking the script.
1798
1799        % Set variables.
1800        i_temp = 1 + (i - 1) * (sy - 1) * 4; % Index of first entry per loop.
1801        c1_temp = CY(i_temp , ncy + 1); % first constant.
1802        c2_temp = CY(i_temp + 1, ncy + 1); % second constant.
1803        c3_temp = CY(i_temp + 2, ncy + 1); % third constant.
1804        var_x_temp = data{1,1}(ni + i); % x-coordinate of first support.
1805        var_y_temp = data{1,2}(ni + i); % y-coordinate of first support.
1806        i_temp = 1 + (i - 1) * sy;
```

```
1807      %i_temp = 1 + (i - 1) * ((sy - 1) * dev + 1);
1808
1809      % Assign first entries.
1810      SAY(i_temp,1) = var_x_temp;
1811      SAY(i_temp,2) = var_y_temp;
1812      SAY(i_temp,3) = phi_rotation(var_y_temp, ei, qs, c1_temp, c2_temp,
              c3_temp);
1813
1814      % For the other supports.
1815      for j = 1 : sy - 1
1816          % Representative load on strip.
1817          qs = QY((i - 1) * (sy - 1) + j, 1);
1818          %qs = 1; % Used for checking the script.
1819
1820          % Set variables.
1821          i_temp = 1 + (i - 1) * (sy - 1) * 4 + (j - 1) * 4;
1822          c1_temp = CY(i_temp, ncy + 1); % first constant.
1823          c2_temp = CY(i_temp + 1, ncy + 1); % second constant.
1824          c3_temp = CY(i_temp + 2, ncy + 1); % third constant.
1825          c4_temp = CY(i_temp + 3, ncy + 1); % fourth constant.
1826          var_x_temp = data{1,1}(ni + i + j * sx); % x-coordinate of
                  support.
1827          var_y_temp = data{1,2}(ni + i + j * sx); % y-coordinate of
                  support.
1828          i_temp = 1 + (i - 1) * sy + j;
1829
1830          % Assign entries.
1831          SAY(i_temp, 1) = var_x_temp;
1832          SAY(i_temp, 2) = var_y_temp;
1833          SAY(i_temp, 3) = phi_rotation(var_y_temp, ei, qs, c1_temp,
                  c2_temp, c3_temp);
1834      end
1835  end
1836
1837
1838  %% Create results for compensation at supports in x- and y-direction.
1839
1840  % Create matrices for compensation at supports in x- and y-direction.
1841  SCX = zeros(st,3);
1842  SCY = zeros(st,3);
1843
1844  % Compensation in x-direction.
1845  for i = 1 : sy
1846      % For every support on a strip in x-direction.
1847      for j = 1 : sx
1848          % Index for middle node.
1849          i_m = (sx + 1) / 2;
1850
1851          % Fill x- and y-position of support.
1852          SCX(j + (i - 1) * sx, 1) = data{1,1}(ni + j + (i - 1) * sx);
1853          SCX(j + (i - 1) * sx, 2) = data{1,2}(ni + j + (i - 1) * sx);
1854
1855          % Support x-position smaller then x-position middle node.
```

```
1856            if (j < i_m)
1857                SCX(j + (i - 1) * sx, 3) = hc_compensation(-SAX(j + (i - 1) *
                        sx, 3), -SDX(j + (i - 1) * sx, 3), l1, l2);
1858            end
1859
1860            % Support x-position smaller then x-position middle node.
1861            if (j > i_m)
1862                SCX(j + (i - 1) * sx, 3) = hc_compensation(SAX(j + (i - 1) *
                       sx, 3), SDX(j + (i - 1) * sx, 3), l1, l2);
1863            end
1864        end
1865    end
1866
1867    % Compensation in y-direction.
1868    for i = 1 : sx
1869        % For every support on a strip in y-direction.
1870        for j = 1 : sy
1871            % Index for middle node.
1872            i_m = (sy + 1) / 2;
1873
1874            % Fill x- and y-position of support.
1875            SCY(j + (i - 1) * sy, 1) = data{1,1}(ni + i + (j - 1) * sx);
1876            SCY(j + (i - 1) * sy, 2) = data{1,2}(ni + i + (j - 1) * sx);
1877
1878            % Support y-position smaller then y-position middle node.
1879            if (j < i_m)
1880                SCY(j + (i - 1) * sy, 3) = hc_compensation(-SAY(j + (i - 1) *
                        sy, 3), -SDY(j + (i - 1) * sy, 3), l1, l2);
1881            end
1882
1883            % Support x-position smaller then y-position middle node.
1884            if (j > i_m)
1885                SCY(j + (i - 1) * sy, 3) = hc_compensation(SAY(j + (i - 1) *
                       sy, 3), SDY(j + (i - 1) * sy, 3), l1, l2);
1886            end
1887        end
1888    end
1889
1890    % Combine compensation in x- and y-direction.
1891    % Create matrix for combined compensation.
1892    SC = zeros(st,3);
1893    for i = 1 : sy
1894        for j = 1 : sx
1895            SC(j + (i - 1) * sx, 1) = SCX(j + (i - 1) * sx, 1);
1896            SC(j + (i - 1) * sx, 2) = SCX(j + (i - 1) * sx, 2);
1897            SC(j + (i - 1) * sx, 3) = SCX(j + (i - 1) * sx, 3) + SCY(i + (j -
                    1) * sy, 3);
1898        end
1899    end
1900
1901    %% PART THREE Presenting results
1902    % Plot results for shape of strips in x- and y-direction.
1903
```

```
% Shape of strips in x- and y-direction.
subplot(4,4,1)

% Plot results shape of strips in x-direction.
for i = 1 : sy
    s_temp = 1 + (i - 1) * ((sx - 1) * dev + 1);
    e_temp = i * ((sx - 1) * dev + 1);
    plot3(SHX((s_temp : e_temp),1), SHX((s_temp : e_temp),2), SHX((s_temp
        : e_temp),3), 'black');
    grid on;
    hold on;
end

% Plot results shape of strips in x-direction.
for i = 1 : sx
    s_temp = 1 + (i - 1) * ((sy - 1) * dev + 1);
    e_temp = i * ((sy - 1) * dev + 1);
    plot3(SHY((s_temp : e_temp),1), SHY((s_temp : e_temp),2), SHY((s_temp
        : e_temp),3), 'black');
    grid on;
    hold on;
end

% Set axes range.
axis([min(SHX(:,1)) max(SHX(:,1)) min(SHX(:,2)) max(SHX(:,2)) min(SHX
    (:,3)) max(SHX(:,3))])

hold off

title('Shape [mm]')


%% Plot results for deflection of strips in x- and y-direction and
    supports.

% Deflection of strips in x-direction.
subplot(4,4,2)

% Plot results.
for i = 1 : sy
    s_temp = 1 + (i - 1) * ((sx - 1) * dev + 1);
    e_temp = i * ((sx - 1) * dev + 1);
    plot3(WX((s_temp : e_temp),1), WX((s_temp : e_temp),2), WX((s_temp :
        e_temp),3), 'black');
    grid on;
    hold on;
end

% Deflection of strips in y-direction.
subplot(4,4,2)

% Plot results.
for i = 1 : sx
```

```
1952      s_temp = 1 + (i - 1) * ((sy - 1) * dev + 1);
1953      e_temp = i * ((sy - 1) * dev + 1);
1954      plot3(WY((s_temp : e_temp),1), WY((s_temp : e_temp),2), WY((s_temp :
              e_temp),3), 'black');
1955      grid on;
1956      hold on;
1957  end
1958
1959  % Supports
1960  scatter3(S(:,1), S(:,2), S(:,3), 50, 'filled', 'black');
1961
1962  % Set axes range.
1963  axis([min(WX(:,1)) max(WX(:,1)) min(WX(:,2)) max(WX(:,2)) min(WX(:,3))
         max(WX(:,3))])
1964
1965  hold off;
1966
1967  title('Deflection and supports [mm]')
1968
1969  %% Plot results for deviation of strips in x- and y-direction and
         supports.
1970
1971  % Deviation of strips in x-direction.
1972  subplot(4,4,3)
1973
1974  % Plot results.
1975  for i = 1 : sy
1976      s_temp = 1 + (i - 1) * ((sx - 1) * dev + 1);
1977      e_temp = i * ((sx - 1) * dev + 1);
1978      plot3(DX((s_temp : e_temp),1), DX((s_temp : e_temp),2), DX((s_temp :
              e_temp),3), 'black');
1979      grid on;
1980      hold on;
1981  end
1982
1983  % Plot lines between results and zero.
1984  for i = 1 : sy
1985      s_temp = 1 + (i - 1) * ((sx - 1) * dev + 1);
1986      e_temp = i * ((sx - 1) * dev + 1);
1987      stem3(DX((s_temp : e_temp),1), DX((s_temp : e_temp),2), DX((s_temp :
              e_temp),3), 'Color', [0.8 0.8 0.8], 'Marker', 'none');
1988      grid on;
1989      hold on;
1990  end
1991
1992  % Set axes range.
1993  axis([min(DX(:,1)) max(DX(:,1)) min(DX(:,2)) max(DX(:,2)) min(DX(:,3))
         max(DX(:,3))])
1994
1995  hold off
1996
1997  title('Deviation x-direction [mm]')
1998
```

```
1999 % Deviation of strips in y-direction.
2000 subplot(4,4,4)
2001
2002 % Plot results.
2003 for i = 1 : sx
2004     s_temp = 1 + (i - 1) * ((sy - 1) * dev + 1);
2005     e_temp = i * ((sy - 1) * dev + 1);
2006     plot3(DY((s_temp : e_temp),1), DY((s_temp : e_temp),2), DY((s_temp :
             e_temp),3), 'black');
2007     grid on;
2008     hold on;
2009 end
2010
2011 % Plot lines between results and zero.
2012 for i = 1 : sx
2013     s_temp = 1 + (i - 1) * ((sy - 1) * dev + 1);
2014     e_temp = i * ((sy - 1) * dev + 1);
2015     stem3(DY((s_temp : e_temp),1), DY((s_temp : e_temp),2), DY((s_temp :
             e_temp),3), 'Color', [0.8 0.8 0.8], 'Marker', 'none');
2016     grid on;
2017     hold on;
2018 end
2019
2020 % Set axes range.
2021 axis([min(DY(:,1)) max(DY(:,1)) min(DY(:,2)) max(DY(:,2)) min(DY(:,3))
        max(DY(:,3))])
2022
2023 hold off
2024
2025 title('Deviation y-direction [mm]')
2026
2027 %% Plot results for rotation of strips in x- and y-direction.
2028
2029 % Rotation of strips in x-direction.
2030 subplot(4,4,5)
2031
2032 % Plot results.
2033 for i = 1 : sy
2034     s_temp = 1 + (i - 1) * ((sx - 1) * dev + 1);
2035     e_temp = i * ((sx - 1) * dev + 1);
2036     plot3(PHIX((s_temp : e_temp),1), PHIX((s_temp : e_temp),2), PHIX((
             s_temp : e_temp),3), 'black');
2037     grid on;
2038     hold on;
2039 end
2040
2041 % Plot lines between results and zero.
2042 for i = 1 : sy
2043     s_temp = 1 + (i - 1) * ((sx - 1) * dev + 1);
2044     e_temp = i * ((sx - 1) * dev + 1);
2045     stem3(PHIX((s_temp : e_temp),1), PHIX((s_temp : e_temp),2), PHIX((
             s_temp : e_temp),3), 'Color', [0.8 0.8 0.8], 'Marker', 'none');
2046     grid on;
```

```
2047     hold on;
2048 end
2049
2050 % Set axes range.
2051 axis([min(PHIX(:,1)) max(PHIX(:,1)) min(PHIX(:,2)) max(PHIX(:,2)) min(
         PHIX(:,3)) max(PHIX(:,3))])
2052
2053 hold off
2054
2055 title('Rotation x-direction [rad]')
2056
2057 % Rotation of strips in y-direction.
2058 subplot(4,4,6)
2059
2060 % Plot results.
2061 for i = 1 : sx
2062     s_temp = 1 + (i - 1) * ((sy - 1) * dev + 1);
2063     e_temp = i * ((sy - 1) * dev + 1);
2064     plot3(PHIY((s_temp : e_temp),1), PHIY((s_temp : e_temp),2), PHIY((
             s_temp : e_temp),3), 'black');
2065     grid on;
2066     hold on;
2067 end
2068
2069 % Plot lines between results and zero.
2070 for i = 1 : sx
2071     s_temp = 1 + (i - 1) * ((sy - 1) * dev + 1);
2072     e_temp = i * ((sy - 1) * dev + 1);
2073     stem3(PHIY((s_temp : e_temp),1), PHIY((s_temp : e_temp),2), PHIY((
             s_temp : e_temp),3), 'Color', [0.8 0.8 0.8], 'Marker', 'none');
2074     grid on;
2075     hold on;
2076 end
2077
2078 % Set axes range.
2079 axis([min(PHIY(:,1)) max(PHIY(:,1)) min(PHIY(:,2)) max(PHIY(:,2)) min(
         PHIY(:,3)) max(PHIY(:,3))])
2080
2081 hold off
2082 title('Rotation y-direction [rad]')
2083
2084 %% Plot results for curvature of strips in x- and y-direction.
2085
2086 % Curvature of strips in x-direction.
2087 subplot(4,4,7)
2088
2089 % Plot results.
2090 for i = 1 : sy
2091     s_temp = 1 + (i - 1) * ((sx - 1) * dev + 1);
2092     e_temp = i * ((sx - 1) * dev + 1);
2093     plot3(KX((s_temp : e_temp),1), KX((s_temp : e_temp),2), KX((s_temp :
             e_temp),3), 'black');
2094     grid on;
```

```
2095      hold on;
2096  end
2097
2098  % Plot lines between results and zero.
2099  for i = 1 : sy
2100      s_temp = 1 + (i - 1) * ((sx - 1) * dev + 1);
2101      e_temp = i * ((sx - 1) * dev + 1);
2102      stem3(KX((s_temp : e_temp),1), KX((s_temp : e_temp),2), KX((s_temp :
             e_temp),3), 'Color', [0.8 0.8 0.8], 'Marker', 'none');
2103      grid on;
2104      hold on;
2105  end
2106
2107  % Set axes range.
2108  axis([min(KX(:,1)) max(KX(:,1)) min(KX(:,2)) max(KX(:,2)) min(KX(:,3))
        max(KX(:,3))])
2109
2110  hold off
2111
2112  title('Curvature x-direction [1/mm]')
2113
2114  % Curvature of strips in y-direction.
2115  subplot(4,4,8)
2116
2117  % Plot results.
2118  for i = 1 : sx
2119      s_temp = 1 + (i - 1) * ((sy - 1) * dev + 1);
2120      e_temp = i * ((sy - 1) * dev + 1);
2121      plot3(KY((s_temp : e_temp),1), KY((s_temp : e_temp),2), KY((s_temp :
             e_temp),3), 'black');
2122      grid on;
2123      hold on;
2124  end
2125
2126  % Plot lines between results and zero.
2127  for i = 1 : sx
2128      s_temp = 1 + (i - 1) * ((sy - 1) * dev + 1);
2129      e_temp = i * ((sy - 1) * dev + 1);
2130      stem3(KY((s_temp : e_temp),1), KY((s_temp : e_temp),2), KY((s_temp :
             e_temp),3), 'Color', [0.8 0.8 0.8], 'Marker', 'none');
2131      grid on;
2132      hold on;
2133  end
2134
2135  % Set axes range.
2136  axis([min(KY(:,1)) max(KY(:,1)) min(KY(:,2)) max(KY(:,2)) min(KY(:,3))
        max(KY(:,3))])
2137
2138  hold off
2139
2140  title('Curvature y-direction [1/mm]')
2141
2142  %% Plot results for moment of strips in x-direction.
```

```
2143
2144 % Moment of strips in x-direction.
2145 subplot(4,4,9)
2146
2147 for i = 1 : sy
2148     s_temp = 1 + (i - 1) * ((sx - 1) * dev + 1);
2149     e_temp = i * ((sx - 1) * dev + 1);
2150     plot3(MX((s_temp : e_temp),1), MX((s_temp : e_temp),2), MX((s_temp :
             e_temp),3), 'black');
2151     grid on;
2152     hold on;
2153 end
2154
2155 hold off
2156
2157 title('Moment x-direction [Nmm]')
2158
2159 % Moment of strips in y-direction.
2160 subplot(4,4,10)
2161
2162 for i = 1 : sx
2163     s_temp = 1 + (i - 1) * ((sy - 1) * dev + 1);
2164     e_temp = i * ((sy - 1) * dev + 1);
2165     plot3(MY((s_temp : e_temp),1), MY((s_temp : e_temp),2), MY((s_temp :
             e_temp),3), 'black');
2166     grid on;
2167     hold on;
2168 end
2169
2170 hold off
2171
2172 title('Moment y-direction [Nmm]')
2173
2174 %% Plot results for shear of strips in x-direction.
2175
2176 % Shear of strips in x-direction.
2177 subplot(4,4,11)
2178
2179 for i = 1 : sy
2180     s_temp = 1 + (i - 1) * ((sx - 1) * dev + 1);
2181     e_temp = i * ((sx - 1) * dev + 1);
2182     plot3(VX((s_temp : e_temp),1), VX((s_temp : e_temp),2), VX((s_temp :
             e_temp),3), 'black');
2183     grid on;
2184     hold on;
2185 end
2186
2187 hold off
2188
2189 title('Shear x-direction [N]')
2190
2191 % Shear of strips in y-direction.
2192 subplot(4,4,12)
```

```
2193
2194 for i = 1 : sx
2195     s_temp = 1 + (i - 1) * ((sy - 1) * dev + 1);
2196     e_temp = i * ((sy - 1) * dev + 1);
2197     plot3(VY((s_temp : e_temp),1), VY((s_temp : e_temp),2), VY((s_temp :
            e_temp),3), 'black');
2198     grid on;
2199     hold on;
2200 end
2201
2202 hold off
2203
2204 title('Shear y-direction [N]')
2205
2206 %% Create messages for checks of vertical forces.
2207
2208 disp('CHECKS VERTICAL FORCES:');
2209 disp(['Total load: ', num2str(q * (sx - 1) * (sy - 1) * d^2), ' [N]']);
2210 disp(['Sum of support reactions top strips: ', num2str(sum(FX(:,3))), ' [
        N]']);
2211 disp(['Sum of support reactions lower strips: ', num2str(sum(FY(:,3))), '
         [N]']);
2212 disp('(Note: when q = 0 a load of 1e-16 [N/mm] is used!!!)');
2213 disp(' ')
2214
2215 %% Plot results for compensation of supports.
2216
2217 subplot(4,4,13)
2218
2219 scatter3(SC(:,1), SC(:,2), SC(:,3), 50, 'filled', 'black');
2220
2221 title('Compensation of supports [mm]')
```

Bibliography

Azer Akhundov, 2008. URL http://www.skyscrapercity.com/showthread.php?t=856424&page=22.

Anengineersaspect, 2009. URL http://anengineersaspect.blogspot.nl/2009/10/22-pier-luigi-nervi-structures-on.html.

J. Blaauwendraad and A.W.M. Kok. *Elementenmethode voor constructeurs deel 2 (In Dutch)*. Agon Elsevier, Amsterdam/Brussel, 1973.

C.R. Calladine. *Theory of shell structures*. Cambridge University Press, 1983.

Kristin Dispenza. Zaha hadid's heydar aliyev cultural centre: Turning a vision into reality, June 2011. URL http://buildipedia.com/on-site/from-the-job-site/zaha-hadids-heydar-aliyev-cultural-centre-turning-a-vision-into-reality?print=1&tmpl=component.

Ductal. La crèche pierre budin, 2013. URL http://www.ductal-lafarge.fr/wps/portal/ductal/fr/3_6_1-Detail?WCM_GLOBAL_CONTEXT=/wps/wcm/connectlib_ductal/Site_ductal/AllKeyProject/KeyProjectDuctal+Page_1323158995627/Content+KeyProjectDuctal+fr.

FARO. Faro laser scanner photon 120/20. Technical report, FARO, 2012.

Carl Friedrich Gauss, James Caddall Morehead, and Adam Miller Hiltebeitel. *General investigations of curved surfaces of 1827 and 1825*. General Investigations of Curved Surfaces of 1827 and 1825. The Princeton university library, 1902.

K.F. Gauss. General investigations of curved surfaces. *Royal Society of Göttingen*, 1827.

Leica Geosystems, 2013. URL http://hds.leica-geosystems.com/en/Leica-Cyclone_6515.htm.

Eline Hartog, den. Prefabrication of concrete shells. Master's thesis, Delft University of Technology, December 2008.

Coen Hartsuijker and Hans Welleman. *Non-symetrical and inhomogeneous cross sections (Lecture notes structural mechanics 4, CT3109)*. Delft University of Technology, Faculty of Civil Engineering and Geosciences, 2011.

Coenraad Hartsuijker. *Toegepaste mechanica, Deel 2*. Sdu Uitgever, 2007.

Koen Huyge and Arnoud Schoofs. Precast double curved concrete panels. Master's thesis, Delft University of Technology, June 2009.

Infomatique, 2010. URL http://www.panoramio.com/photo_explorer#view=photo&position=1920&with_photo_id=71462640&order=date_desc&user=711866.

Hans Jansen. Prefabricage bij blobconstructies, een civiele kijk op blobarchitectuur. Master's thesis, Delft University of Technology, November 2004.

Bas Janssen. Double curved precast load bearing concrete elements. Master's thesis, Delft University of Technology, 2011.

W.T. Koiter. *Inleiding tot de leer van stijfheid en sterkte (In Dutch)*. Epsilon Uitgaven, Utrecht, 1985.

Marijn Kok. Textile reinforced double curved concrete elements - manufacturing free-form architecture with a flexible mould. Master's thesis, Technische Universiteit Delft, 2013.

Florian Peter Kosche, 1998.

Lasplash, n.d. URL http://www.lasplash.com/publish/International_151/Sydney_Opera_House_Review.php.

Lauren and James, 2012. URL http://campervancouver.blogspot.nl/2012/03/our-day-in-la-with-car.html.

Lev Manovich. Nurbs theory - conceptualizing cultural processes: from discrete categories to continuous curves, December 2008. URL http://lab.softwarestudies.com/2008/12/theory-for-nurbs-era-from-timelines-and.html.

Nedcam. Photo's of cnc mills of nedcam. 2012a.

Nedcam, 2012b. URL http://www.nedcam.nl/.

NLArchitects. Het funen, 2012. URL http://www.nlarchitects.nl/ucslideshow/55.

Renzo Piano. Progettazione sperimentale per strutture a guscio - experimental project of shell structures casabella. *Kunststoffen in dragende Konstrukties*, pages 38–49, 1969.

Marlies Quack. Dubbel gekromde gevelelementen: van ontwerp naar uitvoering. *Optimum*, 2001.

M. K. H. M. Roosbroeck. The construction of prefab concrete shells. Master's thesis, Delft University of Technology, 2006.

Roel Schipper, 2013.

Roel Schipper and Bas Janssen. Manufacturing double-curved elements in precast concrete using a flexible mould - first experimental results. In *fib Symposium Praque 2011*, June 2011.

Shadesofgreendesign, 2011. URL http://shadesofgreendesign.com.au/biomimicry-structural-lessons-from-orchid/.

Lars Spuybroek. *Nox: Machining architecture*. Thames & Hudson, 2004. ISBN 9780500285190.

A.K. van der Vegt. *Kunststoffen in het kort*. Centraal Boekhuis, 1994. ISBN 9789066742185.

K. J. Vollers. De pinbed wizard. March 2013. URL http://www.materia-ic.nl/wp-content/uploads/downloads/2013/03/MateriaXpertSeries20130321_KV.pdf.

Karel Jan Vollers and Daan Rietbergen. A method and apparatus for forming a double-curved panel from a flat panel, December 2008.

Karel Jan Vollers and Daan Rietbergen. Werkwijze en mal voor het vervaardigen van een gebogen paneel., July 2010.