Interactive Formfinding For Optimised Fabric-Cast Concrete

by
Andreas Bak

Supervised by
Dr. Paul Shepherd
Prof. Paul Richens

A thesis submitted for the degree of Master of Philosophy
University of Bath
Department of Architecture and Civil Engineering
October 2011

This page is intentionally left blank.

ACKNOWLEDGEMENTS

I give my thanks to all that participated or helped me during this thesis. Special thanks go to my two supervisors Dr. Paul Shepherd and Professor Paul Richens for the many hours they spent on supervision. And also thanks for the inspirational conversations and discussions with Dr. Chris Williams.
I also give my thanks to Mark West for providing unpublished material.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Producing organic shapes in concrete has been a challenging problem since complex freeform buildings became a major trend in contemporary architecture. Many different techniques for casting doubly curved shapes have been proposed. Most of them produce elements which exactly match a preconceived design, but at a high cost in manufacture. Fabric formwork techniques (such as those pioneered at the Centre of Architectural Structures and Technology at the University of Manitoba (CAST)) are relatively economical, but require a form-finding approach which takes into account the physics of casting, as well as structural and functional requirements of the finished element.

This thesis presents a specialised methodology for the design and manufacture of optimized concrete elements cast in fabric formwork. Using a novel software tool, the approach proposed in this thesis lies in between the largely intuitive methods reported by CAST and the precise but expensive manufacturing methods normally used in practice. Combining topological optimisation with computational formfinding, the software guides the designer towards a shape that is economical in both material and manufacturability.

By combining knowledge of computational structural analysis, optimisation algorithms, fabric simulation and the practical casting techniques of fabric formwork; this thesis bridges the gap between structurally optimized forms, and those developed intuitively by fabric casting.

A prototype software tool (FabricCast) specialized for the design of centrally supported slabs is presented in detail, with a few design studies realised using plaster scale models.

CHAPTER 1 INTRODUCTION

This chapter describes the motivation and background for this thesis. The need for a method of optimising the shape of concrete slabs cast using fabric formwork techniques is explained. Finally it contains a short description of the outcome of this thesis.

1.1 CONTEXT

Producing organic shapes with concrete has been a key issue since complex freeform architecture became a major trend in contemporary architecture. Many different techniques of casting doubly curved shapes have been proposed (Jepsen et al., 2010). Common to many of them is that they force the shape to mimic the desired design, but at a high price in the cost of manufacture.
At the other end of the scale is the research in fabric formwork conducted by Mark West and his architectural students at the Centre for Architectural Structures and Technology (CAST) at the University of Manitoba in Canada.

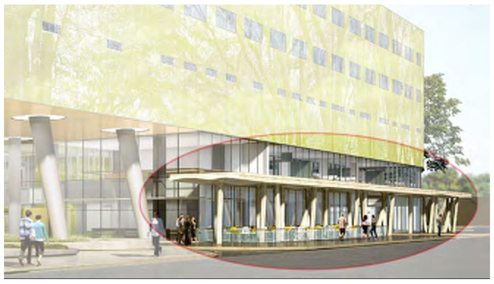


Figure 1.1: Render of conopy at the Women's hospital in Winnipeg (West, 2011)

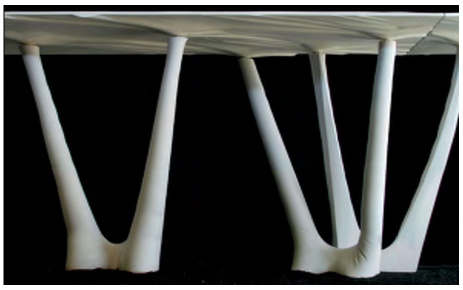


Figure 1.2: Plaster model of fabric cast columns supporting the slab (West, 2011)

CAST have been exploring the natural shapes of a fabric under the influence by forces such as supports, self-weight of concrete and pre-stress and the use of this natural shape as a formwork for casting concrete elements.

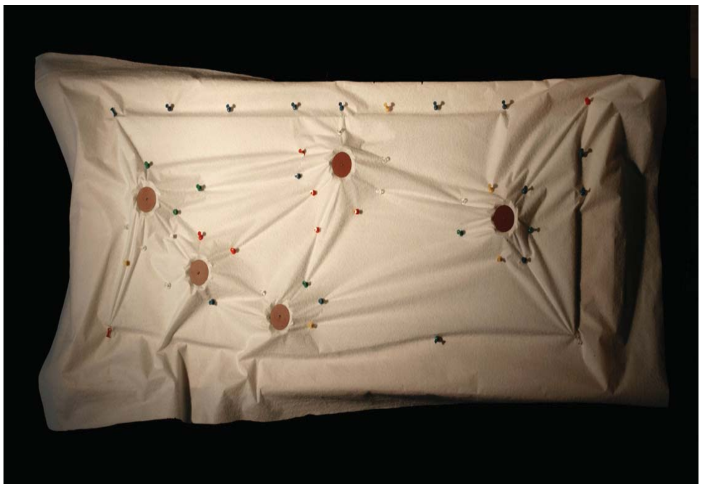


Figure 1.3: Scale model of fabric arranged and pre-stressed on a flat surface and pinned down (West, 2011)

Many well-defined structural elements have been investigated over the past several years. The focus of this research has lately turned towards concrete slabs.

A fabric formed slab has recently been designed for a canopy at the new Women's Hospital in Winnipeg (Figure 1.1, Figure 1.2 and Figure 1.3).

The slab was designed by placing fabric on a flat base then rearranging it in to the desired shape taking into account the support conditions. The driving force for the design using this method was aesthetics and the structural experience and intuition of the manufacturer.

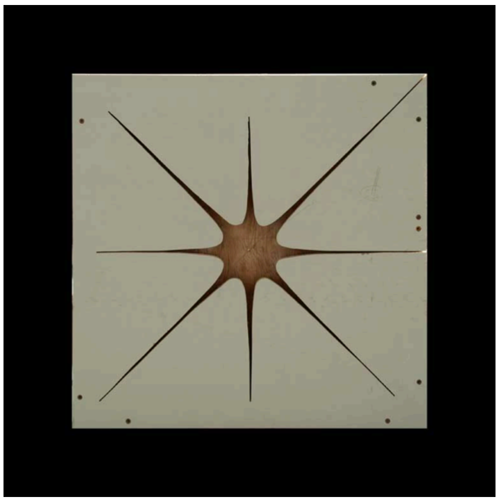


Figure 1.4: Cut-out pattern for fabric mould (West, 2011)

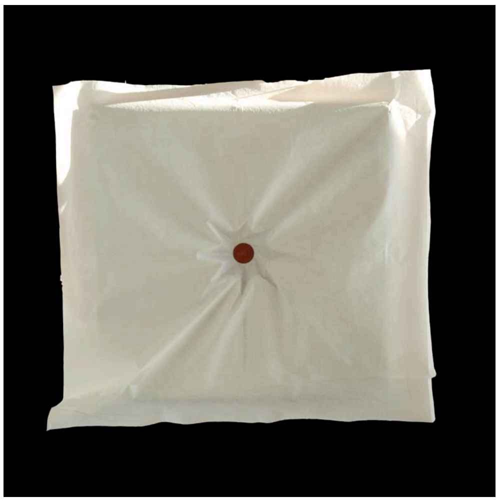


Figure 1.5: Fabric placed on top of cut-out pattern (West, 2011)

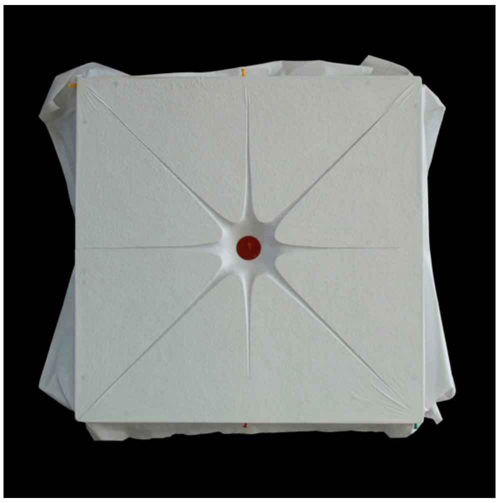


Figure 1.6: Fabric is pre-stressed and vacuum applied (West, 2011)

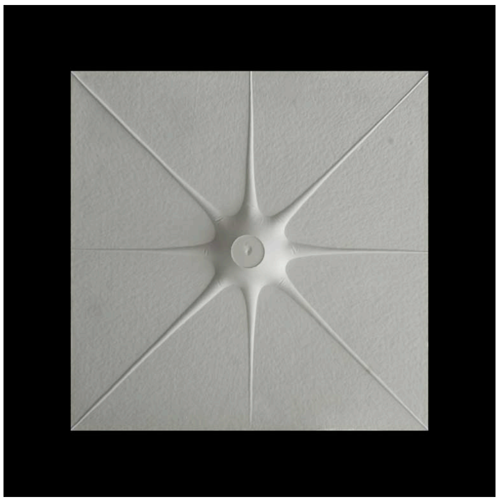


Figure 1.7: Finished plaster model of star capital slab (West, 2011)

A different technique was used to create a capital star slab. In this case the fabric was placed over a predefined cut-out pattern, pre-tensioned and forced in to shape by a vacuum (Figure 1.4, Figure 1.5, Figure 1.6 and Figure 1.7). The shapes created using the casting techniques conducted by CAST can be aesthetically pleasing but are only structurally optimised to the extend of the designers intuition. This thesis proposes a way of optimising the shapes of slabs cast using fabric formwork.

1.2 CONTEMPORARY DRIVERS

Demands from the aeronautic and automobile industry have pushed the development of several topology optimisation techniques over the last couple of decades. The demands are driven by the desire for lightweight elements with high performance in term of different parameters such as stiffness, natural frequency, strength etc. These algorithms are now widely used in the field of mechanical engineering. Despite the big advantages of these algorithms they have never gained ground in the building industry. The reason for this is that Architecture and Civil Engineering differs from Mechanical Engineering in many ways. One of the differences is the need for aesthetically pleasing design. The optimal design in regards to stiffness etc. might not be aesthetically pleasing and therefore unwanted. The scale of the design is also a huge difference that should be considered when using topology optimisation as design tool. In mechanical engineering the optimisation techniques are often used on small parts with a very well defined domain and boundary conditions. When increasing the scale to, for example, a slab, one has to consider different materials and manufacturing techniques. Suddenly the domain and boundary conditions become a design parameter (Dombernowsky and Søndergaard, 2010).

Common to the already well-known topology optimisation techniques is that they produce organic looking shapes that usually cannot be cast using conventional techniques. This thesis proposes an optimisation algorithm that produces results that can be approximated by casting concrete using fabric formwork. This algorithm is based on the Bi-Directional Structural optimisation algorithm (Huang and Xie, 2007). The algorithm is based on an engineering intuition that an optimal shape of a structure can be found by removing unutilised material from the design. The word 'bi-directional' refers to the fact that material can also be added back to the design if it is determined that the structure would benefit from this.

The proposed customisation of the method takes certain manufacturing constraints into account when determining where material should be removed or added. Instead of considering the full design domain, a sub domain is used. Only material within this sub domain is considered when deciding what to remove or add. The domain is updated at every iteration step of the optimisation. Using this method, a topology that can be approximated using fabric formwork is derived.

This method can produce aesthetically pleasing concrete slabs whilst saving a huge amount of material with the fewest consequences in respect to manufacturability and structural integrity.

The optimised shape is used to determine the constraints of the fabric formwork by digitally draping a fabric over it. This is done by simulating the dynamic behaviour of the fabric in real-time by numerical integrating Newton's second law of motion.

The shape of the draped fabric is used to produce a cut-out pattern in a flat surface. This cut-out can be used as a guide when draping a "real life" fabric through it, thus creating an approximation to the optimised shape.

The aim of this thesis is to create a software tool that places the design somewhere in between the intuitive approach conducted by Mark West and an approach depending on the precise but expensive manufacturing methods available today. The tool should provide an interactive framework for, the design of fabric formwork.

The main part of this thesis describes the development of the software tool, FabricCast and the methods used to achieve its functionality. The software includes user input in form of design domain, boundary conditions, optimisation parameters and material properties of the fabric and optimised material. The software also allows for user interaction by directly manipulating the fabric.
By combining topology optimisation techniques with computational form finding the software will help the manufacturer make the right decisions towards an optimal design of the slab based on the applied boundary conditions.
In other words, this research bridges the gap between the 'digital optimised' and the 'manufactured shape' by combining knowledge of computational structural analysis, optimisation algorithms, fabric simulation and the practical casting techniques of fabric formwork.

1.3 STRUCTURE OF THESIS

Chapter 2 contains a review of relevant literature. Previous work in the area of this thesis is described and the potential methods are discussed and an approach for the thesis is decided.

Chapter 3 describes the functionality and structure of the software tool developed as a part of this thesis. An object-oriented approach is presented and a series of classes created as part of the software is described.

Chapter 4 describes the different methods and their implementation in the software. The methods are outlined and the contributions from the author are described in detail.

Chapter 5 contains a case study using the software developed. A quadratic slab supported by a single column in the centre is investigated and two plaster scale models cast using fabric formworks are presented.

Chapter 6 concludes this thesis. A summary of the work presented in the thesis is given and future development is proposed.

CHAPTER 2 LITERATURE REVIEW

This chapter outlines concepts, techniques and recent developments in the multiple fields covered in this thesis. Each section contains an argument for the choice of method.
The first section outlines the past work in relation to the main topic of the thesis, fabric formed slabs. The next sections outline some other topics relevant to the thesis. This section covers a big variety of subjects such as, topology optimisation, fabric simulation and collision handling. A full review of these topics is not within the scope of this thesis, but a description of the most common used approaches is given.

2.1 FABRIC FORMED SLABS

The literature on slabs cast using fabric formwork is very sparse as it is a fairly new field of research. One of the main contributors to the field is the Centre for Architectural Structures and Technology (CAST) at the University of Manitoba. It houses one of the leading labs in the development and design of fabric formed structures. They specialise in a huge variety of pre-cast and in situ cast elements including beams, trusses, columns, wall panels, vaults and slabs. Only a single published paper exists on the work done by CAST (West and Araya, 2009). The author has, through personal communication, acquired some unpublished work on the manufacturing techniques of concrete slabs cast in fabric formwork (West, 2011). A couple of examples can be found in the introduction to this thesis.
For further information the reader is directed to the CAST website where other upublished papers can be downloaded (CAST, 2011).
Although topology optimisation in architecture is a topic that has experienced growth over the past years, and an increasing amount of literature is available, a very limited amount of literature has been found on computational form finding and optimisation of fabric formed slabs.
Dombernowsky and Søndergaard (2010) show a few examples of topology optimised concrete slabs using off-the-shelf software. A full-scale prototype of a topology optimised slab was created using CNC milling of polystyrene blocks. This research was done as part of the research group “UNIKA beton” that explores the possibilities of constructing alternative concrete moulds using robot technology.
A method for designing fabric formed concrete panels was proposed by Schmitz (2006). A surface finite element model of the fabric formwork, and a volumetric finite element model of a non-stiff mass, was defined based on the load paths in a panel. The non-stiff mass serves as the weight of the concrete and applies loads to the fabric model. These two models combined were used to iteratively increase the thickness of the volumetric model based on the displacements of the fabric formwork model until a certain convergence criterion was reached. This form finding technique is highly dependent on the finite element mesh chosen and the correct determination of load paths. This is a task that increases in complexity when the model is subject to irregular boundary conditions.
Van Mele and Block (2010) describe a form finding method for fabric cast concrete shells. The goal of this method was to find the closest possible tension only equilibrium surface to a given target surface, thus creating a basis for fabric formwork of compression only shells. The method is based on the force

density method and thrust network analysis. The outcome of the method is an equilibrium network describing the pre-stress of the fabric formwork. However they do not describe how such an equilibrium network is translated into physical boundary conditions for the fabric formwork.
The master's thesis "Evolutionary Optimization of Fabric Formed Structural Elements" by Veenendaal (2008) is the literature that most closely relates to the main topic of this research. The thesis explores a method using differential evolution (DE) to optimise the shape of beams cast using well-established fabric formwork techniques. A combination of fabric form finding using dynamic relaxation techniques and non-linear finite element analysis of the form found shape is used as a performance criterion of each step in the evolutionary process.

2.2 TOPOLOGY OPTIMISATION

Topology optimisation covers the idea of placing the material where the structure benefits from it the most. This optimal distribution is dependent on the support conditions and the governing loads on a structure. Examples of such optimisation can be found in nature in for example the growth of trees. The base of the tree slowly branches out and thereby evolves into a shape that resists the forces afflicted by the wind.
This concept have been explored numerically and applied to the optimisation of structural elements in a couple of different ways.

SOLID ISOTROPIC MICROSTRUCTURE WITH PENALIZATION

In the late 1980's a famous paper that would be the birth of topology optimisation was published (Bendsøe, 1988). The paper was based on prior research in shape optimisation and eliminated some of the constraints of the shape optimisation techniques. The existing optimisation methods were constrained to the assumption that the initial topology, defined by parametric equations, was fixed during the optimisation process. Therefore the final design could only be found within a predefined domain of solutions.
Without going into the mathematical details, the method is based on the idea of an artificial composite material with microscopic voids, or microstructures, and a density function that varies between 0 and 1 to describe the impact of these microstructures on the material properties. Homogenisation theory was then used to compute the effective material properties based on the density function. The optimal distribution of the material within a predefined design domain was then calculated by treating the problem as a sizing problem.
Because the nature of the method was to produce non-smooth estimates of the exact boundary (Bendsøe, 1988) the method was proposed as the first part of at two step procedure, where regular shape optimisation would be the second step. This early version suffered from a few drawbacks such as mesh dependency and fictitious material properties.
Bendsøe later developed the method where the artificial material densities were penalised thus reducing the occurrence of intermediate densities and reducing the problem of the non-smooth boundaries of the optimal design (Bendsøe, 1989). This was further developed and became the basis for the term Solid Isotropic Microstructure with Penalization (SIMP).

BI-DIRECTIONAL EVOLUTIONARY STRUCTURAL OPTIMISATION

Another well known and widely used algorithm is the Evolutionary Structural Optimisation (ESO) algorithm by Xie (1993). This method is much simpler than the SIMP method and based on engineering intuition, in contrast to the highly mathematical based SIMP method.
The algorithm works by discretising the design domain and iteratively analysing and removing unutilised material until a certain amount has been removed. This "hard-kill" binary nature of ESO is another contrast to the SIMP method in which intermediate values, or analogously porous material, is allowed.
The rejection ratio (RR) defines how much material should be removed at a certain step in the optimisation process. The evolutionary rate (ER) determines how much the RR should be increased after each steady state.
The disadvantages of the early ESO method was the slow evolution due to the fact that the design parameters RR and ER had to be kept very low to avoid elements being removed prematurely and creating an irreversible void that would lead to a non-optimal design.
ESO begins from a full domain and slowly evolves to the optimised topology whereas the opposite applies to the Additive Evolutionary Structural Optimization (AESO) (Querin et al., 2000). This algorithm approaches the optimal design from the under designed minimum domain by adding material around areas with highly utilised material.
Bi-Directional Evolutionary Structural Optimisation (BESO) was introduced by Querin et al. (1998). The method is a combination of the ESO and AESO method in which elements can be added back as well as removed. This method was further researched and improved by Huang and Xie (2007) removing numerical problems such as checkerboard patterns and mesh independency. Furthermore the stability of the method was improved and problems with non-convergence resolved.

The author chose to proceed with the BESO method in this thesis for the following reasons. Firstly it was chosen because of the simplicity of the algorithm and the easy implementation into existing software code.
The discrete nature of the BESO method allows easy evaluation of the optimised shape without the use of surface reconstruction techniques.
The SIMP method is less computationally efficient because the entire domain has to be analysed at every iteration step. In BESO the number of equations decreases when elements are removed. Furthermore, to speed up the process further, the algorithm can be started at an initial guess close to the desired volume fraction. This is especially useful when applying the algorithm to three-dimensional structures (Huang and Xie, 2007).
With the recent developments of BESO, a lot of the early issues with the algorithm have been eliminated, thus creating a fast and stable solution.

TOPOLOGY OPTIMISATION IN ARCHITECTURE

Topology optimised structures in architecture are a rare, mainly because of the limits set by manufacturability as mentioned previously. Despite this a few examples do exist.

Topology optimisation techniques were used to design two of the façades of the Akutagawa River Side office building (Ohmori et al., 2005).
In 2003, architect Arata Isozaki entered a competition to design a new train station in Florence with a proposal using topology optimisation as a design tool (Cui et al., 2003). The project didn't win, but the architect later went on to design the Qatar National Convention center with a similar design approach (Cui et al., 2005). The convention center is currently under construction.

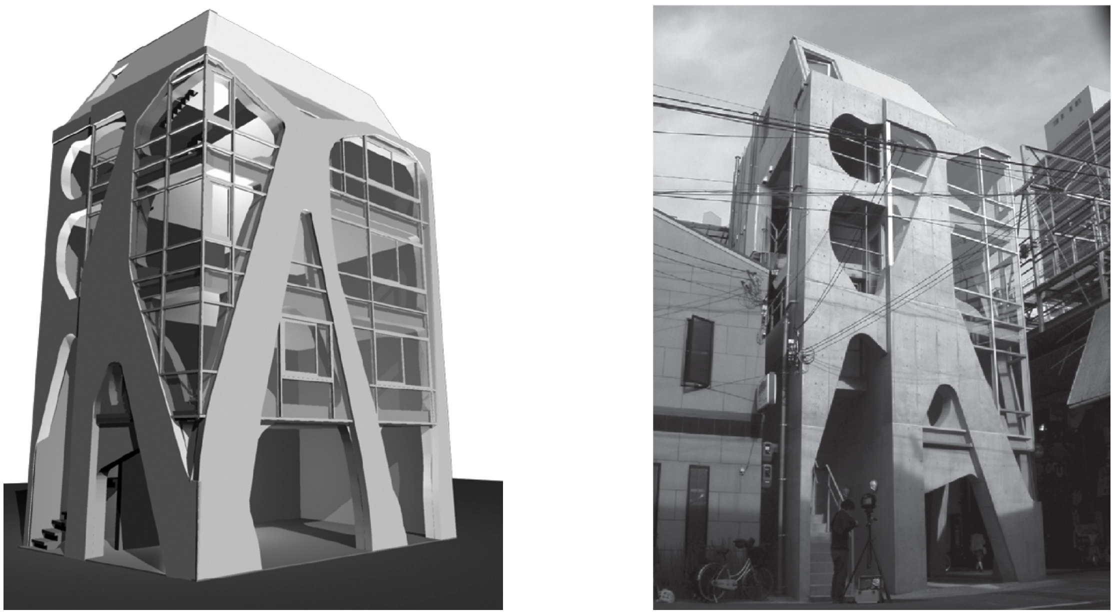


Figure 2.1: Akutagawa River Side office building (Ohmori et al., 2005)

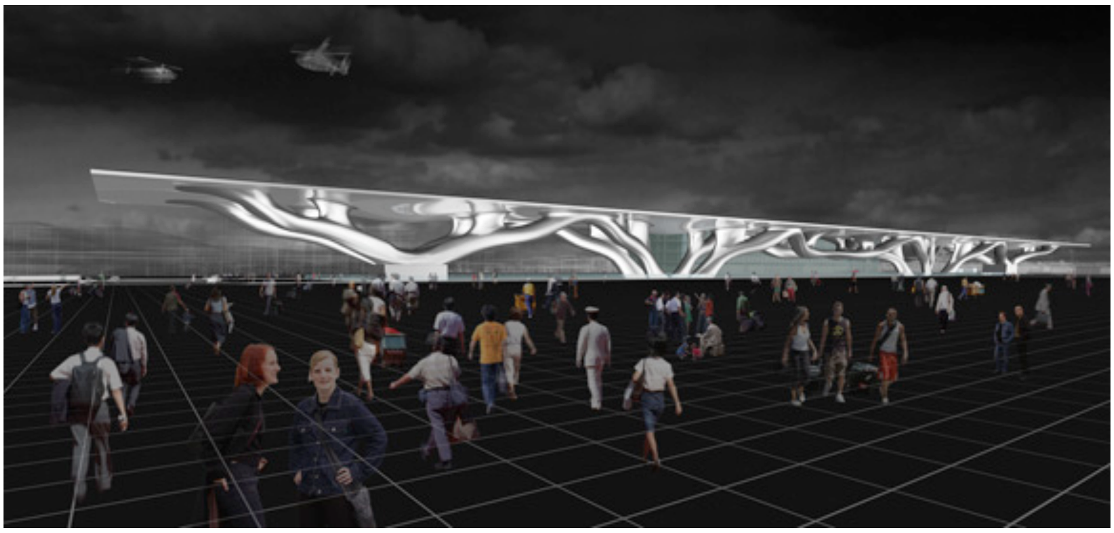


Figure 2.2: Proposal for new train station in Florence (Cui et al., 2003)

Figure 2.3: Qatar Convention Centre (http://www.qatarconvention.com)

2.3 FABRIC SIMULATION

The aim of this research is to simulate the drape of a fabric over an optimised shape.

Cloth simulation, a more common term for fabric simulation, covers the challenges of the simulation of the non-linear and highly deformable mechanical and dynamic behaviour of cloth under the influence of forces and collisions inflicted by the surrounding environment. This can be done with different levels of precision dependent on the application of the cloth simulation.

In computer games the methods used have to be computational efficient because of the limited computation power left over from other tasks. Using approximate methods to simulate cloth usually sacrifices precision for real-time performance. This sacrifice is not critical as the user would not be able to see the difference between a precise model and an approximate one.

The garment industry is another big contributor to the research on cloth simulation but with a different goal. They seek to precisely simulate the behaviour of cloth, taking into account its mechanical properties. By doing so they can create a virtual dressing room that simulates the behaviour of a garment on a person even in real-time.

The main challenges of cloth simulation can be divided into a few main topics concerning mechanical properties, dynamic behaviour and collision handling, including self-collision.

SIMULATING DYNAMIC BEHAVIOUR

Particle systems are the most popular solution to simulate the dynamic behaviour of cloth. This approach offers an intuitive and simple way of modelling the behaviour of cloth and can, in combination with different numerical integration schemes, be easily adapted to a specific need, for example convergence speed or real-time performance.

The integration schemes are used to find the time dependent coordinates of the cloth deforming under influence of load. These coordinates are given implicit as a solution to Newton's second law of motion. This second order differential equation is often simplified to a set of two first order equations (Nealen et al., 2006). These equations can be solved numerically.

A number of numerical integration methods exist and a number of them are discussed by Volino and Magnenat-Thalmann (2001). The simplest method is the forward Euler integration where finite differences are used to calculate the positions and velocities for the next time step.

Explicit numerical methods suffer from instability when the time step is above a certain threshold. For simulations with stiff objects this threshold can be very low, thus reducing the speed of the simulation. This is due to the inaccuracy of the explicit formulation. If the time step is too big the solution for the next step can overshoot the equilibrium state and an accumulation of error, essentially a gain of energy, will eventually result in the model exploding into an irreversible instable state. This is especially true for the simple Euler method and therefore it is seldom used in practice.

An improvement of the Euler integration method is to reverse the order of the calculations of the position and the velocity (Nealen et al., 2006). For non-dissipative systems this reduces to the Störmer-Verlet integration scheme that was popularised by Verlet (1967), but used by Störmer as early as 1905. This method is much more stable than the simple Euler method.

The instability problem can also be solved using implicit methods. These methods do however require a system of algebraic equations to be solved at each time step, making them more computational inefficient.

It is argued that the arbitrarily large time steps allowed in implicit methods balance the increase in computational effort, but this is at the expense of realistic dynamical behaviour. Therefore implicit methods should be used when quick convergence to a rest position is required (Volino and Magnenat-Thalmann, 2001).

MATERIAL MODEL

To be able to define the internal forces of a viscoelastic non-linear behaviour fabric a material model is needed. Because of manufacturing methods cloth is often anisotropic, which further complicates the mechanical model.

Using a spring-mass system, the material behaviour is modelled by calculation of the force interacting on a node based on the neighbouring nodes and the fictitious spring linking them together. Using this simple method it is quite impossible to model the correct in-plane forces.

Volino, Magnenat-Thalmann et al. (2009) proposes a simple and accurate way of modelling the in-plane forces of a non-linear anisotropic material.

The method is at the crossroad between the finite element and the mass-spring systems. It considers forces between particles, but is calculated on the basis of accurate computations of the surface mechanical state in the element connecting to the node.
In finite elements, a usual approach is to select a tensor that relates the deformation of an element to the strain and stress. By choosing the correct tensor the material non-linearity and viscoelasticity can be modelled.
A simple example of such a tensor is Hooke's linear material law that relates the material stress to the material strain. For isotropic materials this tensor only depends on Young's modulus and Poisson's ratio (Nealen et al., 2006).
However a common approach is to use stress-strain relations derived from experimental studies to precisely implement the non-linear behaviour.
A few different approaches modeling the bending stiffness of the material is available (Grinspun et al., 2003) (Volino and Magnenat-Thalmann, 2006).
In this thesis a simple laplacian-smoothing algorithm is used where each internal node of the surface is moved to a new position given by the average position of the neighboring nodes projected to the surface normal.

The high computational time of early attempts to use the popular finite element method to solve the non-linear behaviour proved to be impractical but with the latest development in the field the method is gaining popularity (Magnenat Thalmann, 2010). Further modelling techniques such as finite element and mesh free methods is described in the paper by Nealen et al. (2006).

2.4 COLLISION HANDLING

The software developed for this thesis needs to be able to handle the collision between a static optimised object and a fabric object. This is a problem that has been investigated in detail in the computer gaming industry.
With the development of computer games came a need for handling collisions of objects in real-time. Even though computers today are much faster than back when the first computer games was developed collision handling is still a key challenge in developing fast a responsive computer games because of the move into 3D.
The term collision handling covers the detection of overlapping or intersection of objects such as people with walls etc. Some objects in modern computer games are immensely complex and consist of hundreds or thousands of polygons that all need to be checked for collision. All this has to happen in the blink of an eye to prevent the gamer experiencing lags in the game. The increasing complexity of computer games has led to abundant research on this topic.

BOUNDING GEOMETRIES

Bounding volumes is a widely used technique to simplify intersection tests by using simple polyhedra to encapsulate complex geometry. This sacrifices precision for speed to a degree dependent on the bounding volume used to represent the geometry (Figure 2.4).

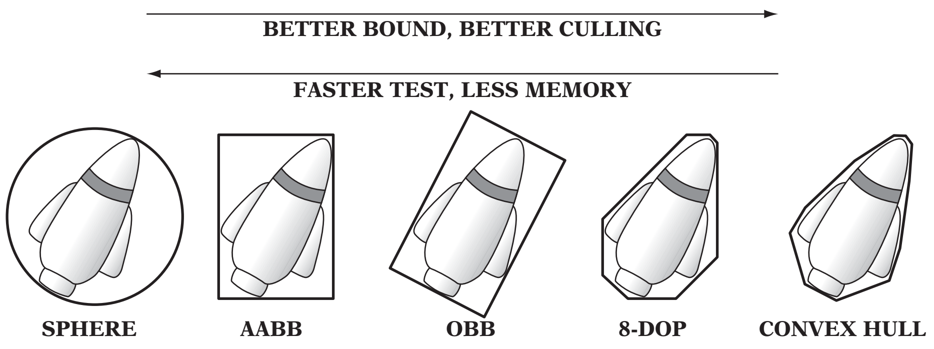


Figure 2.4: Bounding volumes (Ericson, 2005)

The common choice of bounding volume when precision is not an issue is a Bounding Sphere (Larsson, 2008). Axis Aligned Bounding Boxes (AABB) (Yi-Si et al., 2010) are commonly used to represent geometry that is more or less aligned to the coordinate system and therefore can be represented by a bounding box aligned to the coordinate system without compromising too much on the precision of the intersection test. Object Oriented Bounding Boxes (OBB) can be used to further improve approximation of the geometry (Gottschalk et al., 1996).
k-DOP's are bounding volumes that are convex polytopes whose facets are determined by half spaces whose outward normal is predetermined. AABB is a sub category of k-DOP's, a 4-KOP in two dimensions or a 6-KOP in three dimensions (Klosowski et al., 1998).
The Convex Hull of the geometry is also used as a bounding box (Barber et al., 1996). The Convex Hull is the smallest possible convex volume that encapsulates all the vertices of the geometry.

The Bounding Boxes are often used in combination with space subdivision to achieve the fastest possible algorithms (Gottschalk et al., 1996, Yi-Si et al., 2010).

DIVIDING SPACE

A common approach to speed up collision handling is to reduce the number of pairs that need to be tested for intersection. Hierarchies are created by space subdivision to represent the complex geometry, thus reducing the number of calculations by concentrating on a smaller part of the geometry. Such subdivisions are usually performed on the static objects involved in the intersection test.
Binary Space Partitioning (Fuchs et al., 1980) is a technique developed for solving the hidden surface problem when rendering three-dimensional geometries, but it is also serves as a tree structure for collisions detection and other interactive purposes (Ar, 2000). The tree consists of nodes containing planes dividing the scene into two parts, one part above the plane and one part below. Each part is recursively divided into two by a new plane, thus creating a new node in the tree for every subdivision. The decision of where to place the plane can be determined geometric or topologic criteria. Another approach to subdivision of space is the use of octrees. Octrees recursively divide the scene into eight boxes until the amount of objects inside each box is uniform (Wilhelms and Gelder, 1992).

The book “Real-Time Collision Detection” provides a solid foundation for the understanding of the different aspects of collision handling (Ericson, 2005).
The choice of collision handling method comes down to the nature of the geometry in question. In the case of this thesis AABB would be the obvious choice of bounding volume, but due to the fact that the mesh is of a very simple nature, and because of the even distribution and axis alignment, a much simpler algorithm can be derived to detect collision. Therefore the author created a simple collision handling strategy, which could later be changed later, if needed, to implement some of the space dividing methods mentioned above.

CHAPTER 3 SOFTWARE

In this chapter the structure of the software tool that was developed for this thesis is described. The tool helps a designer towards an optimal design of a concrete slab. It consists of a volumetric mesh generator, a finite element solver, a topology optimisation algorithm and a fabric simulation with collision handling. These elements of the software are all available to the user through a common graphical user interface (GUI). The GUI allows the user to input design data such as material properties, geometry, optimisation parameters and boundary conditions. The geometry is displayed as a three-dimensional rendering in a window that allows the user to interact with it.
A short description of the functionality of some selected parts of the software is given. The methods used to achieve the functionality is not described in detail in this chapter, the reader is therefore referred to chapter 4 for further details.
The chapter also covers the choice of programming platform and an outline description of the software flow from initial user input to final model.

3.1 PROGRAMMING PLATFORM

Processing (Processing, 2011) is an open-source cross platform JAVA (Oracle, 2011) based programing language especially used for applications with visual contents. It is a very visual platform that lets the programmer quickly experience the effects of any changes in the code.
Although Processing has a lot of qualities, in terms of ease of use and quick development, the Processing Integrated Development Environment (IDE) supplied is very limited in functionality and a software project of this size would quickly get unmanageable. It is also quite limited in terms of available libraries and no finite element package for continuum mechanics exists. For a further introduction to Processing the reader is referred to literature (Shiffman, 2008).
Due to the reasons mentioned above this project used Processing's big brother JAVA. This was an obvious choice because the syntax of JAVA is similar to that of Processing, so a change would not require much readjustment and code written in Processing could without much effort be rewritten in JAVA.
JAVA is a programming language with a huge number of libraries and IDE's available. It features automatic garbage collection, multi threading, easy development and good documentation. JAVA is available for a variety of platforms, which also makes is an attractive option when choosing programming platform.
JAVA is an object oriented programming language that depends of the development of different types of objects with various properties and methods specific for that object. An object can contain references to other types of objects.
Classes are used to define the type of the object along with the methods and properties. Some objects are part of the JAVA language, but custom objects can be defined by writing bespoke classes.
The object-oriented approach is very intuitive and allows different parts of the software to be developed separately and later combined. In the next section some types of objects, or classes, developed for FabricCast is described.

3.2 CLASSES

As part of the software for this thesis, numerous classes were developed. In this section a few selected classes and their basic functionality is described. More detailed descriptions of the different methods used to achieve the functionality can be found in chapter 4 and source code of the classes can be found on the DVD (Appendix A).

As part of FabricCast the author developed the following classes.

Package bc	**Mesh boundary conditions**
Class Constrain	Defines constrains in world coordinates
Class NodalForce	Defines nodal forces in world coordinates
Package collision	**Collision handling**
FabricMeshIntersection	Detecting and handling collisions between a mesh and a fabric object
Package gui	**Graphical user interface**
MainForm	Container for all graphics and interface objects, extends the JAVA JFrame class
Package Interaction	**User interaction**
HandleBox	Three dimensional interactive box for selection of nodes and elements
RepelingForce	Interactive force that repels fabric objects
Sphere	Interactive sphere object that collides with fabric
Package optimisation	**Optimisation algorithms**
Abstract class Optimisation	Optimisation of mesh objects
Class Beso	BESO algorithm, extends the Optimisation class
Class NewBeso	BESO algorithm with candidates, extends the Optimisation class
Class ConstrainedBeso	BESO algorithm with candidates constrained to bottom, extends the NewBeso class
Package topology	**Topological elements**
Class Edge	Mesh edge, extends the Shape class
Class Fabric	Description of the fabric, extends the Mesh class
Class Face	Mesh face, extends the Shape class
Class Mesh	Description of mesh topology
Class Quad3D	Description of a volumetric mesh of quads, extends the Mesh class
Class Shape	Simple shape class
Class Vertex	Mesh vertex, extends the Shape class
Class Voxel	Mesh voxel, extends the Shape class
Class VoxelHex8	Eight-node voxel, extends the Voxel class
Package util	**Utility classes**
Class Utilities	File writing, math and geometry utilities
Package visualisation	**3D graphics**
Class NodeResults	Computation of node values from finite element object used for visualisation
Class ProcessingRenderer	Rendering of 3D graphics, extends the Processing PApplet class

Table 3.1: Outline of classes developed for the software.

MESH

Meshes consist of a variety of smaller objects such as vertices, edges, faces etc. These objects contain information about connectivity with other objects in the mesh. For example a vertex contains information of which edges, faces or voxels it is connected to and visa versa.

The software includes two different types of mesh objects with completely different topologies and purposes.

The first type of object is the volumetric mesh consisting of vertices and voxels representing the optimised shape (`Quad3D`). The second is two-dimensional mesh of vertices, edges and faces representing the fabric (`Fabric`).

Although they differ in topology and use they share some common properties and therefore a parent class called `Mesh` was created. These classes along with the `Vertex`, `Voxel`, `VoxelHex8`, `Edge`, `Face` and `Shape` class are part of the package `topology`.

The `Quad3D` class allows the user to set up the volume for the optimisation algorithm. It allows the user to input parameters like dimensions, mesh increments, design domain and boundary conditions, such as supports and nodal forces. `Quad3D` is used to generate the mesh topology and contains the information input by the user. The class acts as a container class for objects that define smaller parts of the topology such as vertices and voxels. The class also contains methods used to manipulate the mesh and set or get specific data. The mesh object can be passed to the optimisation algorithm, finite element solver, graphics renderer and collision handling for further manipulation.
One of the main reasons for creating a custom mesh class for the optimised topology is to keep the topological information intact even though elements are removed due to the optimisation algorithm. This allows elements to be turned on and off without worrying about having to regenerate the topological information.
A Boolean flag in the `Voxel` class determines whether an element is solid or void. Only the solid elements are passed to the finite element analysis and the graphics renderer.

OPTIMISATION

One of the main functions of the software is to apply an optimisation algorithm to a volume. The purpose of the algorithm is to optimise the topology of the volume based on applied boundary conditions. The algorithm works in an iterative manner turning voxels of the volumetric mesh on or off based on result from a structural analysis.
A `Quad3D` object is passed to the optimisation class and the object is manipulated according to the result of the optimisation by switching elements on or off. The updated `Quad3D` object is then converted into a Finite Element model and a structural analysis determines the new internal forces of the volume. These results are updated in the `Quad3D` object and the updated object is passed to the optimisation class again.
This procedure continues until a convergent state has ben reached.
The outcome of this algorithm is often a topology with internal voids, but as it is not possible to create these voids with the fabric formwork manufacturing method proposed in this thesis a customised version of the algorithm is needed.
The optimisation algorithm is implemented in the software as a few distinct classes. The `Beso` class represents the original BESO algorithm that the customised algorithm is based on. This class is mainly created for the sake of comparing results with the customised algorithm. It still remains a part of the finished software. The class `NewBeso` is a class that only considers a sub domain of the full domain as removable or addable. ConstrainedBeso is a class that inherits the capabilities of the `NewBeso` class but where the criterion for the sub domain is defined. This criterion comes in to affect when the candidates for removal is selected. Further explanation of this can be found in chapter 4.

The distinction of the `Beso` and the `NewBeso` class is the way the algorithm decides which elements should be turned on or off.

STRUCTURAL ANALYSIS

Another major part of the software is the implementation of a structural analysis method. This method is used iteratively as part of the optimisation cycle described in the previous section. In topology optimisation, especially when the topology optimisation method relies on a discretised domain, the Finite Element method is the common choice of analysis procedure because this method also relies on a discretised domain. Furthermore the output of the Finite Element method is element nodal displacement that along with element stiffness matrices can be used to calculate the strain energy of an element, which is the basis of evaluation of elements in the BESO method, used in this thesis. See chapter 4.

Robot Structural Analysis (ROBOT), a commercial finite element package available at the university, was explored as an option for structural analysis (Autodesk, 2011). Through the ROBOT Application Programming Interface (API) it is possible to gain access to the full operability of the software from within other software environments. This allows custom code to take advantage of the advanced features that ROBOT has to offer. Although this is encouraging a few drawbacks exist that make ROBOT a less appealing choice for the framework for the finite element analysis in this thesis. ROBOT does not possess the abilities needed to implement a custom volumetric optimisation algorithm. It is impossible to gain full control over the mesh definition and the scripting complex geometry is cumbersome.
Another commercial software option is ANSYS (ANSYS, 2011). This software package also allows control over the functionality of the software through a scripting interface and is specialised for analysis of solids. As the author has no experience with ANSYS and because it is estimated that a link between different software packages would be considerably slower that implementing the finite element method directly into the code, ANSYS is also not a good option for the purpose of this thesis.
Total control over the analysis and meshing can be achieved by implementing an existing package directly into the software. However Robot has been used to validate the finite element code implemented in FabricCast.
The majority of the open source finite element libraries available are based on the programming language C++ (Wikipedia, N. d.). However JAVA has been chosen as the programming language because of the reasons outlined previously.
Changing the programming language would require some extra time during the software development, as the author isn't familiar with the syntax of C++ programming.
A JAVA based finite element is presented by Nikishkov (2010), who describes the basics of the finite element method and the programming of algorithms used to create a working object oriented finite element library (JFEM). More important for the sake of this thesis is the fact that the source code explained is available as a free download. A free source code with a detailed documentation makes it easier to customise the method.

The reader is referred to literature for basic derivation of the finite element method used in the source code (Nikishkov, 2010).

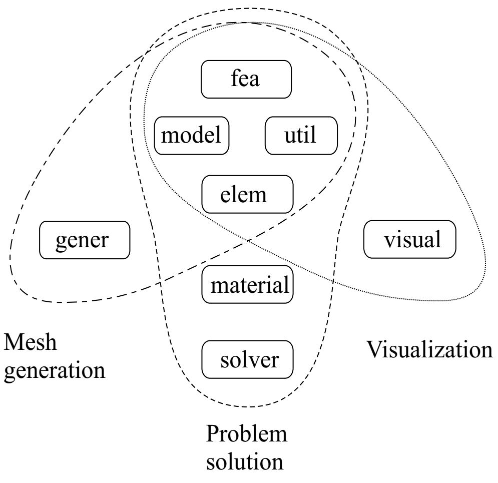


Figure 3.1: Overview of the packages included in the FE source code (Nikishkov, 2010)

The JFEM library consists of three main parts, a finite element mesh generator (pre-processor), a finite element solver (processor) and a visualiser (post-processor). The source code is split in to a variety of distinct packages of classes.

The classes compulsory to get the solver working as an integrated part of the software are included in the following classes: `fea`, `model`, `util`, `elem`, `material` and `solver`. See Figure 3.1.

Package fea	**Main classes**
Class FE	Symbolic constants
~~Class Jfem~~	~~Main class for solution of elastic and elastic–plastic problems (finite element processor)~~
~~Class Jmgen~~	~~Main class for mesh generation (preprocessor)~~
~~Class Jvis~~	~~Main class for visualization of models and results (postprocessor)~~
Package model	**Finite element model and loading**
Class Dof	Degree of freedom
~~Class ElemFaceLoad~~	~~Element face loading~~
FeLoad	Load increment for the finite element model
FeLoadData	Load data for the finite element model
FeModel	Description of the finite element model
FeModelData	Data for the finite element model
FeStress	Computing stress increment
Package util	**Utility classes**
~~Class FePrintWriter~~	~~Helper class for organizing printing to a file;~~
~~Class FeScanner~~	~~Scanning finite element data;~~
GaussRule	Several Gauss integration rules
UTIL	Printing error messages, dates, etc.
Package elem	**Finite elements**
Abstract class Element	Finite element
~~Class ElementQuad2D~~	~~Two-dimensional quadratic isoparametric element~~
~~Class ElementQuad3D~~	~~Three-dimensional quadratic isoparametric element~~
~~Class ShapeQuad2D~~	~~Two-dimensional quadratic shape functions and their derivatives~~
~~Class ShapeQuad3D~~	~~Three-dimensional quadratic shape functions and their derivatives~~
~~Class StressContainer~~	~~Stresses and equivalent strains at integration point~~
Package material	**Constitutive relations for materials**
Class ElasticMaterial	Constitutive relations for an elastic material
~~Class ElasticPlasticMaterial~~	~~Constitutive relations for an elastic–plastic material.~~
Package solver	**Assembly and solution of global finite element equation systems**
Abstract class Solver	Solution of the global equation system
Class SolverLDU	Profile LDU (lower, diagonal and upper matrix decomposition) symmetric solver
Class SolverPCG	Preconditioned conjugate gradient solver with sparse row format storage.

Table 3.2: Classes included in the software. Classes with strikethrough were not used in the software, but still exist to avoid compile errors. Short descriptions courtesy (Nikishkov, 2010)

Although some of the classes aren't needed to run the solver they have to be present to avoid compile errors. See Table 3.2 for an overview of the classes contained in the library. Figure 3.2 shows the dependencies between the different classes of the JFEM solver.

A detailed description of the functionality can be found in the literature (Nikishkov, 2010)

One of the main challenges of implementing the FE code is to translate the custom `Quad3D` object, described above, into a volumetric finite element model. All mesh connectivity information has to be converted to an object of the type `FeModel`, all constraints have to be converted into `Dof` and all loads have to be converted into `FeLoad` objects. A method in the `Quad3D` class converts the information stored in the object to the equivalent finite element formulation. This approach allows total control over the mesh class without making any changes to the finite element library. Information on the entire mesh is stored in the class and only the necessary information is passed to the `FeModel` object. For example the entire topology of the structure is stored but only enabled voxels are converted into elements for the structural analysis.

A few customised classes are added to the JFEM library to make it more suitable for this thesis. This is described in more detail in chapter 4.

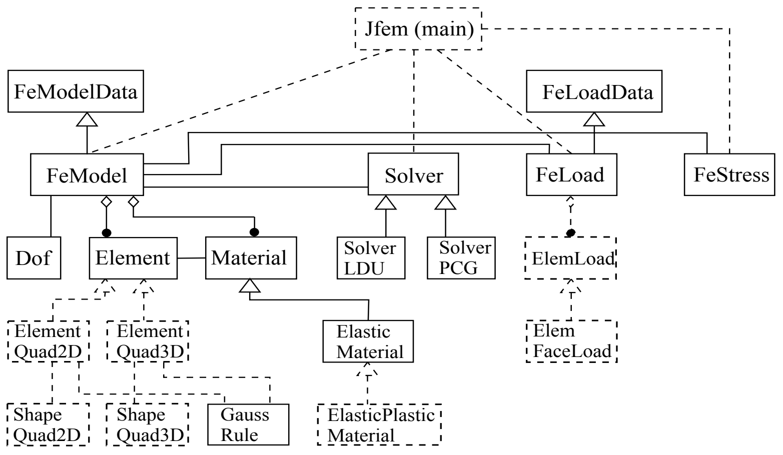


Figure 3.2: Dependencies of the finite element processor. The classes marked with dashed lines are not used by the software. Figure is from Nikishkov (2010), but edited by the author

FABRIC

As mentioned, two types of mesh topologies are implemented in the software. One is the volumetric mesh mentioned above and the other is a surface mesh topology representing the fabric (`Fabric`). The topology of the surface mesh is represented as a series of vertices, edges and faces containing references to connecting elements. Apart from describing the topology of the mesh this class also handles the calculation of internal and external forces for the dynamic simulation of the fabric.

The `Fabric` object can be passed to the `ProcessingRenderer` for a graphical representation. The object interacts with the `Quad3D` object through an intersection handler described later in this chapter. Interaction tools allow the user to directly interact with `Fabric` object. These tools are described in the next section.

INTERACTION

A few interactive tools are available in the software. Some of them are used to set up the design domain and boundary conditions. Others are used to manipulate the fabric object in real-time. A graphical representation of the interactive tools available to the user is shown in Figure 3.3.

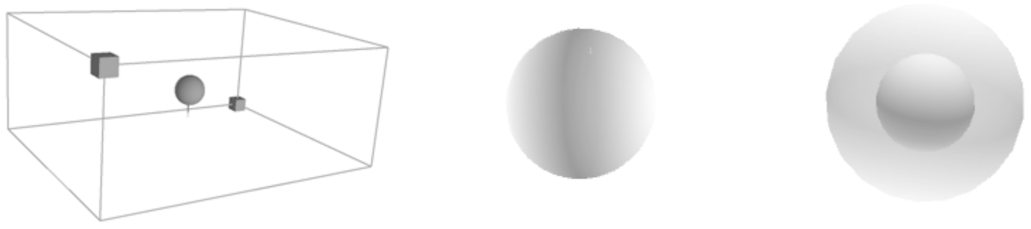


Figure 3.3: Interactive tools. From left to right: HandleBox, RepelingForce and Sphere

The `HandleBox` class allows the user to select voxels or vertices. By selecting these elements it is possible set up the design domain and boundary conditions using the GUI. The `HandleBox` is essentially a box that can be translated in space and resized by dragging the handles with the mouse. All elements inside the `HandleBox` are modified when for example loads are applied.
The `Sphere` class is a sphere that collides with the fabric object allowing the user to 'push' the fabric in to the desired shape. The `RepelingForce` class has a similar function, but instead of colliding with the fabric it adds a force to the vertices of the fabric in the direction of the vector between the vertices and the force with a magnitude of $\frac{1}{\left\|\mathbf{f}-\mathbf{p}\right\|^4}$, where $\mathbf{f}$ is the position vector of the force and $\mathbf{p}$ is the position vector of the vertex. Both the `Sphere` and the `RepelingForce` can be translated in space by clicking and dragging the mouse.

ProScene (2011) is a library developed for the Processing language. Included in the library are classes for interaction of Processing 3D scenes such as camera control and mouse interaction. This library was used as a basis for the interactive part of the software.
The `InteractiveFrame` class is a part of the ProScene library. By adding an instance of this class to a mouse grabber pool, intersections between the mouse pointer and the InteractiveFrame object can be detected on the fly. The class contains methods for geometric manipulation of the object such as translation and rotation by clicking and dragging the mouse. These transformations can be constrained to the world axes by using built-in functions of the class. By constraining all three axes the user can fix a point in space.
This type of object is attached as a property to the custom classes that the user needs to interact with. In the case of this thesis the `Vertex`, `HandleBox`, `Sphere` and `RepelingForce` classes contain an `InteractiveFrame` object.

Figure 3.4 shows a fabric object with InteractiveFrame objects attached at the vertices. The InteractiveFrame highlighted with green is intersecting the mouse pointer. On the right the InteractiveFrame is translated along the z-axis thereby applying a deformation. The fabric reacts to this deformation accordingly.

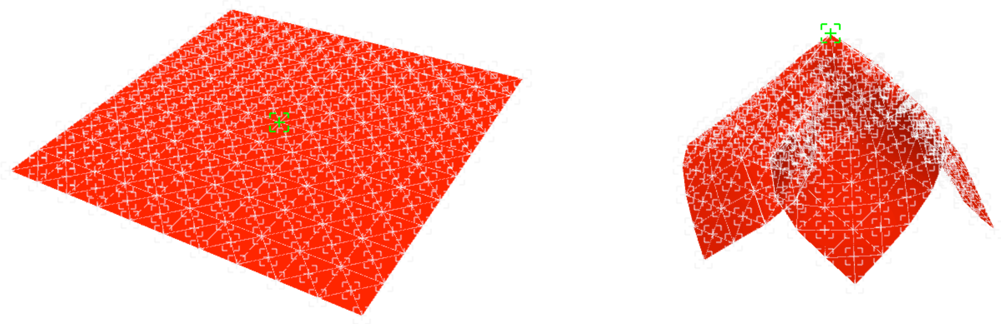


Figure 3.4: InteractiveFrame placed at vertex positions

COLLISION

The software contains a function allowing the user to interactively drape a fabric over the optimised topology and thereby determining the way the slab could be manufactured.
The class that combines the optimised topology with the surface mesh representing the fabric is called `FabricMeshIntersectionHandler`. By passing a `Quad3D` and a `Fabric` object to this class it can be determined whether an intersection has occurred. If an intersection has been determined the class corrects the vertex positions of the fabric to a position outside the volumetric mesh, thus simulating a collision between the fabric and the optimised topology.

UTILITIES

This class contains a number of static methods that allow the programmer to call methods of the class without initiating an object. This is useful when certain methods are used widely between the different classes. This is the case for mathematical functions such as the rounding and remapping of numbers.
The `Utilities` class also contains some methods for writing data to files on the hard drive. A simple method allows any string to be written as a text file. When the optimisation algorithm has converged this is used to write a summary of the optimisation process to a text file.
The `Quad3D` and the `Fabric` class can be output as a VRML file for further manipulation. The VRML file only contains topological information of the meshes. A VRML file is a common file format that can be read by various commercial modelling software packages such as Rhinoceros. (McNell, 2011)
Another method is available for saving the entire information stored in the Quad3D object to a file using JAVA serialisation. This allows the user to save meshes with all the information such as boundary conditions, topology and geometry etc. The files saved using this method can't be read by other software but allow an optimisation to be paused or restarted.

3.3 SOFTWARE STRUCTURE AND FLOW

The diagram in Figure 3.5 shows the dependencies of the classes in the software. Some classes are responsible for the storing and creation of other classes. For example the `Vertex` class doesn't have much use on it's own, but when combined with subclasses of the `Fabric` class it is suddenly a very useful object.

The figure also shows the relationship between the classes created by the author and the classes in the

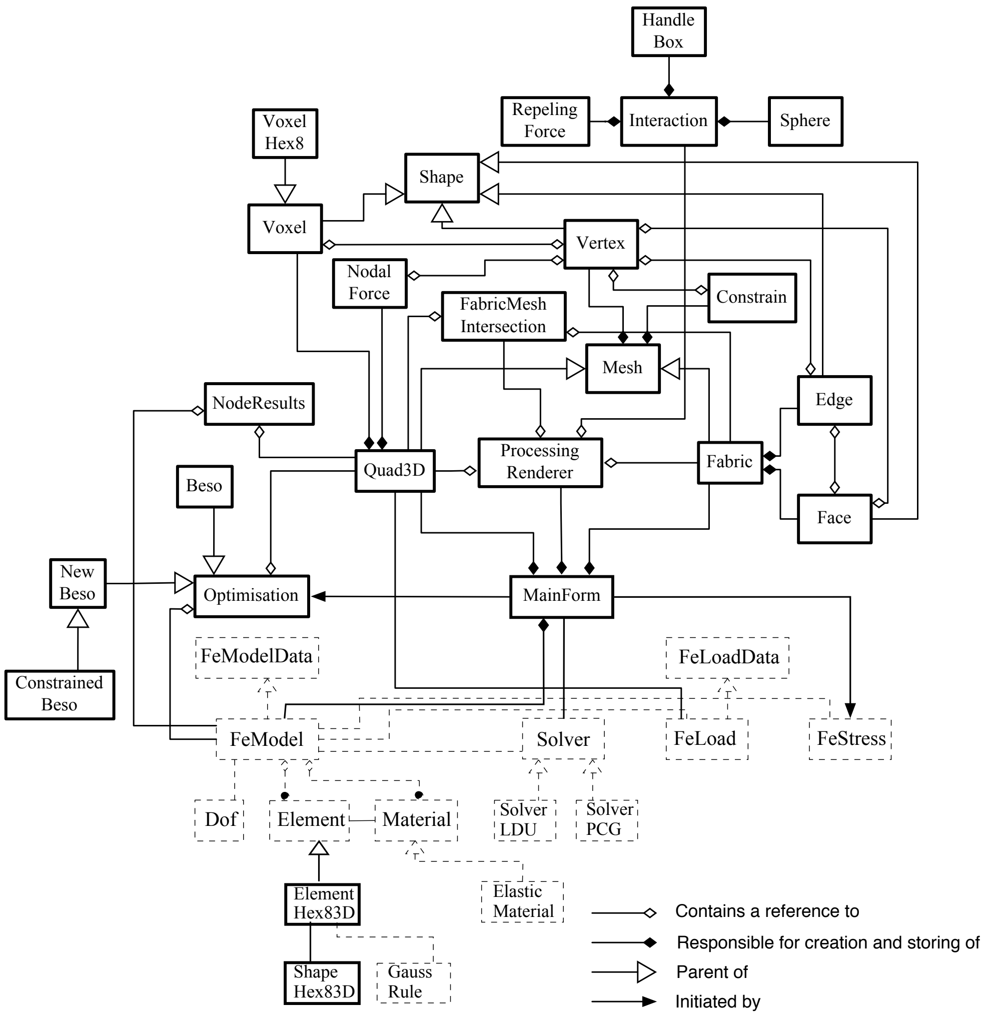


Figure 3.5: Dependency diagram of FabricCast

JFEM library (marked with dashed lines).

The GUI lets the user input different properties such as geometry, mesh, boundary conditions, materials and optimisation parameters. An option is also available to enable or disable voxels and define the design space for the optimisation. When the user has defined the appropriate parameters, the optimisation process is started.

The optimisation algorithm outputs a summary of the process and saves the Quad3D object to a file when a convergent state is reached.

Before the next stage of the process the user can input some additional parameters to the software, including fabric material properties, gravity and potential constraints. After these parameters have been defined, the draping process is started. During this process the user can interact with the fabric using the interaction tools mentioned previously.

When the desired shape of the fabric is found, the fabric can be exported to VRML file and imported into a commercial 3D modelling environment.

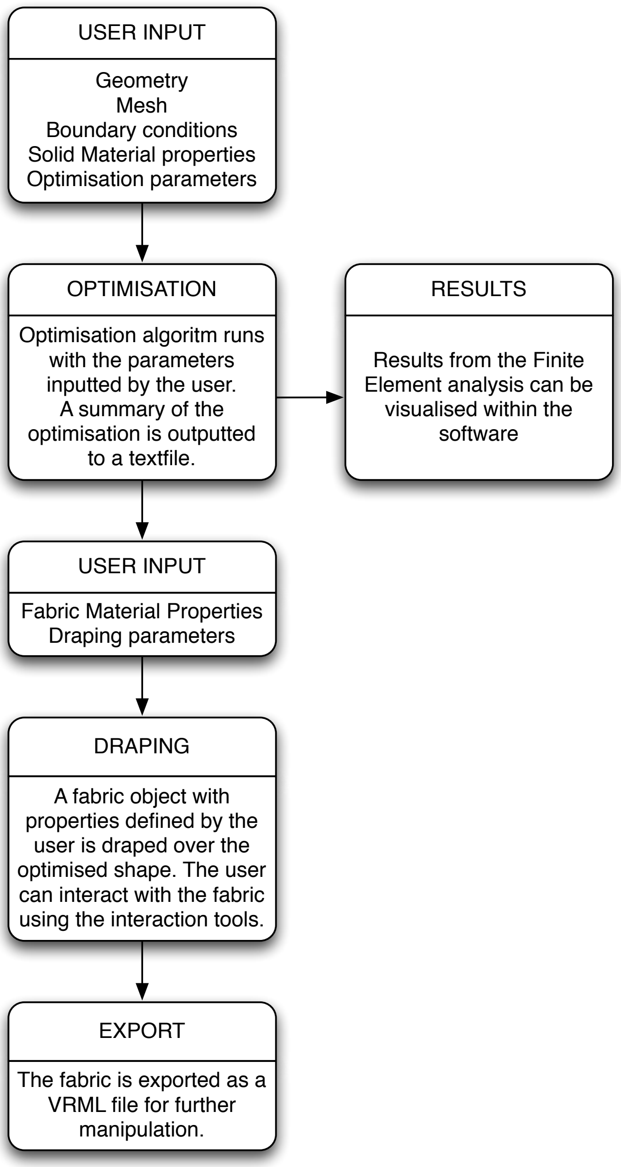

USER INPUT
Geometry
Mesh
Boundary conditions
Solid Material properties
Optimisation parameters
OPTIMISATION
Optimisation algoritm runs with the parameters inputted by the user.
A summary of the optimisation is outputted to a textfile.
RESULTS
Results from the Finite Element analysis can be visualised within the software
USER INPUT
Fabric Material Properties
Draping parameters
DRAPING
A fabric object with properties defined by the user is draped over the optimised shape. The user can interact with the fabric using the interaction tools.
EXPORT
The fabric is exported as a VRML file for further manipulation.

Figure 3.6: Software flow

CHAPTER 4 METHODS & IMPLEMENTATION

In this chapter all relevant methods used in the development of FabricCast are explained. Conceptual explanations are combined with descriptions of individual implementations in the software. A series of test was conducted on each method to validate that it had been correctly implemented and was performing as expected. These tests are also described in this chapter.

4.1 FINITE ELEMENT ANALYSIS

The method known today as the Finite Element method (FE) has its foundations in the field of aeronautics in the 1950's and its origin can be traced back to the work of Navier in the 1820's (Samuelsson and Zienkiewicz, 2006).
Today finite element procedures are indispensible in a huge variety of engineering disciplines and are used to analyse structures, solids and fluids.

METHOD OUTLINE

The finite element method is a technique for solving partial differential equations by discretising a continuous domain into finite elements with unknown nodal displacements. By subdividing the domain the infinite number of equations of the continuum is reduced to a finite number. A system of linear algebraic equations is assembled and solved to determine the unknown nodal values based on the applied boundary conditions. Values at arbitrary points outside of nodes are then determined using interpolating functions or so called shape functions. These shape functions are an approximation of the variation of values between the nodes. The finite element method provides good precision even when simple shape functions such as linear functions are used.
From the nodal displacement values the strains and stresses are calculated using the appropriate constitutive equations for the material in question.
The constitutive equations used in this thesis are based on the assumption of linear-elastic isotropic material. This assumption is far from the real-world behaviour of concrete, but as the main focus of this thesis is the manufacturability of optimised shape this assumption is considered valid.
The Finite Element source code implemented in this software is developed by Nikishkov (2010). In this section the addition of a new element to the source code is described. For further description of the source code the reader is referred to chapter 3 or literature (Nikishkov, 2010).

ADDING A NEW ELEMENT

In the following a quick outline of the method in continuum mechanics form is explained to summarise the important parts of the method and to identify the changes and addition to the JFEM library made by the author.
The equations that need to be solved for a continuum mechanics problem can be written in matrix notation as follows (Nikishkov, 2010)

$$[k]\{u\}=\{f\} \tag{1}$$

$$\{f\}=\{p\}+\{h\} \tag{2}$$

$$[k]=\int_V [B]^T [E][B]dV \tag{3}$$

$$\{p\}=\int_V [N]^T \{p^V\}dV+\int_S [N]^T \{p^S\}dS \tag{4}$$

$$\{h\}=\int_V [B]^T [E]\{\varepsilon^t\}dV \tag{5}$$

where $[k]$ is the element stiffness matrix, $\{u\}$ is a vector of unknown element node displacements and $\{f\}$ is the load vector consisting of actual forces $\{p\}$ and thermal forces $\{h\}$.

Matrix $[N]$ is a matrix containing the shape functions for the element.

$$[N]=\begin{bmatrix} N_1 & 0 & 0 & N_2 & \dots \\ 0 & N_1 & 0 & 0 & \dots \\ 0 & 0 & N_1 & 0 & \dots \end{bmatrix} \tag{6}$$

Matrix $[B]$ is called the matrix displacement differentiation matrix

$$[B]=\begin{bmatrix} B_1 & B_2 & B_3 & \dots \end{bmatrix} \tag{7}$$

$$[B_i]=\begin{bmatrix} \frac{\delta N_i}{\delta x} & 0 & 0 \\ 0 & \frac{\delta N_i}{\delta y} & 0 \\ 0 & 0 & \frac{\delta N_i}{\delta z} \\ \frac{\delta N_i}{\delta y} & \frac{\delta N_i}{\delta x} & 0 \\ 0 & \frac{\delta N_i}{\delta z} & \frac{\delta N_i}{\delta y} \\ \frac{\delta N_i}{\delta z} & 0 & \frac{\delta N_i}{\delta x} \end{bmatrix} \tag{8}$$

All the local element equations (1) can be assembled in to a global system of equations and the unknown nodal displacements can be obtained by solving a system of linear algebraic functions. Strain and stress values can be found using constitutive equations. The JFEM solver handles all this and only the element shape functions and their derivatives are left to be determined.

The JFEM library includes a single three-dimensional 20-node element (Figure 4.1b).

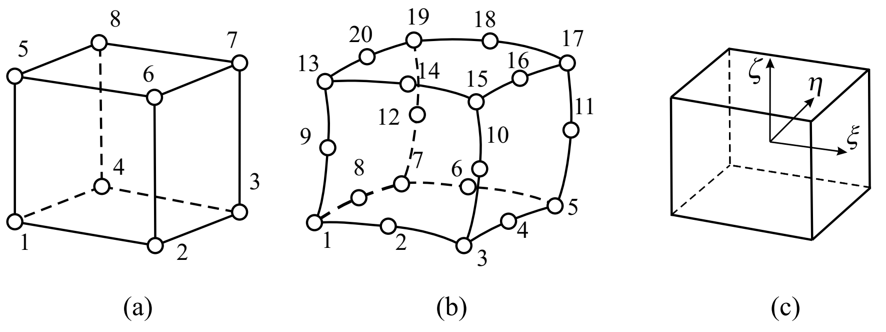


Figure 4.1: (a) Linear eight-node element. (b) Quadratic twenty-node element. (c) Local coordinate system. (Nikishkov, 2010)

This element has quadratic shape functions but by setting the mid-side node connectivity to 0 these can be reduced to linear functions. This effectively creates an 8-node element equivalent (Figure 4.1a). The JFEM solver ignores nodes with 0 connectivity thereby reducing the number of equations that need to be solved. This reduction greatly improves the speed of the calculations with the cost of accuracy in the interpolation functions.

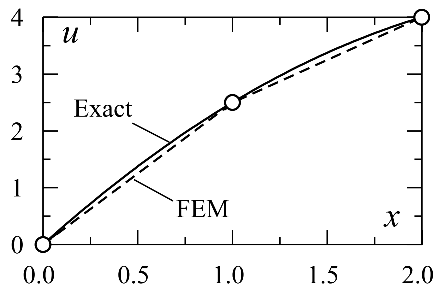


Figure 4.2: Comparison between finite element solution and exact solution of a system with two 2-node linear elements. (Nikishkov, 2010)

This loss of accuracy is preferable to the long computation times for the sake of this thesis. This loss of accuracy is illustrated in the example in Figure 4.2.
Another reason for choosing the eight-node element over the 20-node element is that the memory usage for a system of 20-node element quickly exceeds the memory capacity of the computer used for solving (the element stiffness matrix is 60 x 60). However setting the mid-side node connectivity to 0 does not remove the node from memory and the element matrices is still 60x60 in size. The only thing changing is the fact that the matrices contain more zero values than before and therefore the solver is faster. An 8-node element class using 24 x 24 matrices is implemented instead.

The new element that is implemented can be described as a three-dimensional isoparametric hexahedral linear eight-node element. The term isoparametric means that displacement fields and geometry are represented in parametric form. The parametric values can be interpolated to global coordinates using the shape functions. The shape functions are defined using the local coordinates ξ, η and ζ $(-1 \le \xi, \eta, \zeta \le 1)$ referring to the coordinate system in Figure 4.1c.

The term linear refers to the shape functions of the element. They can we written as follows

$$\begin{aligned} &N_i = \frac{1}{8}(1+\xi_0)(1+\eta_0)(1+\zeta_0), \\ &\xi_0 = \xi\xi_i, \\ &\eta_0 = \eta\eta_i, \\ &\zeta_0 = \zeta\zeta_i \end{aligned} \tag{9}$$

where ξ, η and ζ is the variable parametric coordinates and ξ_i, η_i and ζ_i is the parametric coordinates of the ith node.
The partial derivatives of the shape functions are as follows

$$\begin{aligned} &\frac{\delta N_i}{\delta\xi} = \frac{1}{8}\xi_i(1+\eta_0)(1+\zeta_0), \\ &\frac{\delta N_i}{\delta\eta} = \frac{1}{8}\eta_i(1+\xi_0)(1+\zeta_0), \\ &\frac{\delta N_i}{\delta\zeta} = \frac{1}{8}\zeta_i(1+\xi_0)(1+\eta_0) \end{aligned} \tag{10}$$

The shape functions and their derivatives are implemented in the class `ShapeHex83D`. Element methods for computing the element stiffness matrix and element strains etc. are written as part of the class `ElementHex83D`. These methods uses the same names as methods in other element classes and are called by the solver. The JFEM library remains untouched apart from the addition of one line in the abstract class `Element`, to incorporate the new element.
It is noted that the performance, considering both speed and memory usage, of the Finite Element analysis can be improved by implementing a solver customised for the type of topology presented in this thesis.

```
// Implemented element types
    static enum elements {
        quad8 {Element create() {return new ElementQuad2D();}},
        hex20 {Element create() {return new ElementQuad3D();}},
        hex8  {Element create() {return new ElementHex83D();}};
```

Figure 4.3: Line added to JFEM source code

VALIDATION

As mentioned earlier the structural analysis software ROBOT was used to validate the results of the finite element library. A simple cantilever beam with a quadratic cross-section was modelled in both programs with the dimensions 100x10x10 mm. The volume was discretised in to cubic elements with the dimensions 2x2x2 mm. A linear material with Young's modulus at 210000 MPa and Poisson's ratio at 0.3 was used. All nodes at the free end were loaded with a nodal force with a magnitude of 1000 N in the negative direction of the z-axis totalling a load of 36 KN. All nodes at the fixed end were constrained for displacement in all directions.
The displacement expected is calculated using a formula derived from the beam's differential equation.

$$\delta = \frac{1}{3}\frac{PL^3}{EI}$$

where P is a point load at the tip of the cantilever beam, L is the length of the beam, E is Young's modulus and I is the moment of inertia.
Using the values from above the displacement is

$$\delta = \frac{1}{3}\frac{1000N\cdot(100mm)^3}{210000Mpa\cdot\frac{1}{12}\cdot 10mm\cdot(10mm)^3} = 68.57mm$$

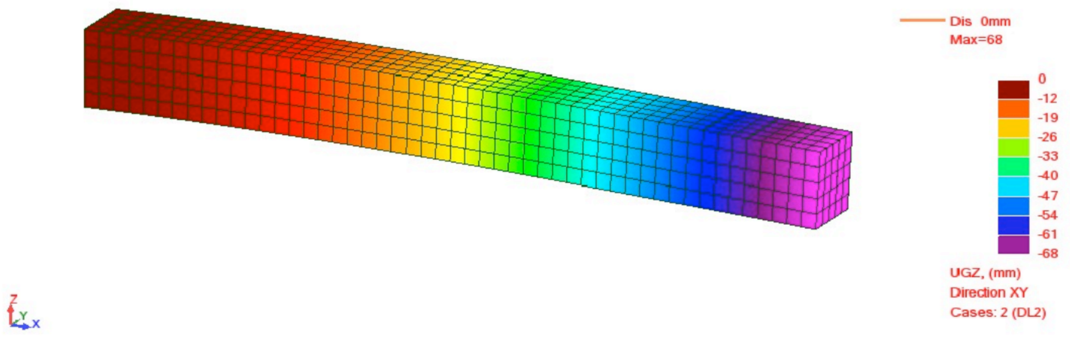


Figure 4.4: Deformed cantilever with colour map from ROBOT

Analysis	**UX**		**UY**		**UZ**	
	min	**max**	**min**	**max**	**min**	**max**
ROBOT	-5,1	5,1	-0,13	0,13	-68,28	0,0
FabricCast	-5,01	5,01	-0,14	0,14	-67,05	0,0

Table 4.1: Global displacement of cantilever in mm

Figure 4.4 and Figure 4.5 shows the deflected shape of the cantilevered beam from ROBOT and FabricCast respectively. In both cases the deflected shape corresponds to what is expected. The deflection values in the direction of the world axes are displayed in Table 4.1.
The deviation from the exact value calculated using the differential equation of the beam is due to the

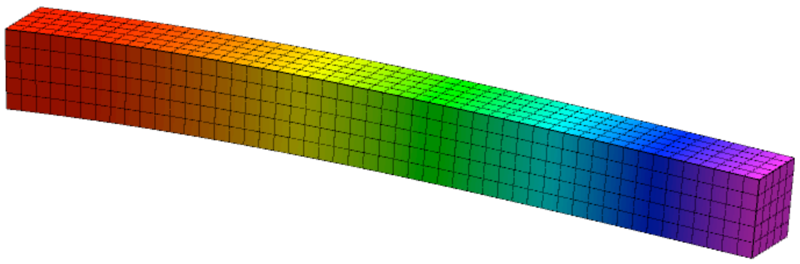


Figure 4.5: Deformed cantilever from FabricCast with colour map

fact that the solid model takes material contraction in to account. The fact that the load is evenly distributed over the cross-section of the beam oppose to a single point load can also contribute to the deviation of results.
The finer the mesh gets the smaller this deviation is.
The small diversion in results between ROBOT and FabricCast is due to some numerical differences in the methods. All though Robot uses same volumetric elements and shape functions floating point errors, different solving and integration methods can cause these small deviations.
As the deviation gets smaller with mesh refinements it is decided that FabricCast produces reliable results.
The displacement values from FabricCast is hereby validated and as the displacements are the results needed for the topology optimisation algorithm to work this is enough to declare that a working FE library is implemented.

4.2 OPTIMISATION ALGORITHM

The aim for topology optimisation algorithms is to optimise the layout of a given volume of material within a predefined design domain with certain boundary conditions. The optimal layout depends on the objective function set by the designer. The optimal design might be a maximum stressed design, a

design with optimal serviceability limit state performance or a design where certain natural frequency of the structure is avoided. A design using one objective function might not be the optimal using another and for example stresses can reach an unacceptable limit before the optimum of a maximum stiffness design is reached. In these cases a constraint function can also be set by the designer to avoid a design with stresses beyond the limit of the material for example.
For simplicity no constraint function has been set as the focus of this thesis is the manufacturability of the shapes not the actual optimisation of concrete shapes.

METHOD OUTLINE

The optimisation algorithm used in this thesis (BESO) is a simple algorithm that evaluates the elements of a discretised design domain and decides where the material is needed based on element sensitivity numbers. These numbers are calculated based on the strain energy of an element and its surrounding elements.
As briefly explained in the chapter 2 the BESO algorithm is an algorithm that unlike the ESO algorithm allows material to be added back in as well as removed. This ensures that material that has been prematurely removed can be added back to the structure and it allows the existing material to rearrange when a sudden change in topology occurs, i.e. member degeneration.
The most comprehensive and current description of the BESO method is by Huang and Xie (2007), as described in the following.

The objective of the optimisation is to minimise the mean compliance thus achieving a solution with optimal stiffness. The compliance is a measurement for how flexible a structure is and is defined as the outer work of a structure. The higher the mean compliance the more flexible the structure is.
Compared to other optimisation techniques where material parameters, such as density, are the design variables, the BESO algorithm treats the element itself as the design variable. The change in the mean compliance or total strain energy of the structure is equal to the strain energy of the element acting as the variable. It is yet to be proved that using strain energy as an optimisation criteria leads to an optimal design, but recent research show that numerically the method generally finds an optimal solution (Edwards et al., 2007).
The element can assume the values 0 or 1 denoting if the element is present in the structure. The mathematical definition of the optimisation process is as follows

Minimize $$C = \frac{1}{2}\mathbf{f}^{\mathrm{T}}\mathbf{u}$$

Subject to: $$V^* - \sum_{i=1}^{N} V_i x_i = 0,$$
$$x_i \in \{0,1\}$$

In this formulation C is known as the mean compliance $\mathbf{f}$ is the applied load vector and $\mathbf{u}$ is the displacement vector. V^* is the prescribed total structural volume and V_i is the volume of the *ith* element whereas x_i is a variable that can assume the value 0 or 1 declaring the absence or presence of an element.

To further evaluate the performance of a design a factor called the performance index is introduced as

$$PI = \frac{1}{C\sum_{i=1}^{N} V_i} \tag{11}$$

The factor evaluated the total mean compliance and the total volume of the structure. The higher the performance index the better the structure performs.

SENSITIVITY VALUES

To evaluate whether an element is applicable for removal, a value called the sensitivity number is calculated. This number represents the change in the mean compliance when the element is removed and is equal to the strain energy of the element. This value can be calculated directly from results of the finite element formulation

$$\alpha_i^e = \frac{1}{2}\{u_i\}^T\{K_i\}\{u_i\} \tag{12}$$

where u_i is the ith element displacement vector and K_i is the element stiffness matrix. The superscript e denotes that the value refers to an element.
For void elements the sensitivity value is initially set to 0.
As in the case of the ESO method this number is enough to decide whether an element should be removed. For the BESO algorithm to work the sensitivity values of the void elements that are not involved in the FE analysis need to be determined.
The first step of this process is to calculate the sensitivity numbers of a node by averaging the sensitivity values of elements connected to it as follows

$$\alpha_j^n = \frac{\sum_{i=1}^{M} V_i \alpha_i^e}{\sum_{i=1}^{M} V_i} \tag{13}$$

where the superscript n denotes that the value refers to a node. M is the total number of elements connecting to the jth node.
The element values are calculated from the nodal values using a mesh independency filter that also serves the purpose of smoothing the sensitivity values thus reducing the occurrence of checkerboard patterns caused by discontinuities in sensitivity numbers across element boundaries.
The mesh independency filter works by identifying nodes that influence the sensitivity number of an element. Only nodes within a certain distance of the element are used to calculate the element

sensitivity number. This distance is denoted $r_{\min}$ and can in 2D be visualised as the radius of the circle with centre in the centroid of an element creating the subdomain Ω_i. See Figure 4.6.
The influence of the nodes on the elements within this domain is based on a linear weight factor.

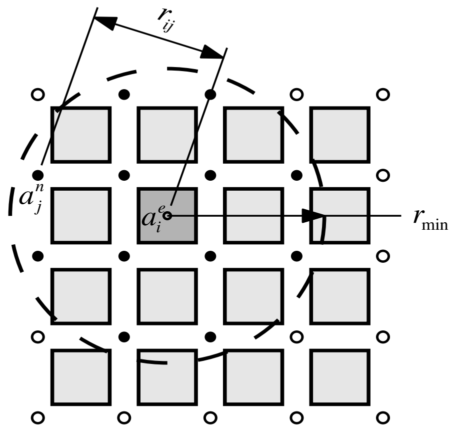


Figure 4.6: Sensitivity value smoothing

The element sensitivity numbers with the mesh independency filter is calculated as

$$\alpha_i^e = \frac{\sum_{j=1}^{M} w(r_{ij})\alpha_j^n}{\sum_{j=1}^{M} w(r_{ij})} \tag{14}$$

where M is the total number of nodes connected to the element and $w(r_{ij})$ is the weight factor defined by

$$w(r_{ij}) = r_{\min} - r_{ij} \tag{15}$$
$$(j = 1,2,...,M)$$

where r_{ij} is the distance between the node and the centroid of the ith element and the jth node. Huang and Xie (2007) note that $r_{\min}$ should be at least half the size of an element. This is a bit confusing statement, as it isn't clarified how the size should be measured. If the size of the element is set as the minimum dimension of the element and if $r_{\min}$ is set to half this size then the subdomain defined by $r_{\min}$ (an inscribed circle) would not contain any nodes and the sensitivity value would be 0. Huang and Xie further recommend that $r_{\min}$ should be chosen between 1-3 times the size of an element. It noted that although $r_{\min}$ can be chosen freely it should not change with mesh refinement so $r_{\min}$ should be chosen considering the biggest possible element in the design.
In an earlier version of the BESO method the averaged nodal value was linearly extrapolated to nodes surrounding a void element, thus determining the element sensitivity values by averaging the extrapolated values surrounding void elements (Huang et al., 2006). This procedure caused a problem

of mesh dependency, where more voids occurred in a solution with same boundary conditions but a denser mesh.

A further improvement to the accuracy of the assessment of sensitivity values to the number of void elements can be achieved by considering the history of sensitivity numbers for each element, and averaging the current value with that of the previous iteration as follows

$$\alpha_i = \frac{\alpha_i^k + \alpha_i^{k-1}}{2} \tag{16}$$

where α_i is the sensitivity value of the ith element and k is the iteration number.

The smoothed sensitivity number for all the elements including the void elements can thus be determined.

ADDING AND SUBTRACTING ELEMENTS

The next step is to decide the volume of the next iteration step thus deciding how many elements should be added or removed. Because of the bi-directional nature of the algorithm the number of elements to remove can either be positive or negative. In other words the volume can increase or decrease at each step of the iterative process until the objective volume is reached. The current volume is first compared to the objective volume thus deciding whether to increase or decrease the volume for the next iteration. The volume for the next iteration is calculated as,

$$\begin{gathered} V_{k+1} = V_k(1 \pm ER) \\ (k = 1,2,3,...) \end{gathered} \tag{17}$$

where k is the iteration number and V is the total volume. The variable ER is called the Evolutionary Rate factor and decides the speed of the algorithm. ER should be kept low to avoid an optimisation process where too much material is removed with each iteration thus creating a non-optimal design. When the objective volume is reached, ER is set to 0 for the remaining number of iterations until convergent solution is found.

When all sensitivity numbers have been calculated the values are sorted from highest to lowest. Solid elements that satisfy

$$\alpha_i \leq \alpha_{del}^{th} \tag{18}$$

are then deleted, and void elements satisfying

$$\alpha_i > \alpha_{add}^{th} \tag{19}$$

are added.

At this point in the procedure, Huang and Xie (2007) introduces a factor called the admission volume ratio (AR) to insure that not too many elements are added at a single iteration. A fulfilling argument for the introduction of this factor is not given, but an explanation could be that the sensitivity number of the void elements that are candidates for being added to the structure is still based on an approximation and the more elements that are added at each iteration the more likely it is that the solution diverges from the optimal or loses its integrity.
The admission ratio is defined as the number of elements added divided by the total number of elements in the design domain.
If AR is less than or equal to $AR_{\max}$, which is a user defined maximum volume, addition then the thresholds is set to

$$\alpha_{del}^{th} = \alpha_{add}^{th} = \alpha_{th} \tag{20}$$

where the threshold value a_{th} corresponds to the volume for the next iteration. If n elements are required for the next iteration the threshold value is defined as the nth sensitivity value in the sorted list of sensitivity values for all elements.
If $AR < AR_{\max}$ then the number of elements n_{add} to be added is found by multiplying $AR_{\max}$ by the total number of elements in the design domain. The sensitivity values of the void elements are sorted and the threshold for addition of elements α_{add}^{th} is the sensitivity number ranked just below the value corresponding to the index n_{add} in the sorted list of void elements. The threshold for deletion of elements is determined such that the following equation is satisfied

$$V_{del} = V_{k+1} - V_k + V_{add} \tag{21}$$

where V_{del} and V_{add} are the deleted volume and added volume respectively. It is noted that in Huang and Xie (2007) a typographical error has transposed the factors V_k and V_{k+1} and this has been corrected in the implementation by the author.

This rather confusing method can best be illustrated with a small example.

Example

A design domain contains of 1000 elements of which 100 are void. In this example N denotes the total number of elements in the design domain.
All elements are assumed to have a volume of 1. The maximum volume addition ratio is set to $AR_{\max} = 1\%$ and $ER = 0.0555$. The objective volume is 500 elements therefore ER should be subtracted in equation (17)

$$V_{k+1} = V_k(1 \pm ER) = 900 \cdot (1 - 0.0555) \simeq 850$$

Therefore the domain should contain 850 solid elements in the next iteration step.
If the next iteration step calls for an addition of 10 ($N \cdot AR = 1000 \cdot 0.01 = 10$) or less elements the threshold is set using (20), if not, the admission volume ratio is bigger than the maximum value.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			98	99	100
98	98	96	88	75	74	51	44	34	23	22	20	17	16	15			2	2	1

Figure 4.7: Example: sorted sensitivity values of 100 void elements

The 100 values of the void elements are sorted from highest to lowest $(\alpha_1^{void}, \alpha_2^{void}, \alpha_3^{void}, ..., \alpha_{100}^{void})$ (Figure 4.7). The threshold for addition is calculated as

$$\alpha_{add}^{th} = \alpha_{(AR_{\max}N)+1}^{void} = \alpha_{(0.01 \cdot 1000)+1}^{void} = \alpha_{11}^{void} = 22$$

All void elements with a sensitivity value above this threshold are "switched on".
Next step is to determine the threshold for removal by using equation (17)

$$V_{del} = V_{k+1} - V_k + V_{add} = 900 - 850 + 10 = 60$$

1	2	3			836	837	838	839	840	841	842	843	844			998	999	1000
98	98	96			51	44	34	23	22	20	17	16	15			2	2	1

Figure 4.8: Example: sorted sensitivity values of 900 solid elements

Thus the threshold for removal should ensure that 60 element are removed from the structure.
The solid elements are sorted from highest to lowest (Figure 4.8) and the threshold is

$$\alpha_{del}^{th} = \alpha_{900-60}^{solid} = \alpha_{840}^{solid} = 22$$

All solid elements below this threshold are "switched off".

This approach of finding threshold values is a bit cumbersome and another approach has been used in the algorithm implemented in FabricCast. This is described in the next section.

The iterative process of finite element analysis and element removal continues until the objective volume has reached a convergence criterion. The criterion is defined as,

$$error = \frac{\left|\sum_{i=1}^{N}(C_{k-i+1} - C_{k-N-i+1})\right|}{\sum_{i=1}^{N} C_{k-i+1}} \leq \tau \tag{22}$$

where k is the current iteration number C is the mean compliance, N is an integral number set to 5 (Huang and Xie, 2007) insuring stable compliance over 10 successive iterations and τ is the residual defined by the user.

CUSTOMISATION OF ALGORITHM

A few minor alterations were made to the method outlined in the previous section. In this section these alterations are explained.

Because of the manufacturing restrictions inherent in the fabric-formed concrete, only removal of elements from the external bottom face of the structure is allowed. In other words elements with no solid element directly connected to them in the negative direction of the z-axis are candidates for removal. This ensures that no internal voids are created in the structure allowing the shape to be cast with fabric formwork. Similarly void elements are only eligible for addition if they have a solid element directly above them, which is in the positive direction of the z-axis. These elements are tested in every optimisation cycle and are from this point on referred to as 'candidates'. Figure 4.9 illustrates the candidates in a two-dimensional domain.
Void elements about to be added back are also checked an extra time because the element above could have been removed in the same iteration step after the list of candidates was created.

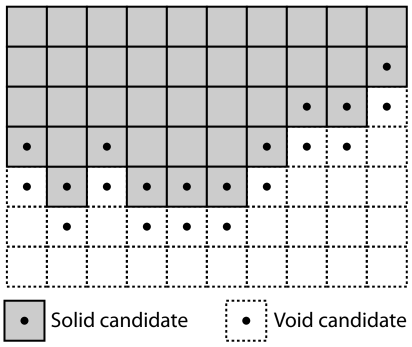


Figure 4.9: Candidates

A strategy based on removal of a certain number of elements instead of removing elements below a threshold is implemented because of the slightly confusing threshold determination in the original BESO method. This logic applies well to the idea of candidates, as the number of candidates varies with every iteration step.
At the beginning of the optimisation process the number of elements in the final design is calculated as

$$N^{obj} = N^{*}RRV \tag{23}$$

where N^{obj} is the objective number of elements, RRV is the removal rate of volume and N^{*} is the total number of elements in the design domain.

Next the number of elements in the structure for the next iteration step is calculated as

$$N_{k+1} = N_k(1 \pm ER) \tag{24}$$
$$(k = 1,2,3,...)$$

The number of elements to be removed can now be decided simply by subtracting the number of elements in the next iteration by the number of elements in the current

$$N^{del} = N_k - N_{k+1} \tag{25}$$

It is noted that N^{del} can also have a negative value, indicating that elements should be added to the current design. To insure the integrity of the design, N^{del} cannot be higher than a predefined value

$$N^{del}_{\max} = N^{solid}DR$$

where N^{solid} is the number of solid elements in the candidate list and DR is a deletion rate factor set by the user considering the design layout and mesh refinement. This is illustrated in the following example A reasonable value for DR acquired by numerical experience is 0.1.

Example

A 10x10x10 mesh has 1000 elements and 100 candidates for removal in the first iteration. ER in equation (24) is set to 0.1 then the number of elements to be removed is calculated using equation (24) and (25) as

$$N^{del} = N_k - N_k(1 - ER) = 1000 - 1000(1 - 0.1) = 100$$

In this case all of the candidates are removed and no real evaluation of the sensitivity number has taken place.

The example is a rare case because ER is usually set to a lower value than 0.1, but nonetheless this is a point of concern and even for lower values of ER this could drastically influence the design.

At this point the candidates are sorted by sensitivity value (lowest to highest). The number of void elements (N^{void}) is calculated and candidates that satisfy the following statement are "switched off"

$$i \leq N_{k-1}^{void} + N_k^{del} \tag{26}$$

where i is the index number of the candidate in the sorted list and k is the iteration number. The elements satisfying

$$i > N_{k-1}^{void} + N^{del} \tag{27}$$

are "switched on". Some of the elements satisfying equation (26) may already be void and therefore cannot be switched off but the correct number of elements in the next iteration should be correct following this method. An analogue argument applies to solid elements satisfying equation (27).

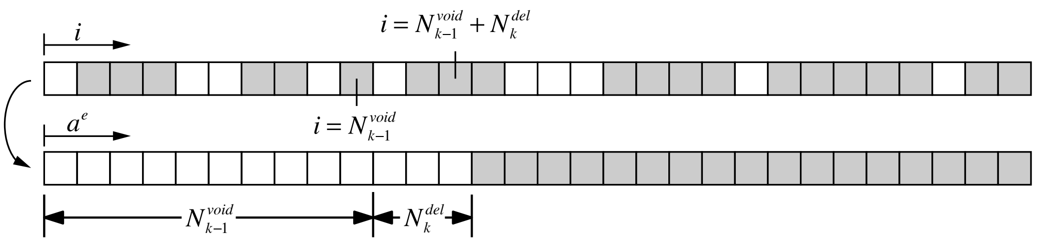


Figure 4.10: Toggling sorted candidate voxels

An issue, with both the original and the customised algorithm, that to be resolved was discovered during different tests. Because of the bi-directional nature of the algorithm an oscillating state occurs where the objective will be passed continuously if ER is set too high. On some occasions the algorithm never settles and a convergent stage is never reached (Figure 4.11 left).
There are a few ways to make sure this doesn't happen. One approach is to allow a certain deviation from the objective volume and setting ER to 0 when the volume is within a certain percentage of the required volume. Another is to decrease ER whenever the objective volume has been passed and thereby damping the oscillation until the objective volume is reached (Figure 4.11 right). This second approach was added to the algorithm in FabricCast.

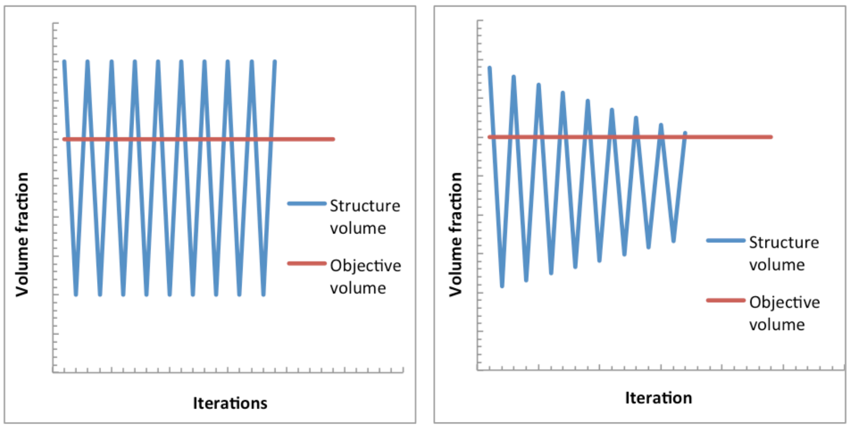


Figure 4.11: Oscillation state without (left) and without (right) damping

The entire optimisation algorithm is illustrated as a flowchart in Figure 4.12

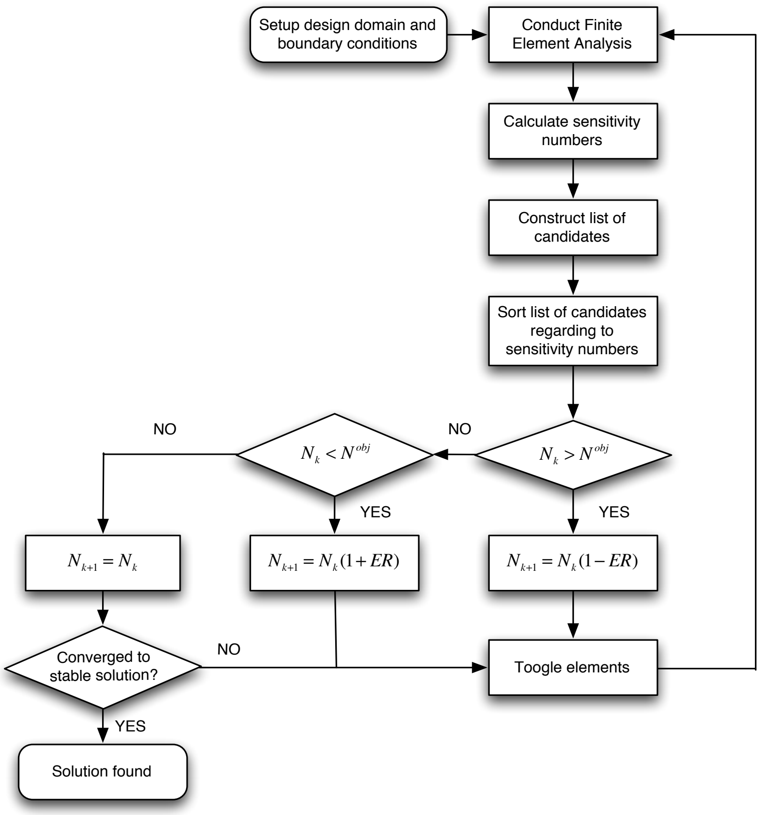

Setup design domain and boundary conditions
Conduct Finite Element Analysis
Calculate sensitivity numbers
Construct list of candidates
Sort list of candidates regarding to sensitivity numbers
NO
$N_k > N^{obj}$
YES
NO
$N_k < N^{obj}$
YES
$N_{k+1} = N_k$
$N_{k+1} = N_k(1+ER)$
$N_{k+1} = N_k(1-ER)$
Converged to stable solution?
NO
YES
Toogle elements
Solution found

Figure 4.12: BESO flowchart

VALIDATION OF CUSTOMISED ALGORITHM

This section contains a few cases that validate the correct implementation of the optimisation algorithm in the software. In the first two cases the optimised design is compared to tests performed by Huang and Xie (2007). In the last the algorithm that only allows the removal of bottom elements is used. As there doesn't exist any examples using this strategy the validation is done by presenting a couple of performance graphs.

Case 1 is a simple two dimensional cantilever plate with a single point load of 100 N at the free end (Figure 4.13). The material has a Young's modulus of 100000 MPa and a poisons ratio of 0.3.

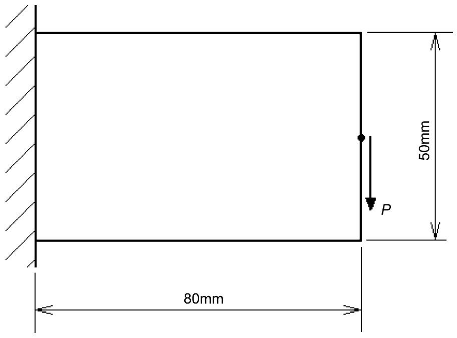


Figure 4.13: Case 1 (Huang and Xie, 2007)

Optimisation parameters

r_{min} = 3 mm
RRV = 0.5
ER = 1%
τ = 0.01%

The structure in Figure 4.14 is modelled in 2D, using 4-node surface elements and the one in Figure 4.15 is modelled in 3D, using 8-node volumetric elements, since there are no surface elements implemented in FabricCast.
Although one is a two-dimensional case and the other a three-dimensional case the resulting optimal topology should be the same. The 2D plate is modelled with a 32x20 mesh and the optimisation

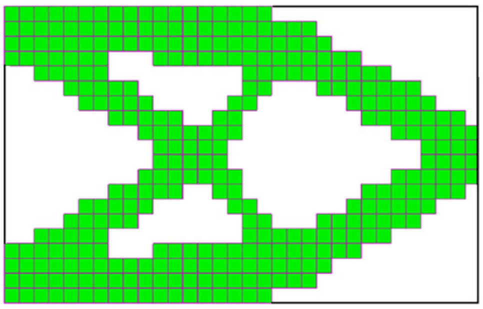


Figure 4.14: Benchmark topology (Huang and Xie, 2007)

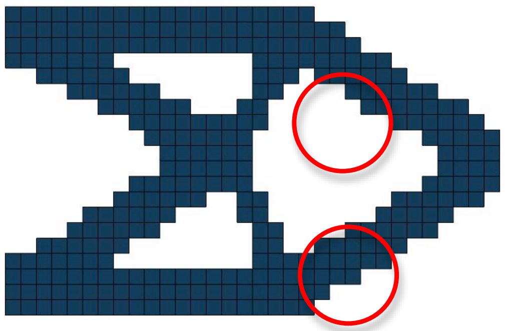


Figure 4.15: Topology from FabricCast

process starts with a full design domain of 640 elements. The 3D structure is modelled with a 32x20x1 mesh.
Figure 4.14 shows the optimised topology of the cantilever plate using the original BESO method. Figure 4.15 shows the optimised topology using the customised algorithm.

The topologies are basically the same but the original method seems to produce a topology that is symmetric around the central axis. This is not the case for the customised algorithm, as highlighted with red circles in Figure 4.15. This is due to the fact that the number of voxels removed using the original method is decided based on a threshold value rather than a forced number. When forcing the algorithm to remove a certain number, voxels with the same sensitivity value may or may not be removed based on the order in which they are evaluated. Overall the customised algorithm performs well and the difference between the two is more of an aesthetic nature.

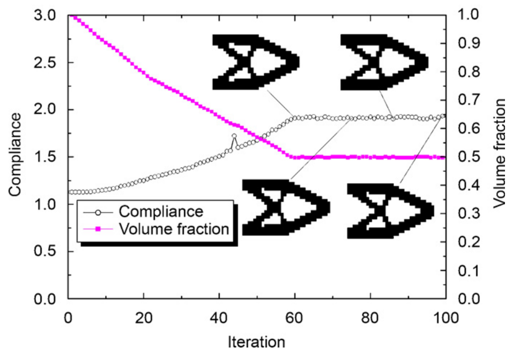


Figure 4.16: Performance graph (Huang and Xie, 2007)

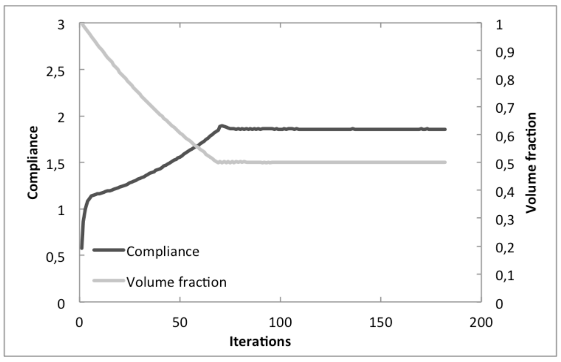
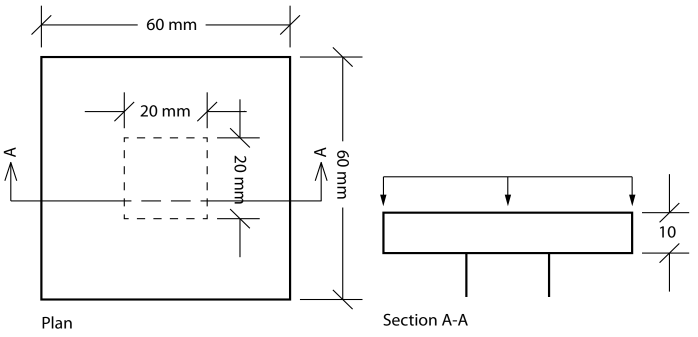


Figure 4.17: Performance graph from FabricCast

Figure 4.16 shows the development of the topology and the corresponding volume fractions and mean compliance using the original BESO. The corresponding graphs from the customised algorithm can be seen in Figure 4.17. They show a steady increase of the mean compliance as elements are gradually removed. Both methods reach the objective criterion after approximately 60 iterations where the removal of volume stops and the algorithm searches for converged state. There is a good coherence between the two graphs.

Case 2 is using the version of the algorithm where the removal of elements is restrained to a list of candidates. In this example the candidates qualified for removal are all solid elements that have a free bottom face. The candidates eligible to be added back are void elements that have a solid voxel connected to the top face. The structure is a slab supported by a central column with an evenly distributed load on the top. The material of the slap has a Young's modulus of 100000 MPa and a Poisson's ratio of 0.3. Because of symmetry only a quarter of the structure was modelled with mesh of 15x15x5.



Optimisation parameters

r_{min} = 3 mm
RRV = 0.5
ER = 0.5%
DR = 10%
τ = 0.01%

Figure 4.18: Case 2

Figure 4.19 shows the optimised topology. The tendency to produce an asymmetric result is also present in this case. Chapter 5 deals with this issue.
The performance graphs in Figure 4.20 show a similar development in both volume fraction and compliance to case 1.
Figure 4.21 shows the development of the performance index. The drop of performance index in the first few iteration steps is due to the default element sensitivity values for the first iteration. The rest of

the graph shows an increasing performance index, and therefore a gradually improving design towards the objective volume.

The cases presented in this section show that a working optimisation algorithm has been implemented and the performance is as expected.

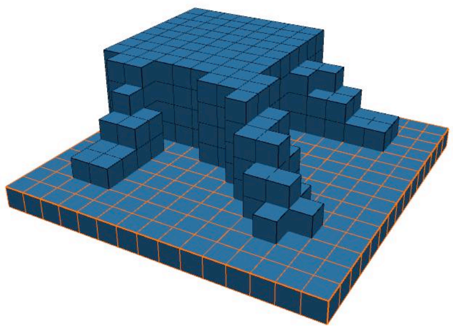


Figure 4.19: FabricCast topology turned upside down.

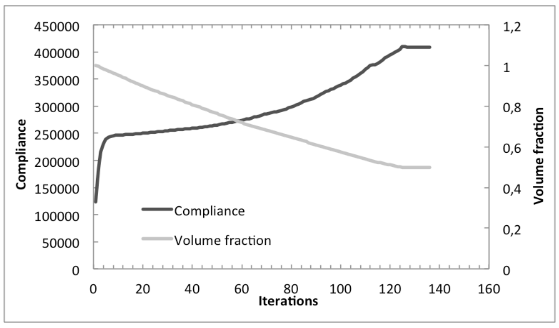


Figure 4.20: Performance graph from FabricCast

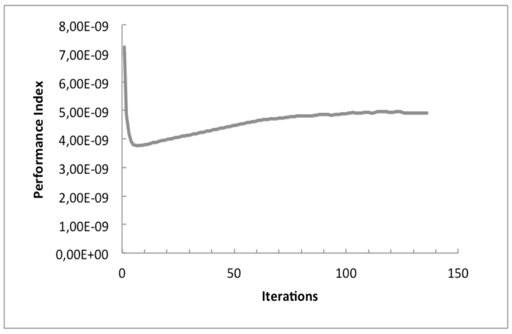


Figure 4.21: Performance index

4.3 CLOTH SIMULATION

This section contains a description of the methods used to simulate the behaviour of fabric draped over an object under the influence of gravity.

The equations of damped structural motion and constitutive equations of elasticity are used to determine the behaviour of a fabric in real-time. This simulation can be described as a form-finding process. Beginning with an initial arbitrary shape where motion is caused by imposing loads and stresses at the nodes, a formfound shape occurs when the shape comes to rest in an equilibrium state.

In this thesis the aim is to simulate a fabric draping over an object. By modelling the mechanical properties of the fabric and applying the correct forces to the nodes it is possible to simulate how fabric behaves under the influence of external forces, for example gravity.

INTEGRATING NEWTON SECOND LAW OF MOTION

Newton's second law of motion can be used to describe the dynamics of diverse physical systems and objects. In the case of this thesis it is used to simulate the dynamic behaviour of cloth. By relating the object's mass m with the acceleration a the applied forces f can be calculated as

$$f = m \cdot a \tag{28}$$

or in differential form

$$f = m\frac{d^2x}{dt^2} \tag{29}$$

where x is the position of the node and t is time.

Because this equation involves a second derivative of time it is a second order differential equation.

This equation can be reduced to a first order equation by the introduction of the variable v

$$v = \frac{dx}{dt} \tag{30}$$

$$\frac{f}{m} = \frac{dv}{dt} \tag{31}$$

Many methods are available for numerically integrating this type of equation and some of them are described in chapter 2. They all vary in precision, stability and computation time and the choice of the method have to be made according to problem that needs to be solved (Volino and Magnenat-Thalmann, 2001).

In this thesis the explicit Verlet integration method has been used. This method provides a medium accuracy with low computational effort, but more importantly it has a high stability when simulating under-damped systems such as particle systems. The algorithm works by incrementally stepping through time, calculating new positions based on the acceleration and the previous positions of the element.

There are basically three versions of the Verlet integration method, the basic method, the leapfrog method and the velocity method. The equation for the basic Verlet method is as follows

$$x_{t+\Delta t} = 2x_t - x_{t-\Delta t} + a_t \Delta t^2 \tag{32}$$

where x_t, $x_{t+\Delta t}$ and $x_{t-\Delta t}$ represents the current position of the element , the position in the next step and the position in the previous step respectively. Δt denotes the time-step and a_t the current acceleration.

A form of viscous damping can be implemented in this representation by introducing the damping coefficient ξ and rewriting the Verlet equation to

$$x_{t+\Delta t} = x_t + (1-\xi)(x_t - x_{t-\Delta t}) + a_t \Delta t^2 \tag{33}$$
$$(0 \leq \xi < 1)$$

Since an equation of motion is being integrated a_t can be determined from

$$a_t = \frac{f_t}{m} \tag{34}$$

Apart from the initialisation stage of the simulation where the previous position is unknown (Jiang, 2010), the only unknown variable at this point is the force.

MATERIAL MODEL

The force consists of contributions from internal and external forces such as gravity and other influences from the surroundings. The external forces are easily defined. Internal forces include tension, compression, bending, and shear forces within the cloth and are harder to define. To accomplish this, a material model is needed.

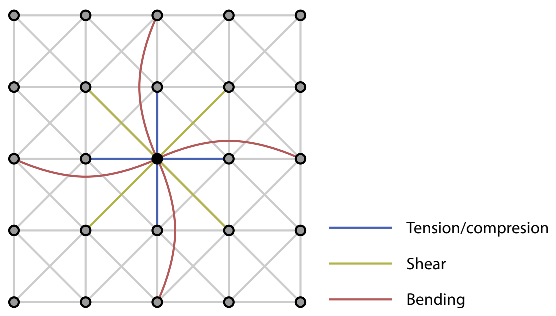


Figure 4.22: Mass-spring system

Mass-Spring model

A common way to calculate internal forces in a cloth is using a mass-spring system, where the cloth is discretised into a network of masses interlinked with springs to the surrounding masses. The masses are connected with springs to counteract tension; diagonal springs for shear, and interleaving springs for bending (Figure 4.22).

By defining the $l_{ij,0}$ as the rest length between the ith and the jth node and given the position vectors of the nodes as $\mathbf{x}_i$ and $\mathbf{x}_j$ the strain in the springs connected to a node can be calculated as follows.

$$\varepsilon_{ij} = \frac{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|}{l_{ij,0}} \tag{35}$$

The force vector generated by the springs acting on a node can be found using the constitutive equations of elasticity

$$\mathbf{f}_i = \sum_{j=1}^{N} \varepsilon_{ij} k_{ij} \frac{(\mathbf{x}_i - \mathbf{x}_j)}{\left\|\mathbf{x}_i - \mathbf{x}_j\right\|} \tag{36}$$

where k_{ij} is the stiffness of the spring between the ith and jth node and N is the total number of springs connecting to the ith node. The force is linear and proportional to the stiffness of the spring, and the stiffness is defined in the direction of the spring.
With this method it is difficult to define an anisotropic material as the direction of which the stiffness is defined depends on the direction of the springs. Furthermore it is difficult to apply the correct mechanical properties of the cloth to shear and bending springs. This method also restricts the topology of the mesh.

Element model

Because of the need to simulate the internal forces correctly, a different methodology devised by Volino et al. (2009) was used in this thesis.
The cloth is represented by a triangle mesh where the in-plane-forces are calculated from the deformation of the triangles instead of springs. The deformation of a triangle is computed from the vertex positions. To do this, the weft $(1,0)$ and warp $(0,1)$ vectors are expressed as weighted sums of the parametric coordinates (u_a, v_a) (u_b, v_b) (u_c, v_c) of the triangle. (Figure 4.23)

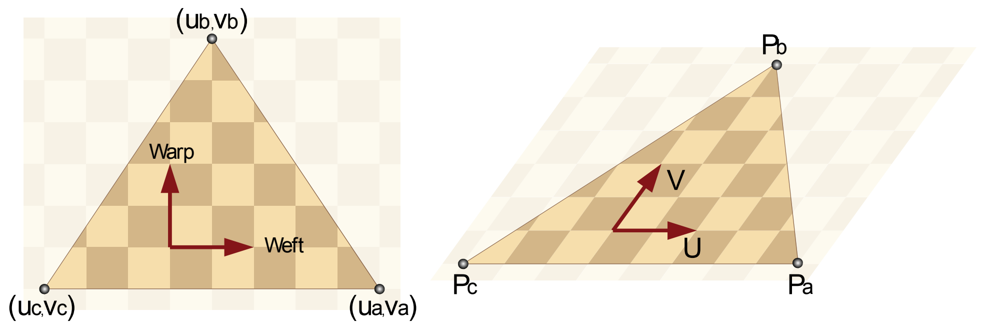


Figure 4.23: Triangle with parametric coordinates (left) and deformed triangle with world coordinates (right) (Volino et al., 2009)

The idea is that the deformation of the weft and warp vector can be found by weighting the deformed vertex positions using the same pre-computed weights. The weft and warp vector can be expressed as weighted sums as follows

$$\begin{aligned}
&\sum_{i\in(a,b,c)} r_{ui}u_i = 1\\
&\sum_{i\in(a,b,c)} r_{ui}v_i = 0\\
&\sum_{i\in(a,b,c)} r_{ui} = 0\\
&\sum_{i\in(a,b,c)} r_{vi}u_i = 0\\
&\sum_{i\in(a,b,c)} r_{vi}v_i = 1\\
&\sum_{i\in(a,b,c)} r_{vi} = 0
\end{aligned} \tag{37}$$

where r_{ui} is the weight of the u component of the ith node and r_{vi} is the weight of the v component of the ith node.

Solving this system of equations leads to the weights

$$\begin{aligned}
r_{ua} &= d^{-1}(v_b - v_c)\\
r_{ub} &= d^{-1}(v_c - v_a)\\
r_{uc} &= d^{-1}(v_a - v_b)\\
r_{va} &= d^{-1}(u_c - u_b)\\
r_{vb} &= d^{-1}(u_a - u_c)\\
r_{va} &= d^{-1}(u_b - u_a)
\end{aligned} \tag{38}$$

where

$$d = u_a(v_b - v_c) + u_b(v_c - v_a) + u_c(v_a - v_b) \tag{39}$$

The parametric coordinates of the triangle need to be pre-computed in order to determine the weights. Assuming that the triangle lies on the XY-plane and the weft and warp direction is parallel to the x- and y-axis of the world coordinate system respectively, the task of finding the parametric coordinates is trivial and the parametric coordinates are equal to the x- and y-coordinates of the undeformed triangle.

In the case that the weft and warp direction is not aligned to the axes the parametric coordinates can be obtained by multiplying the world coordinates with a transformation matrix as follows

$$\begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} x_i \\ y_i \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{40}$$

where θ is the rotation of the weft vector form the x-axis.

The deformed weft U and warp V vector of the deformed shape can be calculated from

$$\begin{aligned} U &= \sum_{i\in(a,b,c)} r_{ui} P_i \\ V &= \sum_{i\in(a,b,c)} r_{vi} P_i \end{aligned} \tag{41}$$

where P_i is the position of the ith node of the triangle referring to world coordinates (Figure 4.23).

The resulting strains are calculated as

$$\begin{aligned} \varepsilon_{uu} &= \frac{1}{2}(U^T U - 1) \\ \varepsilon_{vv} &= \frac{1}{2}(V^T V - 1) \\ \varepsilon_{uv} &= \frac{1}{2}(U^T V + V^T U) \end{aligned} \tag{42}$$

The stresses in the weft and warp directions can be determined using the strains and an appropriate constitutive law. In this thesis Hooke's law for orthotropic materials was used. In three-dimensional cases this involves a 6x6 matrix, but under plane stress conditions this can be reduced to the following

$$\begin{bmatrix} \sigma_{uu} \\ \sigma_{vv} \\ \sigma_{uv} \end{bmatrix} = \frac{1}{1-\nu_{uv}\nu_{vu}} \begin{bmatrix} E_u & \nu_{vu}E_u & 0 \\ \nu_{uv}E_v & E_v & 0 \\ 0 & 0 & G_{uv}(1-\nu_{uv}\nu_{vu}) \end{bmatrix} = \begin{bmatrix} \varepsilon_{uu} \\ \varepsilon_{vv} \\ \varepsilon_{uv} \end{bmatrix} \tag{43}$$

where E_i is Young's modulus in direction i and G_{ij} is the shear modulus in the direction j on the plane with normal in direction i. ν_{ij} is Poisson's ratio corresponding to the contraction in direction j caused by deformation in direction i.

Figure 4.24 shows a fabric hanging from two corners stretched under the influence of gravity in the weft and warp direction respectively. The fabric is modelled with a stiffness of 9000 MPa in the weft direction; a stiffness of 4500 MPa in the warp direction and a shear modulus of 100 MPa. Poisson's ratio in both directions is set to 0 as it is assumed that no interaction between the two directions occurs.

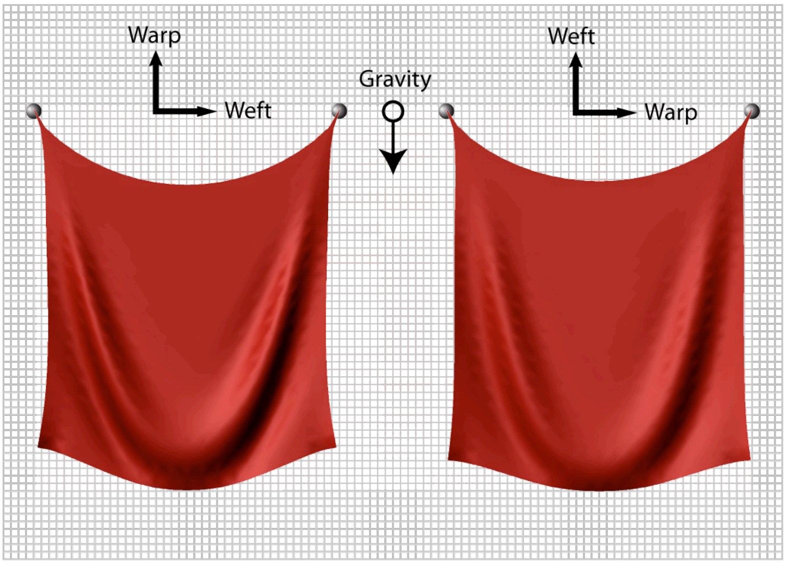


Figure 4.24: Fabric supported at two corners under influence of gravity.

A non-linear material model would simulate the behaviour of the fabric with even more precision and produce a smoother strain distribution (Volino et al., 2009), but for the purpose of this thesis it was decided that an orthotropic linear model was sufficient. It is possible to change the material model without much effort by changing the constitutive law.
From the element stresses, the weights and the weft and warp vectors it is now possible to calculate the forces acting on each node as

$$F_j = -\frac{|d|}{2}(\sigma_{uu}(r_{uj}U) + \sigma_{vv}(r_{vj}V) + \sigma_{uv}(r_{uj}V + r_{vj}U)) \qquad (44)$$

This determines the in-plane forces but the out-of-plane forces still need to be determined and added to the nodal force vector.

A simple yet effective way of calculating bending forces using laplacian-smoothing was implemented in the software. By moving all nodes by a vector defined by the average of the neighbouring nodes average of their neighbouring nodes projected on the vertex normal of the node a smooth surface is achieved. The projection on the normal minimises tangential movement of nodes and thereby shrinkage of the surface is avoided.
To ensure equilibrium in the system the neighbouring nodes are moved in the opposite direction with a force equal to the averaged and projected force divided by the valence of the node.
Bending forces in cloth simulations are mainly added to avoid wrinkling of the cloth and thereby achieve a smoother surface. Figure 4.25 shows the difference between a simulated cloth without (left)

and with (right) bending stiffness. The shading reveals the unsmoothed surface of the cloth without bending stiffness.

The bending stiffness also determines how big the individual creases get and the stiffness should be set according to the thickness and elasticity parameters of the material in question.

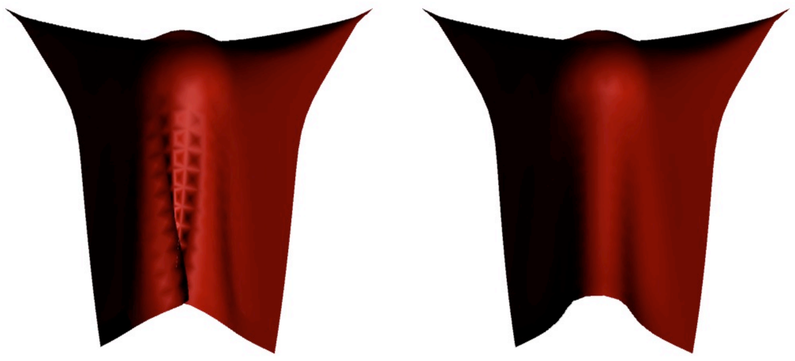


Figure 4.25: Simulation of fabric supported at two corners and draped over a sphere with (right) and without (left) bending stiffness.

4.4 COLLISION HANDLING

Fabric is draped over an optimised solid shape to determine how the shape could be cast using fabric form-work. To simulate this collisions between the fabric and the solid has to be detected and dealt with in a proper manner.

This section describes the custom collision handling technique implemented FabricCast.

The obvious approach of detecting collisions is to test every object against every other object with every iteration. The method is very easy to implement but not very computational efficient.

The collision handling used in this thesis is based on the fact that the mesh is axis aligned and evenly spaced along the different axes. This leads to a very simple and computational effective way of handling collisions.

It is noted that this algorithm is computational independent of the number of elements in the static mesh which makes is very suitable for exactly this application.

Collision handling can be divided in to two distinct problems, detecting a collision and responding to that collision. In the following a method for collision detection and response is described. The method relies on the assumption that the time step used in the fabric simulation is very small and thus the displacement for each iteration step is smaller than the minimum element size.

In the main part of this section the methods is described in two dimensions for simplicity, but can easily be expanded to three dimensions.

VOXEL ADDRESS

By using a quick look-up algorithm to check whether collision handling should be triggered or not the computational time used for the collision handling is greatly reduced.

The algorithm checks whether the new position of a vertex $(\mathbf{x}_{t+\Delta t})$ is inside a solid voxel, if so the collision handling is triggered (Figure 4.26). This test has to be run for every vertex at each calculation cycle, so by reducing this to a minimum a lot of computational power is saved and allows the simulation to be more interactive.

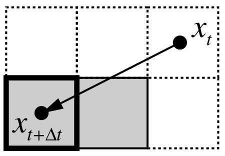


Figure 4.26: Collision detection

The idea of the algorithm is to convert a coordinate in space to an address consisting of three integer values corresponding to the three axes. The values determine how many mesh-increments one should take along a given axis to find the voxel. The integer values are calculated the following way

$$i = \left\lfloor \frac{\mathbf{v}_x}{\Delta x} \right\rfloor \tag{45}$$

$$j = \left\lfloor \frac{\mathbf{v}_y}{\Delta y} \right\rfloor \tag{46}$$

$$k = \left\lfloor \frac{\mathbf{v}_z}{\Delta z} \right\rfloor \tag{47}$$

where $\mathbf{v}$ is the vector between a reference point and the point in question. Δx, Δy and Δz are the mesh increments corresponding to the three different axes (Figure 4.28). The fraction is floored to the nearest integer ensuring consistency when a point is exactly, on a boundary between two elements. These three integers will from this point be referred to as voxel-addresses or simply addresses.

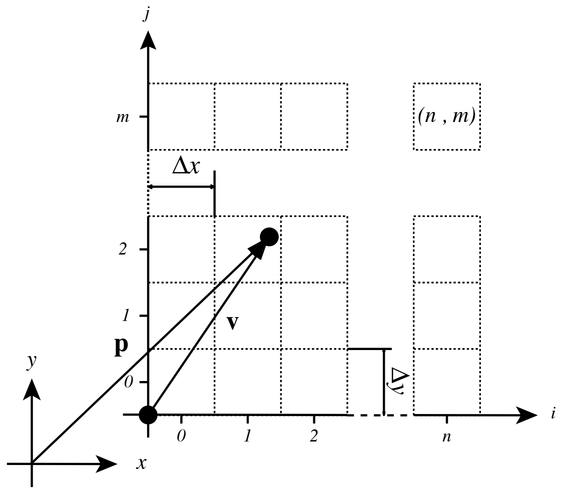


Figure 4.28: Determining voxel address

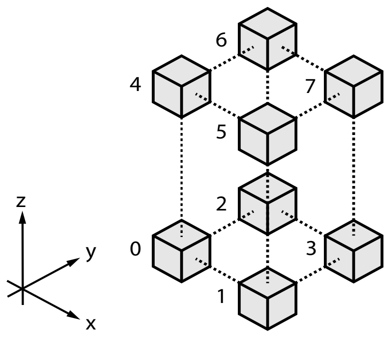


Figure 4.27: Numbering of elements

From this point it is a quick check to see whether the voxel exists and if so is solid. The voxel index number can be calculated as

$$index = i + N_x j + N_x N_y k \tag{48}$$

where N_x, N_y and N_z are the number of elements in the mesh in the three different directions.

Equation (48) depends on the order of the voxels in the list. In this case it is assumed that the elements are ordered as showed in Figure 4.27.

COLLISION HANDLING

The idea of the voxel address used in the collision detection is also used in the collision handling.
At this point the term pseudo-voxel is introduced as the imaginary voxel that has an address outside of the defined mesh. When a collision is detected the address of the position of the vertex in the current time step is found, thus finding the voxel or pseudo voxel containing the coordinate.
Every voxel has six bounding planes that can be intersected. But knowing that the point is inside the voxel, and traveling outward three or more planes can be eliminated. By comparing the address of $\mathbf{x}_t$ with the address of $\mathbf{x}_{t+\Delta t}$ the possible intersection planes can be identified. If two components of the two addresses corresponding to the same axis are equal then there is no intersection through the planes normal to the axis. If the two components are different, the sign of the difference decides which planes to ignore. Figure 4.29 shows two two-dimensional examples with two pairs of addresses. The idea is to identify the possible axis aligned intersection lines of a vertex travelling from one address to another. Initially there are 4 possibilities but knowing the direction the vertex is travelling in a number of these can be eliminated. In the example on the left the first component of both addresses are the same and intersection lines normal to the first axis is eliminated. By looking at the second component the direction of travel along the second axis can be determined. In this case the direction is negative and one more intersection line can eliminated in this case the horizontal line above the address. In the example on the right none of the corresponding components are equal and only two lines can be eliminated based on the direction of travel. The eliminated lines are displayed as dashed lines on the figure.

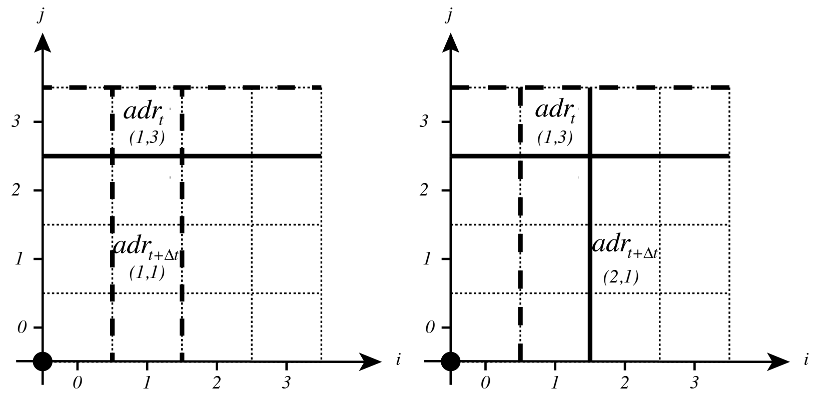


Figure 4.29: Half-space

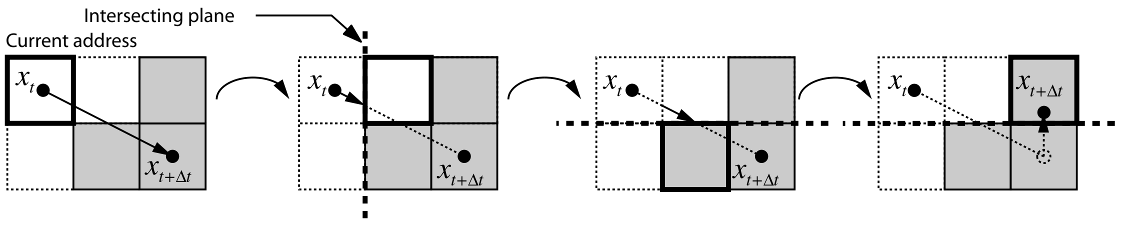
With the reduced list of planes the intersection ratios are calculated. The ratios are calculated in the general case as follows

$$a_n = \left\| \frac{\mathbf{P}_n - \mathbf{x}_t}{\mathbf{x}_{t+\Delta t} - \mathbf{x}_t} \mathbf{n}_n \right\| \qquad (49)$$
$$(n = 1,2,3,\ldots,N)$$

where $\mathbf{P}_n$ is a point on the nth plane and N is the number of planes and $\mathbf{n}_n$ is the unit normal of the nth plane. This is actually a rather trivial task due to the fact that the mesh is axis aligned and the formulation can be simplified by only considering the component of the vector not eliminated by multiplying with the unit normal.

The lowest value of the intersection ratio identifies the plane where the intersection happens first. In special cases there can be two or three minimum ratios. This happens when a point travels through an edge or a corner of a 3D voxel or pseudo voxel. In these cases all the components corresponding to the planes with minimum intersection ratios are identified as intersection planes.

The address is updated by adding or subtracting 1 from the address component corresponding to the identified plane. The new address is used to retrieve a voxel or pseudo voxel. If the new voxel is solid $\mathbf{x}_{t+\Delta t}$ is projected to this plane and the address for the new position is calculated. In practice a slight offset of the plane is used to make sure that the right voxel is returned.

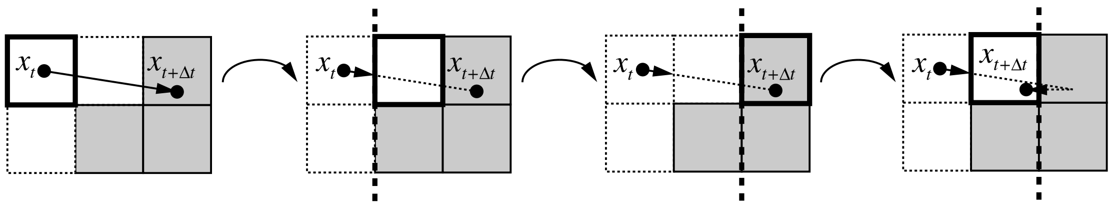


Intersection detected. Set address to current position determine intersection plane.

First intersection plane is found, and address updated. New address is void. Search for intersection planes with new address.

Second intersection plane is found, and address updated. New address is solid.

New position is projected to intersection plane. Address of projected point is solid. Position updated to projected and collision handling is repeated.



Intersection detected. Set address to current position and determine intersection plane.

First intersection plane is found, and address updated. New address is void. Search for intersection planes with new address.

Second intersection plane is found, and address updated. New address is solid.

New position is projected to intersection plane. Address of new point is void. Collisions handling finished.

Figure 4.30: Collision example

If the address returns a void voxel then the position is updated and the collision handling has finished. If not the entire process is run again starting from the address of the projected position.

The entire process is outlined as an example in Figure 4.30 and as a flowchart in Figure 4.31. Figure 4.32 shows a fabric draped over an arbitrary shape as a validation that the collision handling is performing as expected.

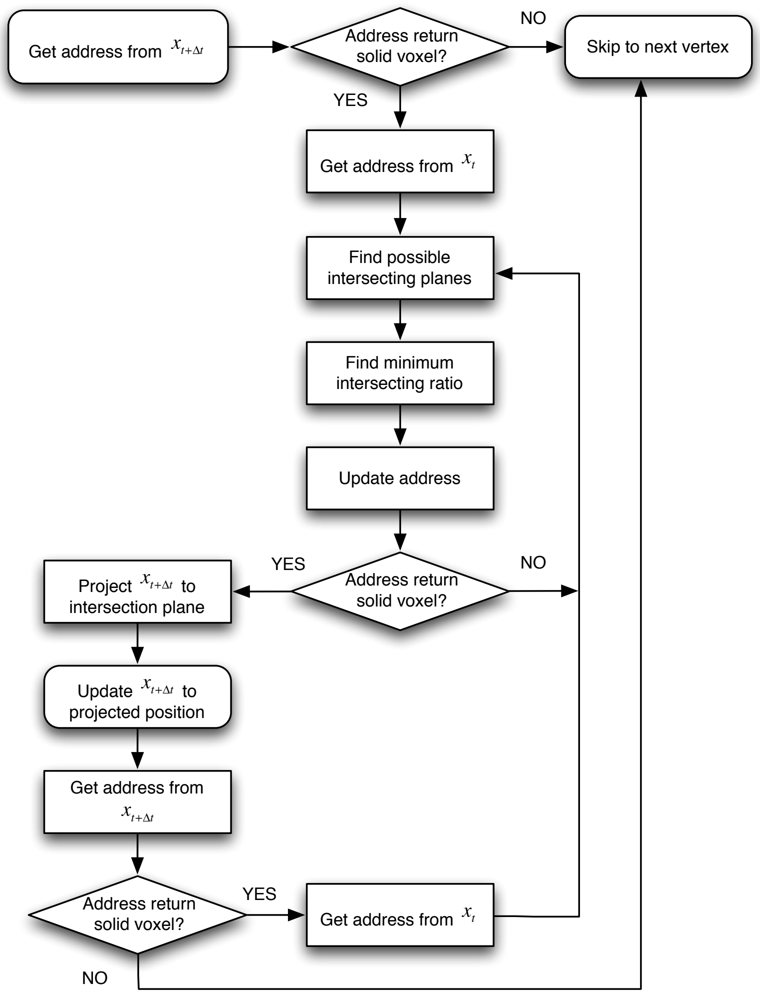

Get address from $x_{t+\Delta t}$
Address return solid voxel?
NO
Skip to next vertex
YES
Get address from x_t
Find possible intersecting planes
Find minimum intersecting ratio
Update address
YES
Address return solid voxel?
NO
Project $x_{t+\Delta t}$ to intersection plane
Update $x_{t+\Delta t}$ to projected position
Get address from $x_{t+\Delta t}$
Address return solid voxel?
YES
Get address from x_t
NO

Figure 4.31: Collision handling flowchart

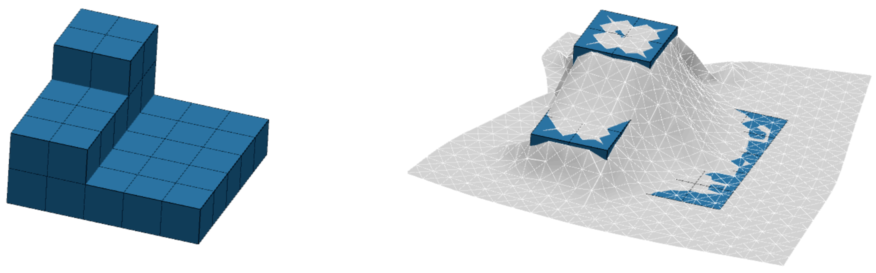


Figure 4.32: Fabric draped over arbitrary shape.

CHAPTER 5 CASE STUDY

This chapter describes a case study of the software tool developed for this thesis. The entire software flow is described from the set up of the mesh to the finished plaster model of a square slab supported by a column at the centre. Two different versions of the software were used in this chapter. An initial run with an early version of the software is used to prove concept and detect any potential improvements. The result of this initial run is used to implement some changes in the software and some results using the latest version are presented.

5.1 THE MODEL

This case study concentrates on a square slab supported by a single square column in the centre. The design-space of the slab is a volume with the dimensions 12x12x0.5m. The column is 2x2m in plan. (Figure 5.1)
The slab is loaded with a uniformly distributed load on the top surface. The elements immediately beneath the loaded surface are defined as non-design space. The elements in the non-design space are part of the structural analysis and therefore contribute to the stiffness of the structure, but cannot be removed, as this would create a design unsuitable for carrying the distributed load.

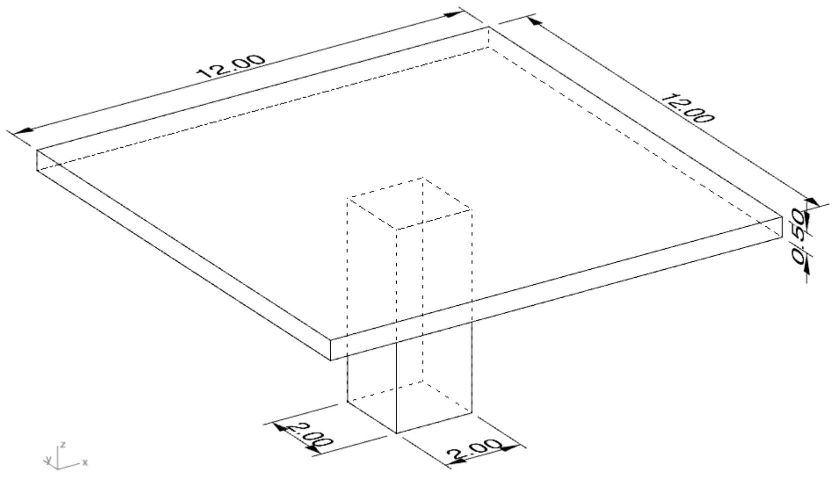


Figure 5.1: Slab dimensions

As noted in chapter 4, the optimisation algorithm used in this thesis tends to cause asymmetric results even though a symmetric outcome is expected. This problem arises because the nature of the algorithm is to remove a certain number of elements. This can cause the algorithm to remove one element and keep another with the same sensitivity value, based on the sequence in which they are evaluated. When this happens the structure develops a tendency to emphasise the asymmetry in future iteration steps. The tendency can be minimised by slowing down the optimisation process, thus allowing the algorithm to ‘self-correct’. However when large problems are analysed an increase in the number of iteration steps is undesirable.

Symmetry planes can be used to eliminate this problem in some cases.

In this case study three symmetry planes have been used. The first two are easily implemented when setting up the model. By introducing these planes the model can be reduced to a quarter size, which also lowers the computational time and memory usage.
The two planes are parallel to the XZ and YZ planes respectively. The symmetry planes are implemented by modelling only one half of the structure divided by the symmetry plane. Supports are added along the edge of the mesh parallel to the planes. In the case of the XZ symmetry plane the edge is constrained in the direction of the Y-axis, thus simulating the interaction with the non-existing half of the topology. These symmetry planes are illustrated in Figure 5.2. The part of the volume that is modelled is marked in red.

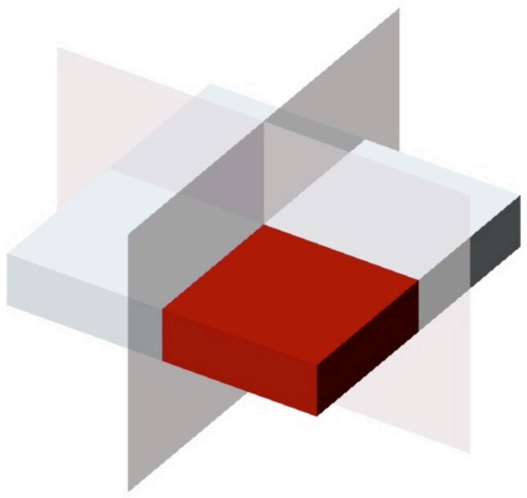


Figure 5.2: Symmetry planes

However this model features a third symmetry plane along the diagonal of the slab. Because of the topology of the mesh this plane splits the element along the diagonal and the third symmetry plane cannot be modelled the same way as the first two.

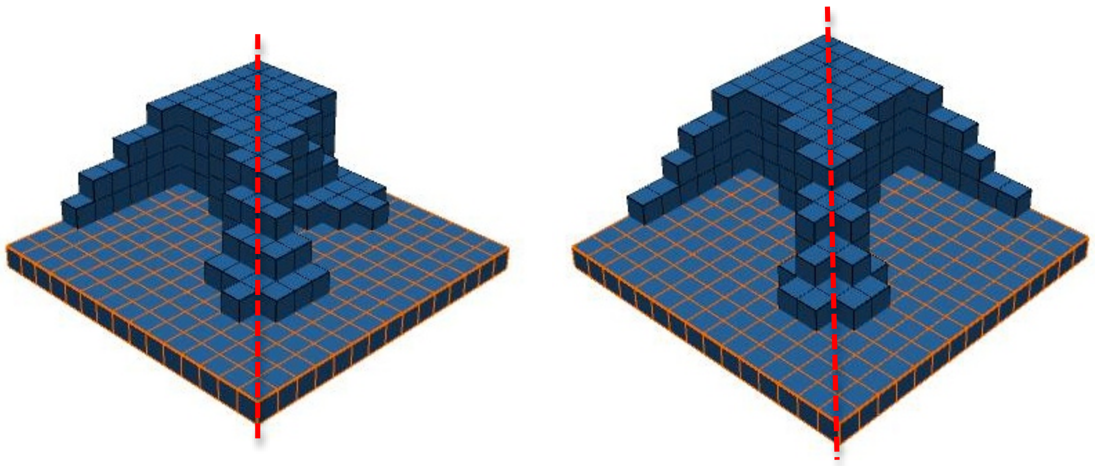


Figure 5.3: Asymmetric (left) and symmetric (right) solution.

The left topology in Figure 5.3 illustrates the asymmetry across the diagonal.

In this case study a small function that checks for symmetry at each iteration step was implemented in the software. When the optimisation algorithm toggles a voxel on or off it performs the same action on the element mirrored by the symmetry plane. This can be done using the voxel addresses introduced for the collision handling (See chapter 4). The addresses of the voxels are unknown at this stage but can be found using the index of the voxels as follows

$$k = \left\lfloor \frac{index}{N_x N_y} \right\rfloor \tag{50}$$

$$j = \left\lfloor \frac{index - N_x N_y k}{N_x} \right\rfloor \tag{51}$$

$$i = \left\lfloor index - N_x N_y k - N_x j \right\rfloor \tag{52}$$

where $index$ is the index of the voxel and N_n is the number of increments in the direction of the nth axis. The variables i, j and k is the address of the voxel.

The mirrored voxel, found by negating the i and j component of the address, is forced to assume the same state as the voxel it is a mirrored image of. This method forces the topology to be symmetric across the diagonal, but does not reduce the computation time or memory usage as all elements is still present in the model. The topology on the right in Figure 5.3 shows the results of an optimisation with three symmetry planes.

The dimension of the volume after being reduced by symmetry planes is 6x6x0.5 m. This volume is discretised in to a 60x60x10 mesh. Figure 5.4 shows a screenshot of the discretised volume in FabricCast. The elements with orange edges represent the non-design space.

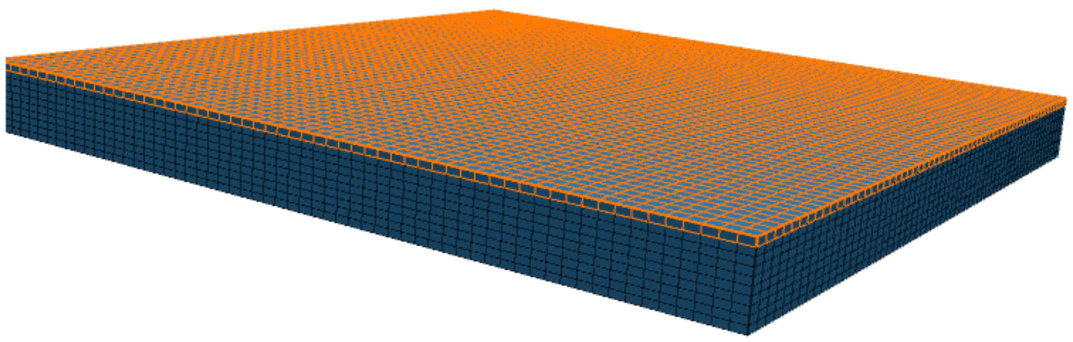


Figure 5.4: Model setup

5.2 INITIAL RUN

The optimised structure presented in this section was created when the development of the software was on going. This was done to test the concept and detect places where the software could be improved. The test let to some changes in the optimisation algorithm and the simulation of the in-plane forces of the fabric. In this early version of the software the optimisation algorithm was based on a earlier version BESO (Huang et al., 2006). The mechanical properties of the fabric were simulated using a simple mass-spring model.
The results presented below were created using this early version of the software.

OPTIMISATION

Optimisation parameters:

RRV = 0.3
RRVi = 0.01
τ = 0.01%

Material properties:

Young's modulus = 50000MPa
Poisson's ration = 0.2

The optimisation starts from a full domain of solid elements and gradually progresses toward a topology with only 30% solid elements. The performance index increases until iteration number 391 where the optimisation process becomes unstable. At this stage the removal of volume does not balance the increase in compliance and the performance index starts to drop. The optimisation algorithm at this stage has a tendency to create checkerboard patterns (Figure 5.5). Updating the optimisation algorithm to a more recent one (Huang and Xie, 2007) reduced the unstable behaviour and occurrence of checkerboard patterns.
The process of the entire optimisation is shown as a chart in Figure 5.6. The topologies shown are turned upside down to display the shapes.

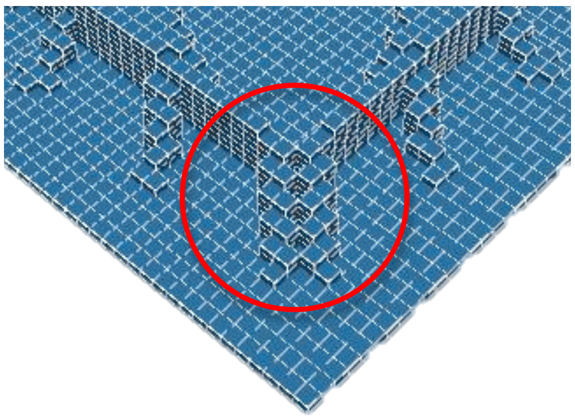


Figure 5.5: Checkerboard patterning

The structure with highest performance index had a volume fraction of 45.5%. This structure was used for further manipulation (Figure 5.7).

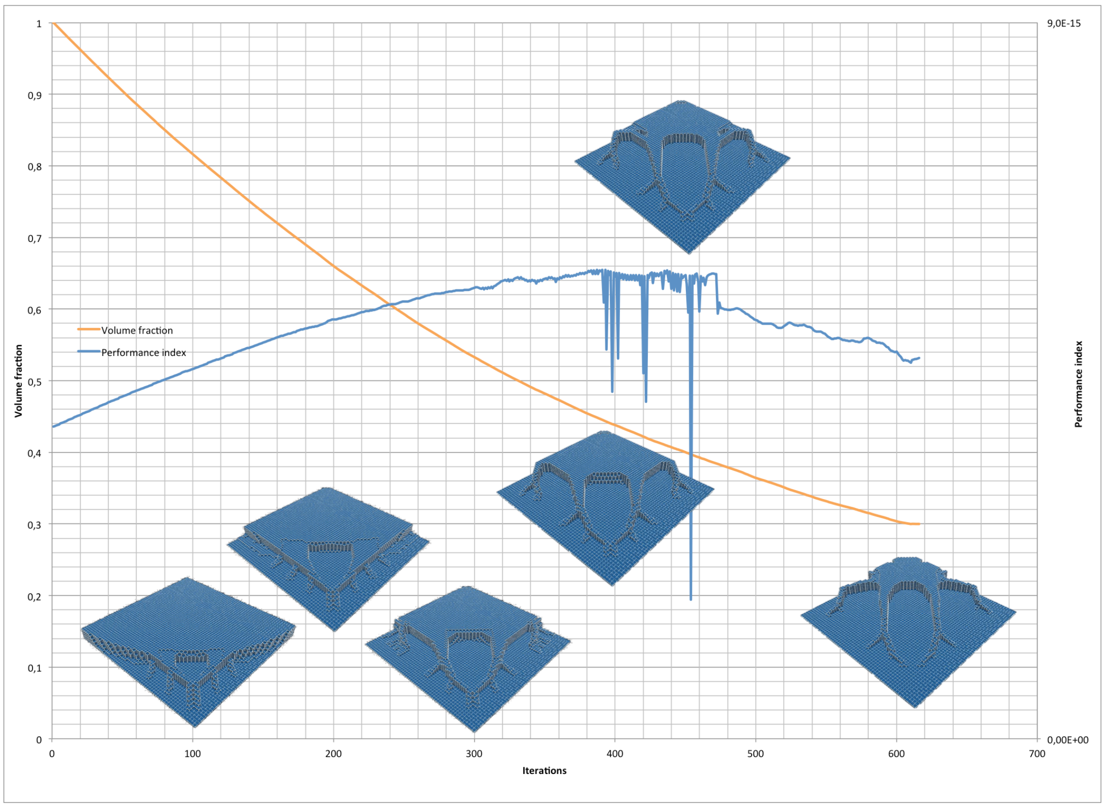


Figure 5.6: Optimisation process

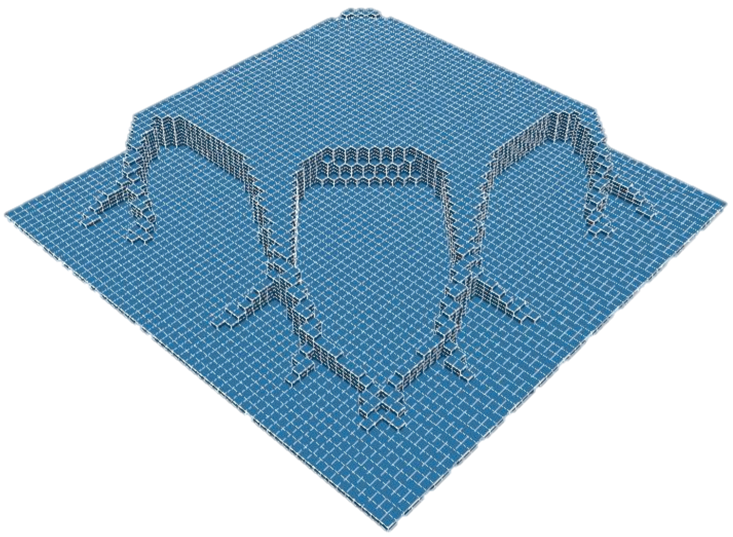


Figure 5.7: Iteration number 391

DRAPING

The next step was to drape fabric over the optimised shape and thereby determine the cut-out pattern for the form-work.

The symmetry planes discussed earlier can also be used in the draping process by only modelling a quarter of the fabric and constraining the fabric along the edges touching the planes. The edges at the symmetry planes are constrained for displacement in the direction of the normals of the planes.

The diagonal symmetry plane was not used in the fabric simulation.

At this stage the mechanical properties couldn't be modelled accurately, only relative stiffness values could be input. A more precise material model was implemented after these tests were conducted.

Nonetheless the software produced a shape of the draped fabric that could be used to find the cut-out pattern and thereby proving the concept.

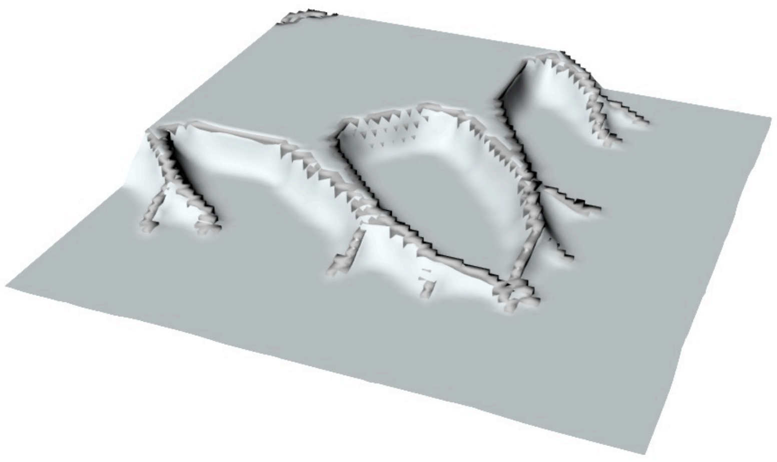


Figure 5.8: Fabric draped over optimised shape

FURTHER MANIPULATION

The fabric was exported to a VRML file that was opened in Rhinoceros where the cut-out pattern was found using Boolean operations between the fabric and an intersecting plane placed by the user. At first the fabric was mirrored two times thus recreating the full geometry that was eliminated earlier because of the symmetry planes (Figure 5.9).Figure 5.11 shows the intersection plane placed just above the flat part of the fabric. The intersection curve between the fabric and the plane defines the cut-out pattern for the formwork. (Figure 5.10)

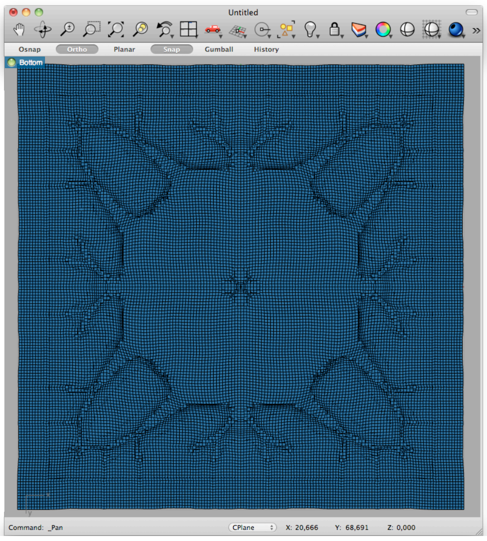


Figure 5.9: Bottom view of full model of fabric

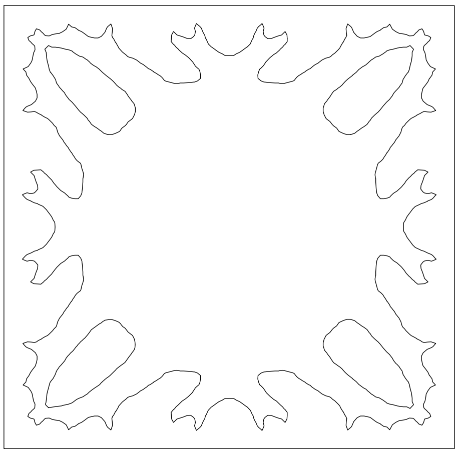


Figure 5.10: Cut-out pattern

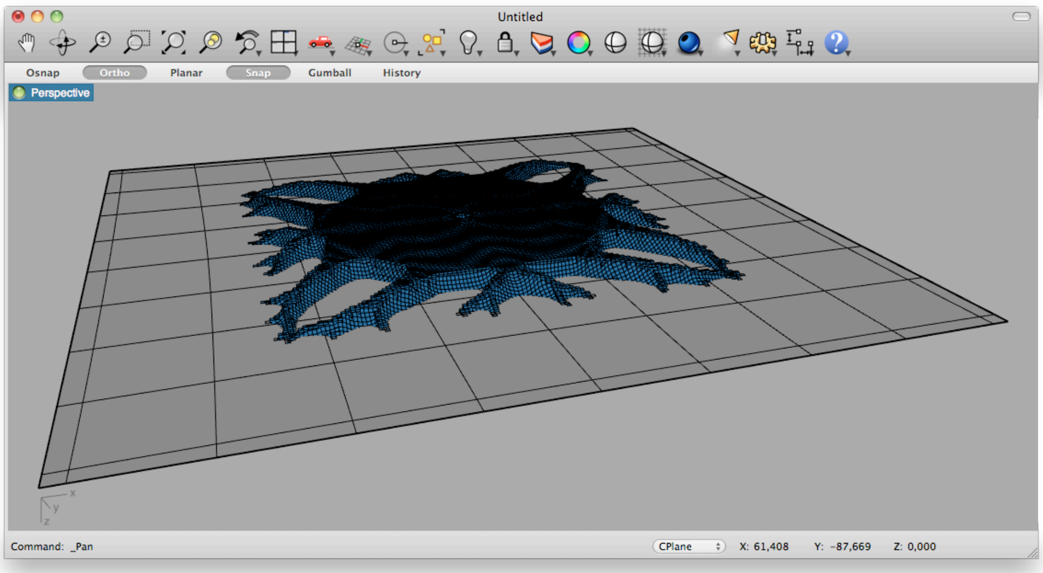


Figure 5.11: Perspective view from Rhinoceros of intersecting plane and fabric

SCALE MODELS

The result from this initial run of the optimisation and formfinding process was used to create a few scale models using plaster.

The method used to cast the scale models was inspired by the techniques developed at CAST and used to create the 'star-capital-slab' described in the introduction of this thesis.

A fabric was placed is placed on a flat base with a cut-out. The cut-out pattern defines the shape of the slab by allowing the fabric to sag under the weight of the plaster. A circular plate, supported from underneath, was placed beneath the cut-out to ensure that the sag of the fabric doesn't exceed the

boundaries of the design space. This plate is marked with yellow in Figure 5.12. The pieces marked with blue are part of the cut-out pattern but had to be supported from underneath as well.
The formwork was made in timber and the cut-out pattern produced by the software tool was cut in a sheet of MDF with a thickness of 6mm.
The scale of the plaster model is 1:12 and dimensions of the formwork are shown in Figure 5.12.

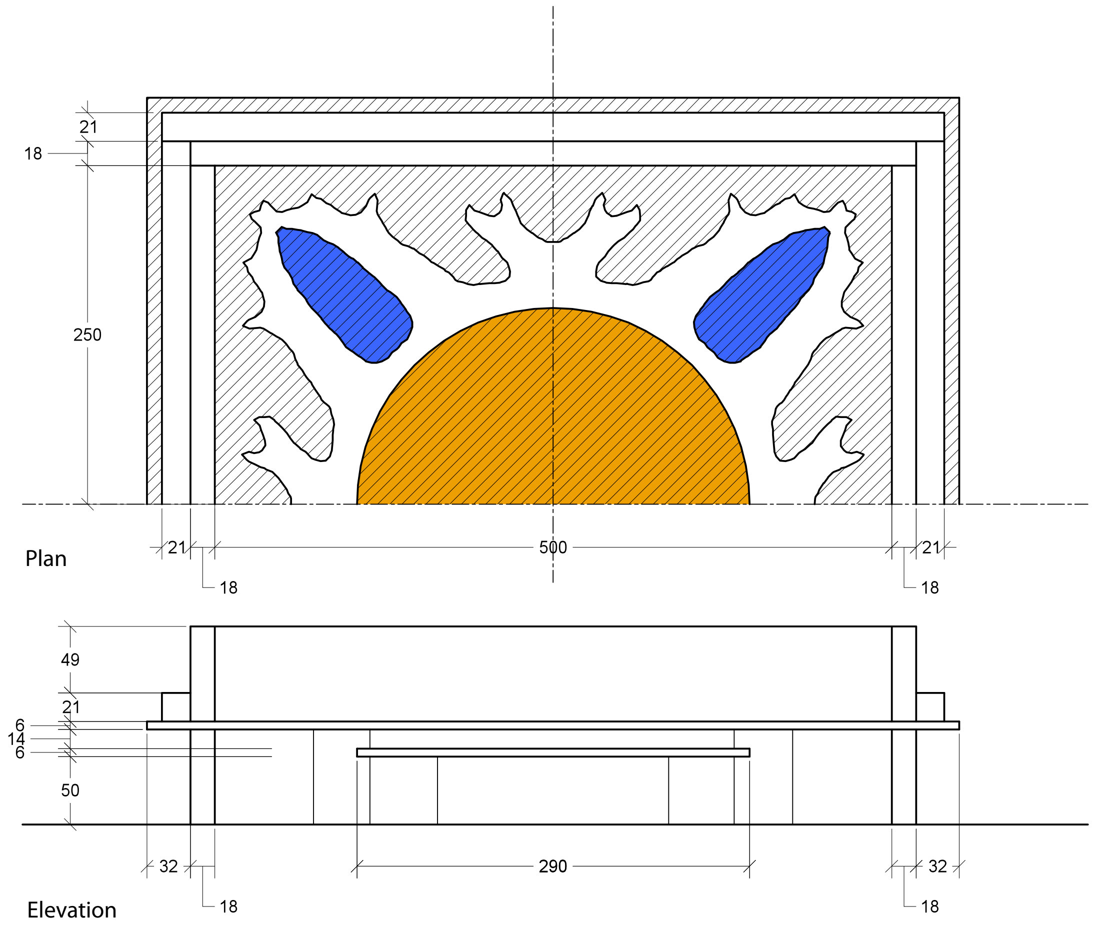
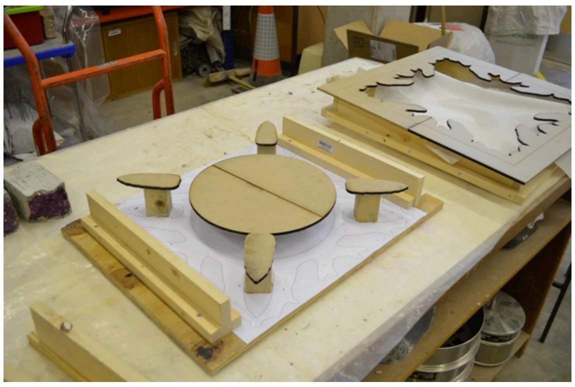
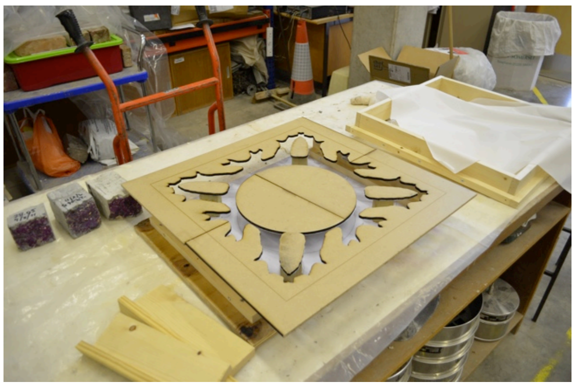
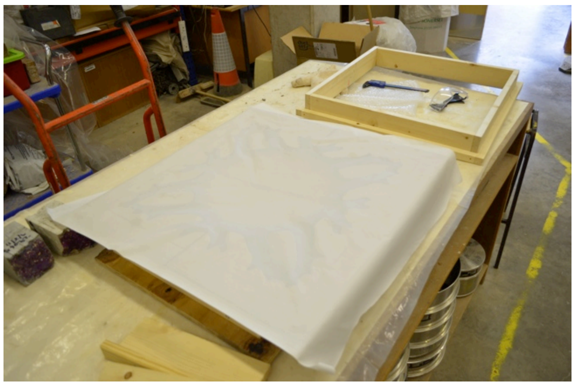


Figure 5.12: Formwork dimensions

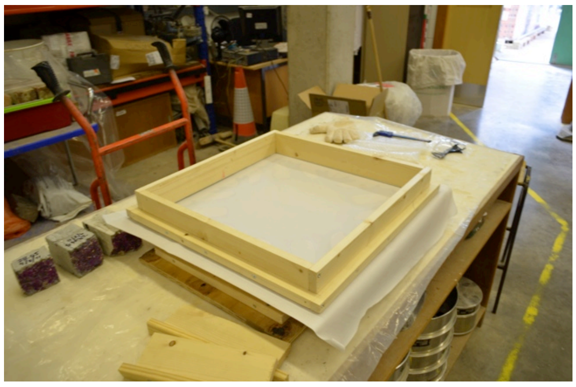
The pictures displayed in Figure 5.13 show the casting process of the plaster models. Picture (a) shows the pieces of MDF that had to be supported underneath. On picture (b) the cut-out pattern was placed on its supports. The fabric was placed on top of the cut-out pattern (c). The frame that goes on top of the fabric created a tight seal against the fabric to prevent the plaster from running out (d). The frame was placed with the seal on top of the fabric (e). At this stage in the process the fabric was stretched until the desired pre-stress level was reached. The same formwork was used to create two models, one with and one without pretension. The results of these two models are discussed later in this chapter.
When the pre-stress level is satisfactory the formwork was fixed using clamps (f).
The plaster was purred carefully in to the formwork (g) and the fabric sagged under the weight of the plaster (h).

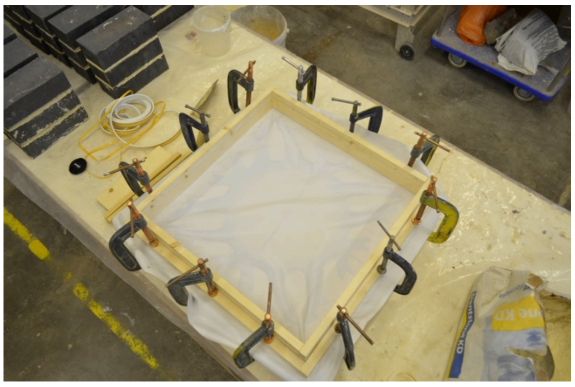

(a)

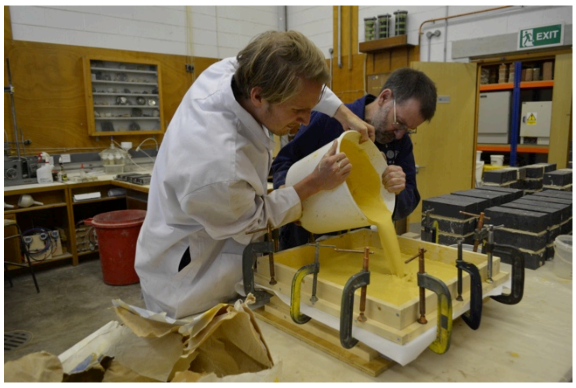

(b)


(c)

(d)


(e)


(f)


(g)

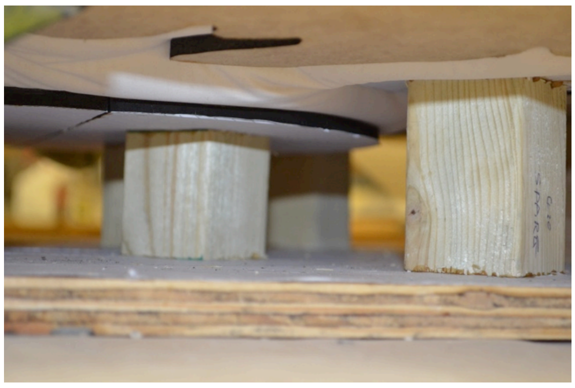
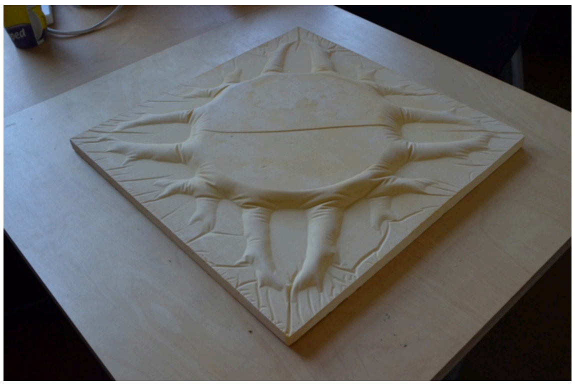
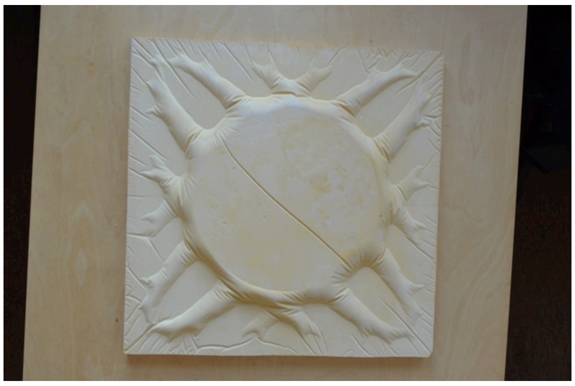

(h)

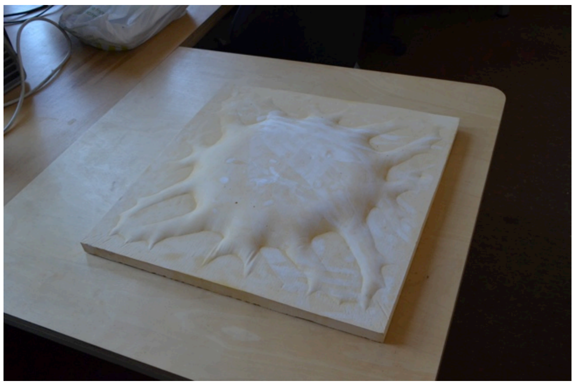
Figure 5.13: Casting process

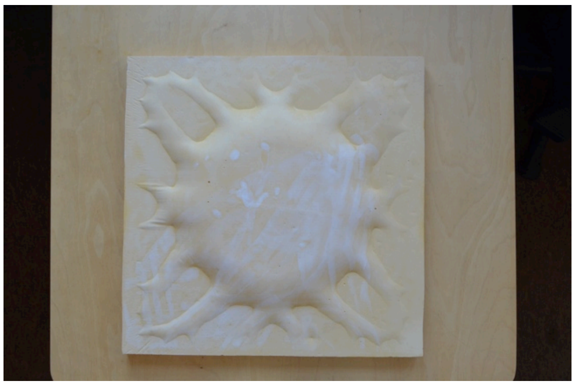
(a)

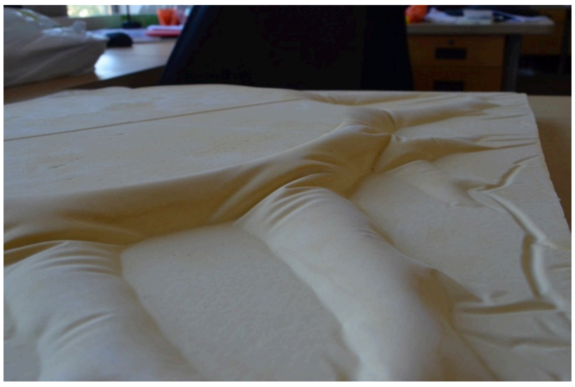
(b)


(c)


(d)


(e)


(f)


(g)

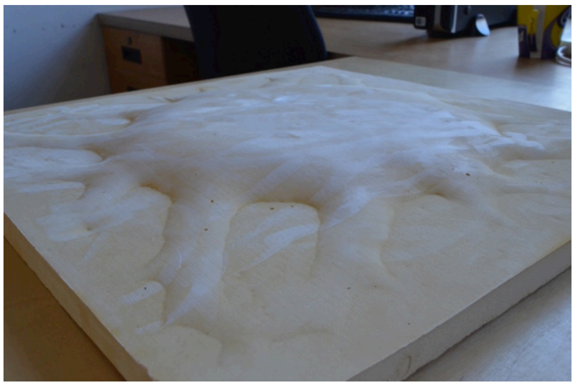

(h)

Figure 5.14: Plaster models

Figure 5.14 shows some pictures of the finished plaster models. Pictures (b), (c), (d) and (g) display the plaster model cast without pre-tensioning the fabric. By not applying pre-tension the fabric was free to move and behave as it pleases. Picture (g) clearly shows the consequences of these movements and creases. In the case of this structure the creases appeared in a very unfortunate position. Picture (g) shows some deep dents in the base of the cantilevering part of the structure thus reducing the cross-section at the point where the highest moment capacity is needed.

Pictures (e), (f) and (h) shows a plaster model in which the fabric was pre-tensioned by hand. This creates a very smooth surface without creases, but because of the pre-tensioning the fabric was not able to sag very deep. Because of the stiff fabric used in the model the finished shape is not very distinct, only the weight of the plaster forces the fabric to deform and stretch into shape.

Both models have their strengths and weaknesses. The desire is to have a model that sags down through the cut-out without making undesired creases.

The problem with this initial run of the optimisation process was that the simulation of the fabric is far from the real physical behaviour because of the material model implemented at this stage. The fabric used for the model is much stiffer than the simulated fabric. That is why the fabric produces creases not detected by the simulation.

5.3 FINAL RUN

This section presents the results from the most recent version of the software tool using the same model used in the initial run presented above.

The production of the scale model revealed some issues with the software and some changes were made to resolve them. The optimisation process was updated to a more recent one (Huang and Xie, 2007) and the mechanical properties of the fabric were simulated using element deformation (Volino et al., 2009) rather than the mass-spring system used in the initial run.

This section only displays the output of the software. The further manipulation of the data and the creation of scale models are left for future work.

OPTIMISATION

Optimisation parameters:

r_{min} = 300 mm
RRV = 0.4
ER = 0.01
DR = 0.1
τ = 0.01%

Material properties:

Young’s modulus = 50000 Mpa
Poisson’s ratio = 0.2

In this final run the optimisation is started from two different topologies referred to as A and B. The first is starts from a domain full of solid elements as the initial run described above (A) (Figure 5.4). The second starts from an initial guess where approximately half of the elements are solid (B) (Figure 5.15). This greatly reduces the computational time, as a smaller number of elements have to be processes at each iteration. Figure 5.15 also shows the tool `HandleBox` that was used to set up the initial topology for the optimisation.

The optimisation algorithm should theoretically reach the same topology regardless of the starting point, but this is not the case in this example. This suggests that the topologies found are not global optima. This is due to the fact that starting the optimisation process at a guess close to the objective volume does not allow the topology to evolve gradually towards an optimum. Some elements may not be considered as part of the structure at any point in the process. Therefore it is argued that the topologies achieved using an initial guess may not be optimal. Nonetheless the results of the optimisation process show an increase in performance from the starting point to the 'optimal topology'. When the size of the problem gets big this may be a sacrifice worth making, considering the reduction in computation time.

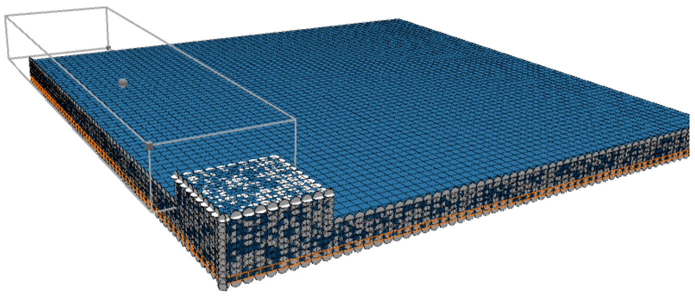


Figure 5.15: Initial guess of topology (B)

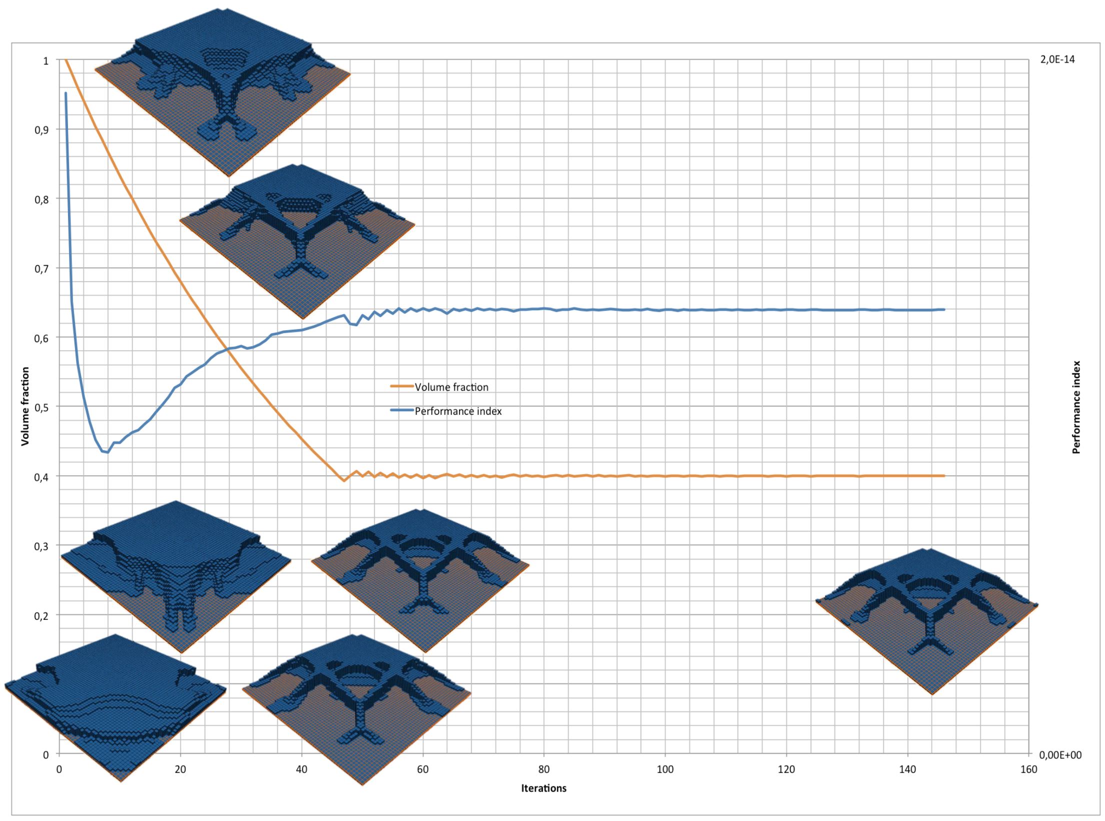


Figure 5.16: Optimisation process of (A)

Figure 5.16 shows the development of the topology from an initial guess of a full design domain to a design with 40% material use. The initial big decrease in the performance index is due to the fact that the sensitivity values of the elements are calculated on the basis an average between the current and the previous iteration. The first iteration uses default values for the sensitivity values, but these values are irrelevant after the first couple of iterations. After this point the performance index increases steadily until the objective volume is reached at the 45th iteration. At this point the volume fraction starts to oscillate around the objective value 0.4. Every time this value is passed the ER value is reduced and the algorithm slowly converges. From this point the topology does not change substantially. The convergence criterion is reached after 147 iterations.

The shape produced by the updated algorithm is a lot smoother than the shape produced by the initial run. Also there are no indications of checkerboard patterns. The difference in the shape from the initial run is due to the change in how the sensitivity values of the void elements are extrapolated from the solid.

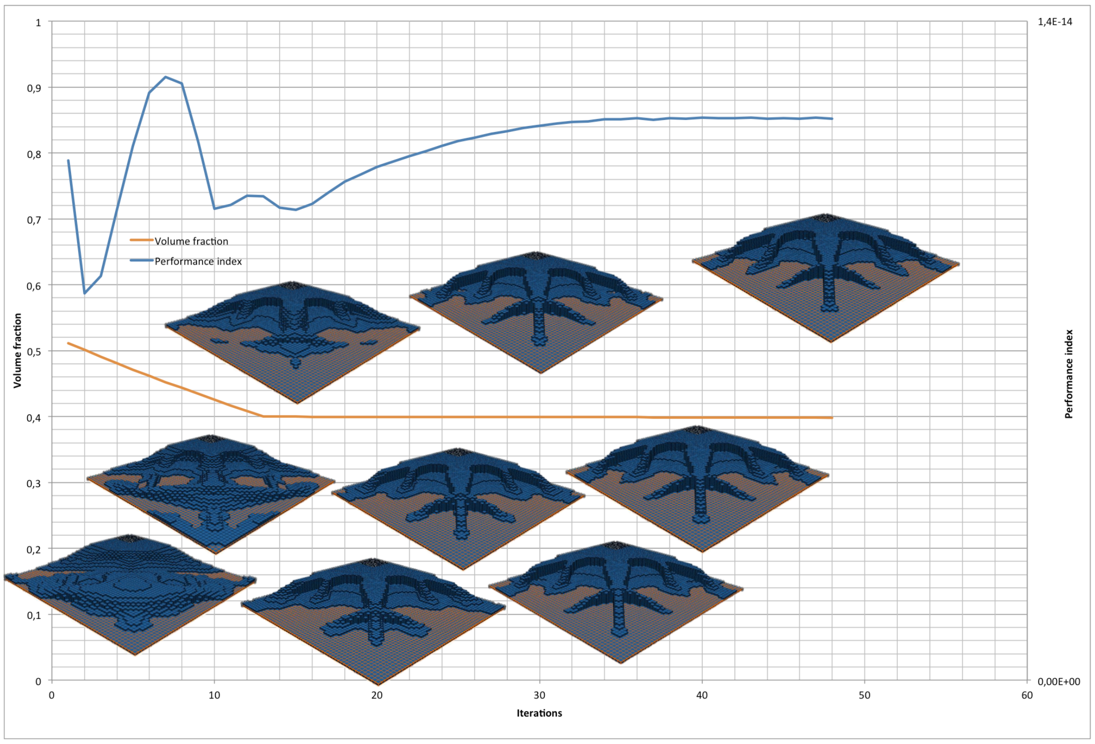


Figure 5.17: Optimisation process of (B)

Figure 5.17 shows the optimisation process started from an initial guess at approximately 50% of the total volume of the deign-space.

The fluctuations in the performance index are due to the considerable changes in topology between iterations. The changes in the topology continue after the objective criterion has been reached. The volume fraction stays steadily at 0.4 while the algorithm searches for a convergent state. Convergence is reached after only 48 iterations. The performance index continues to increase after the objective volume has been reached. This is due to the fact that no volume is added or removed only rearranged at this stage.

DRAPING

Material properties for fabric are usually determined by tests. In the case of this study no material properties for the fabric used could be found, therefore the properties are estimated by the author. Young's modulus of the fabric is set to 9000 MPa in both the weft and warp direction. The shear modulus is 500 MPa and the bending stiffness 200 MPa. Poisson's ratio is set to 0 in both directions as it is assumed that no interaction between the fibres in the two directions occurs.

The fabric used for this simulation is much stiffer than the one used for the initial run. This eliminates some of the problems with creases mentioned earlier at the cost of precision of the shape (Figure 5.18 and Figure 5.19). In the case of topology (A) the fabric bridges over the voids and they will not be present in the cast shape. In both (A) and (B) the optimised shape will be exaggerated when cast using the fabric formwork. The cast shape will still be an optimised shape, but to a lesser extent.

Casting the shapes is left for future work.

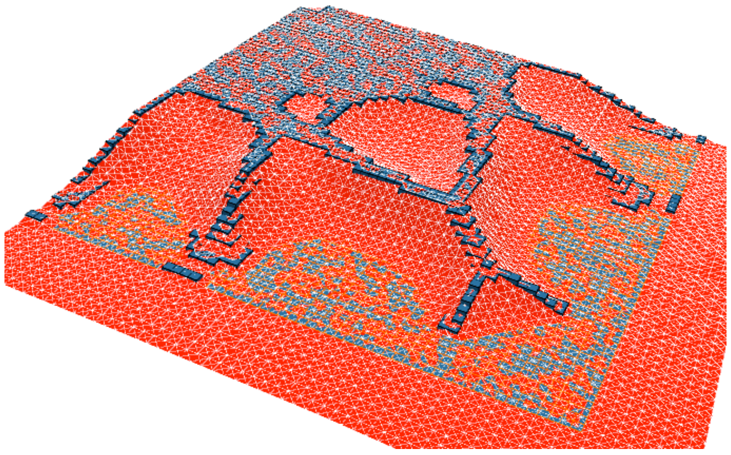


Figure 5.18: Fabric draped over topology (A)

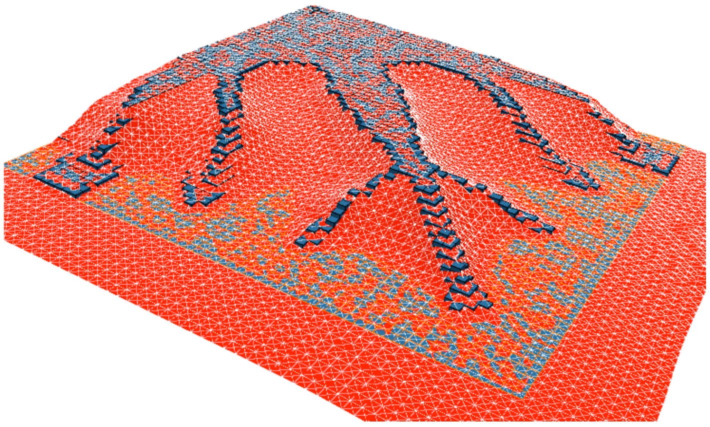


Figure 5.19: Fabric draped over topology (B)

CHAPTER 6 CONCLUSIONS

This thesis has presented a specialised methodology for calculating and casting an optimised shape using fabric formwork techniques. As part of this methodology a novel software tool (FabricCast) was developed. FabricCast can be used to design optimised shapes that are economical both in manufacturability and material usage.

A literature review showed that the research conducted in this particular area is very sparse, but also showed a rising interest in the subject. Therefore an investigation in existing methods and the development of new methods was necessary.

The case study in this thesis concentrates on the design of a slab supported by column in the center, but the method could without much effort be extended to other types of boundary conditions or other types of elements such as beams or panels.

As part of the software a customised topology optimisation algorithm based on known techniques was implemented. The algorithm was customised in such a way that the optimised shapes produced could be cast using fabric formwork techniques.

Different topology optimisation techniques were reviewed and the BESO algorithm was chosen for further development because of the easy implementation into existing code. The 'hard-kill' nature of the algorithm produces shapes with unambiguous definition of boundaries and therefore there was no need to implement surface reconstruction algorithms. Due to the simple nature of the algorithm it was easy to customise to produce the desired type of topologies.

Maximising the stiffness, i. e. minimising the mean compliance of the structure, was the driving force for the optimisation algorithm. It could be desired that different optimisation objectives could be implemented along with some constraint functions, such as minimum/maximum stress constraint, to further improve the performance of the design. After this improvement has been implemented the next logic step would be incorporate multi-objective optimisation techniques along with the ability to analyse multiple load cases.

A simple isotropic material model was used in this thesis to simulate the mechanical properties of concrete. This was done to simplify the problem with the argument that the goal of this thesis was to propose a way to fabricate optimised shapes, not to develop a way to optimise reinforced concrete elements. However a more accurate concrete material model would be preferable for future development.

A verification of the 'cast shape' in terms of a structural and shape analysis is not performed in this thesis. The geometry of the cast shape needs to be compared with the optimised shape to determine the level of approximation in the method and the efficiency and manufacturability of the structure. Furthermore for the structure to be cast in full scale a detailed structural analysis would be needed. This could be an interesting study for future work.

One aim of the thesis was to simulate a fabric draping over a solid shape to mimic the behaviour of a real-life fabric. At first a simple mass-spring model was implemented but it later showed to be inadequate and a more precise material model was used. The numerical integration technique called Störmer-Verlet was combined with this to simulate the dynamic behaviour of the fabric with good precision and stability. However the method suffers from the tendency to become unstable when a stiff material is being simulated. This tendency was counteracted by reducing the size of the time step. By reducing the time step the simulation gets slower and any real-time interaction is impossible. This is a problem that needs to be solved for future development of the method.
The technique for calculating the out-of-plane forces due to the bending stiffness of the fabric used in this thesis is computationally inefficient because it relies on calculating the vertex normals on the fly. This is necessary to avoid the nodes of the fabric drifting tangentially, thus shrinking the fabric. The speed of the simulation could be improved be implementing a faster technique.

The hydrostatic forces of the concrete along with the formwork help shape the design when casting in non-rigid fabric formwork. The effect of the hydrostatic forces is not considered in this thesis, but it could be interesting to incorporate this in the optimisation process to further improve the level of accuracy of the finial design.

Draping fabric over a detailed model with a large number of elements also makes real-time interaction impossible. This is the case even though a fast collision handling method, with a computational time linear and proportional to the number of nodes in the fabric, was implemented.
When working with big models with a large number of elements, the fabric has to be simulated with a corresponding number of elements to avoid the solid elements slipping through the fabric, as the collision handling is based on the vertices of the fabric intersecting with the planes defining the boundaries of the solid elements. The speed of the simulation could be improved by implementing some of the space division techniques mentioned in the literature review.

The BESO algorithm relies on regular elements, thus the optimised shapes consist of an arrangement of cubic elements and not a smooth shape.
A tendency of the fabric to be caught on the sharp corners of the optimised shape was observed. Using smoothing algorithm such as, marching cubes, subdivision surfaces, Poisson smoothing or voxel centre smoothing, etc. could help avoid this undesired tendency and easy the interactive process of the design. However changing the topology of the solid mesh would influence the collision handling and the method used in this thesis would no longer be valid. Standard collision detection, such as bounding volumes and space division, could be implemented instead.
Another approach to avoid the fabric getting caught on the corners is to base the collision detection on the faces of the fabric intersecting the solid, as oppose to the vertices of the fabric intersecting the solid.

After the shape has been formfound it is necessary to export it to another software to create the cut-out pattern. An implementation of this task in the software tool would speed up the design process and eliminate the need for additional software. This implementation would also make it possible to analyse the cut-out pattern and use it as part of the formfinding process.

The manufacture method proposed in this thesis is working under the assumption that a single flat sheet of fabric is used. A further development of the method could include the use of fabric cut-out patterns to create a shape with larger precision. This does however complicate the manufacturing process to a degree that might be undesirable, but nevertheless it is worth investigating further.

The software tool was developed using an object-oriented approach, which holds well with the different parts of the software that needed to be developed. The elements of the mesh and their connectivity information were easily stored as objects and the structure of the software was very intuitive due to the object-oriented approach. This helped to speed up the development and results were quickly available. The programming language used has multi threading capabilities and additional use of this, combined with hardware acceleration, could drastically increase the speed of the optimisation, fabric simulation and overall performance of the software.

A few design studies were realised using plaster scale model. These models drew attention to certain characteristics of the method proposed.
Using the casting techniques discussed in this thesis a compromise between precision and manufacturability has to be made. The optimised shape can be closely approximated when using a fabric not under the influence of pre-stress forces. The shapes created using the non-pre-stress approach closely resemble the optimised shape, but because the fabric is free to crease, undesired aesthetics and structural shapes can emerge. Using an approach with a pre-stressed fabric avoids the creasing but the cast shape is further from the optimised shape. Closer approximation of the optimised shape with a pre-stresses design approach could be achieved by improving the fabric material model and implementing the hydrostatic forces of the concrete and the cut-out pattern for the formwork as part of the design process.

Overall a good base for a framework for the design and manufacture of optimised shapes cast using fabric formwork has been created, but further validation and improvement of the method is necessary.

REFERENCES

ANSYS. 2011. *ANSYS* [Online]. Available: http://ansys.com/ [Accessed October 2011].

AR, S. 2000. Self-customized BSP trees for collision detection. *Computational geometry,* 15**,** 91-102.

AUTODESK. 2011. *Robot Structual Analysis* [Online]. Available: http://usa.autodesk.com/robot-structural-analysis-professional/ [Accessed October 2011].

BARBER, C. B., DOBKIN, D. P. & HUHDANPAA, H. 1996. The quickhull algorithm for convex hulls. *ACM Trans. Math. Softw.,* 22**,** 469-483.

BENDSØE, M. P. 1988. Generating optimal topologies in structural design using a homogenization method. *Computer methods in applied mechanics and engineering,* 71**,** 197-224.

BENDSØE, M. P. 1989. Optimal shape design as a material distribution problem. *Structural and multidisciplinary optimization,* 1**,** 193-202.

CAST. 2011. *Centre for Architectural Structures and Technology* [Online]. Available: http://www.umanitoba.ca/cast_building/ [Accessed October 2011.

CUI, C., OHMORI, H. & SASAKI, M. 2003. *Computational morphogenesis of 3D structures by extended ESO method,* Madrid, ESPAGNE, International Association for Shell and Spatial Structures.

CUI, C., OMORI, H. & SASAKI, M. 2005. Structural Design by Extended ESO Method. *Joho Shori Gakkai Shinpojiumu Ronbunshu,* 2005**,** 149-156.

DOMBERNOWSKY, P. & SØNDERGAARD, A. 2010. Three-dimensional topology optimisation in architectural and structural design of concrete structures. *Symposium of the International Association for Shell and Spatial Structures - IASS* Universitat Politècnica de València (UPV): Editorial de la Universitat Politécnica de Valencia.

EDWARDS, C. S., KIM, H. A. & BUDD, C. J. 2007. An evaluative study on ESO and SIMP for optimising a cantilever tie-beam. *Structural and multidisciplinary optimization,* 34**,** 403-414.

ERICSON, C. 2005. *Real-time collision detection : Christer Ericson,* Amsterdam, Elsevier : Morgan Kaufmann.

FUCHS, H., KEDEM, Z. M. & NAYLOR, B. F. 1980. On visible surface generation by a priori tree structures. *SIGGRAPH Comput. Graph.,* 14**,** 124-133.

GOTTSCHALK, S., LIN, M. C. & MANOCHA, D. OBBTree: A Hierarchical Structure for Rapid Interference Detection. SIGGRAPH, 1996. 171-180.

GRINSPUN, E., HIRANI, A. N., DESBRUN, M. & SCHRÖDER, P. 2003. Discrete shells. *Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation.* San Diego, California: Eurographics Association.

HUANG, X. & XIE, Y. M. 2007. Convergent and mesh-independent solutions for the bi-directional evolutionary structural optimization method. *Finite Elements in Analysis and Design,* 43**,** 1039-1049.

HUANG, X., XIE, Y. M. & BURRY, M. C. 2006. A New Algorithm for Bi-Directional Evolutionary Structural Optimization. *JSME International Journal Series C Mechanical Systems, Machine Elements and Manufacturing,* 49**,** 1091-1099.

JEPSEN, C. R., KRISTENSEN, M. K. & KIRKEGAARD, P. H. Flexible Mould for Precast Concrete Elements. International Symposium of the International Association for Shell and Spatial Structures (IASS), 2010 Shanghai. China Architecture & Building Press, 2726-2737.

JIANG, Y. 2010. *Real-time cloth simulation based on improved Verlet algorithm*

2010 IEEE 11th International Conference on Computer-Aided Industrial Design and Conceptual Design, CAID and CD'2010.

KLOSOWSKI, J. T., HELD, M., MITCHELL, J. S. B., SOWIZRAL, H. & ZIKAN, K. 1998. Efficient Collision Detection Using Bounding Volume Hierarchies of k-DOPs. *IEEE Transactions on Visualization and Computer Graphics,* 4**,** 21-36.

LARSSON, T. Fast and Tight Fitting Bounding Spheres. SIGRAD, 2008.

MAGNENAT THALMANN, N. 2010. *Modeling and simulating bodies and garments,* London, Springer.

MCNELL. 2011. *Rhinoceros 4.0* [Online]. Available: http://www.rhino3d.com/ [Accessed October 2011].

MELE, T. V. & BLOCK, P. A Novel Form Finding Method for Fabric Formwork for Concrete Shells. International Association for Shell and Spatial Structures (IASS), 2010 Shanghai.

NEALEN, A., MÜLLER, M., KEISER, R., BOXERMAN, E. & CARLSON, M. 2006. Physically Based Deformable Models in Computer Graphics. *Computer graphics forum,* 25**,** 809-836.

NIKISHKOV, G. 2010. *Programming finite elements in Java(tm),* London, Springer-Verlag.

OHMORI, H., HIROYUKI, F., TOSHIHIKO, I., ATSUSHI, M. & YASUTOSHI, H. 2005. Application of Computational Mophogenesis To Structural Design. *Joho Shori Gakkai Shinpojiumu Ronbunshu,* 11**,** 45-52.

ORACLE. 2011. *JAVA 1.6* [Online]. Available: http://java.com/ [Accessed October 2011].

PROCESSING. 2011. *Processing 1.5.1* [Online]. Available: http://processing.org/ [Accessed October 2011].

PROSCENE 2011. Proscene.

QUERIN, O. M., STEVEN, G. P. & XIE, Y. M. 1998. Evolutionary structural optimisation (ESO) using a bidirectional algorithm. *Engineering computations,* 15**,** 1031-1048.

QUERIN, O. M., STEVEN, G. P. & XIE, Y. M. 2000. Evolutionary structural optimisation using an additive algorithm. *Finite Elem. Anal. Des.,* 34**,** 291-308.

SAMUELSSON, A. & ZIENKIEWICZ, O. C. 2006. History of the stiffness method. *International Journal for Numerical Methods in Engineering,* 67**,** 149-157.

SCHMITZ, R. P. Fabric-formed Concrete Panel Design. Architectural Engineering National Conference, 2006 Omaha. 1-15.

SHIFFMAN, D. 2008. *Learning Processing : a beginner's guide to programming images, animation, and interaction,* Amsterdam; Boston, Morgan Kaufmann/Elsevier.

VEENENDAAL, D. 2008. *Evolutionary Optimization of Fabric Formed Structural Elements.* Master, Delft University of Technology.

VERLET, L. 1967. Computer "Experiments" on Classical Fluids. I. Thermodynamical Properties of Lennard-Jones Molecules. *Physical Review,* 159**,** 98.

VOLINO, P. & MAGNENAT-THALMANN, N. 2001. Comparing Efficiency of Integration Methods for Cloth Simulation. *Proceedings of the International Conference on Computer Graphics.* IEEE Computer Society.

VOLINO, P. & MAGNENAT-THALMANN, N. 2006. Simple linear bending stiffness in particle systems. *Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation.* Vienna, Austria: Eurographics Association.

VOLINO, P., MAGNENAT-THALMANN, N. & FAURE, F. 2009. A simple approach to nonlinear tensile stiffness for accurate cloth simulation. *ACM Trans. Graph.,* 28**,** 1-16.

WEST, M. 2011. Flat Sheet Formwork for Concrete Structural Slabs. Manitoba: University of Manitoba, Centre of Architectural Structures and Technology.

WEST, M. & ARAYA, R. 2009. Fabric Formwork For Concrete Structures And Architecture. *International Conference on Texitile Composites and Inflatable Structures.* Barcelona.

WIKIPEDIA. N. d. *A list of finite element software packages* [Online]. Available: http://en.wikipedia.org/wiki/List_of_finite_element_software_packages [Accessed October 2011].

WILHELMS, J. & GELDER, A. V. 1992. Octrees for faster isosurface generation. *ACM Trans. Graph.,* 11**,** 201-227.

XIE, Y. M. 1993. Simple evolutionary procedure for structural optimization. *Computers & structures,* 49**,** 885-896.

YI-SI, X., LIU, X. P. & SHAO-PING, X. Efficient collision detection based on AABB trees and sort algorithm. Control and Automation (ICCA), 2010 8th IEEE International Conference on, 9-11 June 2010 2010. 328-332.

APPENDIX A – DVD

DVD CONTENTS:

- FabricCast Source Codea
- Case Study (Chapter 5)
    - Pictures of the optimisation process
    - TXT-Files with optimisation process data
    - VRML file containing the draped fabric
    - VRML file containing the optimised solid
    - .mesh files with optimised topology (can be loaded using FabricCast)