

MASTER

3D printing of concrete structures

Wolfs, Rob

Award date:
2015

[Link to publication](#)

Disclaimer

This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

3D PRINTING OF CONCRETE STRUCTURES

GRADUATION THESIS

R.J.M. Wolfs





Eindhoven University of Technology

Department of the Built Environment

Master Architecture, Building and Planning

Specialization Structural Design

Title: 3D PRINTING OF CONCRETE STRUCTURES

Report number: A-2015.85

Version: 1.0

Date: February 2015

Student: R.J.M. (Rob) Wolfs

ID-number: o675429

Email: r.j.m.wolfs@student.tue.nl
robwolfs@live.nl

Graduation committee: Prof. Dr. Ir. T.A.M. Salet ¹ (Chairman)

Dr. Dipl.-Ing. J. Beetz ¹

Dr.-Ing. G. Zimmermann ²

Ing. L.N. Hendriks ³

¹ Eindhoven University of Technology

² G.tecz

³ CyBe Additive Industries

PREFACE

The thesis that lies before you concludes my graduation project on 3D Printing of Concrete Structures. It is the final part of the masters Architecture, Building and Planning, specialization Structural Design at the Eindhoven University of Technology.

This project has been carried out under supervision of Prof. Dr. Ir. T.A.M. (Theo) Salet, Dr. Dipl.-Ing. J. (Jakob) Beetz, Dr.-Ing. G. (Gregor) Zimmermann and Ing. L.N. (Berry) Hendriks. I would like to express my gratitude to the members of my graduation committee, for their support and feedback throughout the project.

Additionally, my thanks go to the fellow students of the 'fifth floor' of Vertigo. The ambience in which we helped each other pick up the tasks of parametric designing and programming in Python has made it a very pleasant project, during which learned much from one another.

Finally, I would like to thank my friends and family who have supported me over the course of my study and this graduation project.

Rob Wolfs

Eindhoven, February 2015

ABSTRACT

3D Printing is one of the most promising developments of today. It has shown its potential in a wide range of disciplines, varying from the medical world to the food industry and from aerospace engineering to household uses. Unsurprisingly, the building industry has adopted this technique and aims to apply it on a larger scale. 3D Concrete printing results in a low cost and high speed construction method, which allows for a greater freedom in both architectural and structural design. Despite these clear benefits shown by a handful of pioneering companies and institutes spread around the world, the building industry is still behind in the development of 3D printing. This may be attributed to the lack of fundamental research on the (structural) behaviour of the to-be-printed shapes and materials. This graduation thesis aims to contribute to the foundations of this research, by designing a method to study 3D concrete printing technique.

Initially the state-of-the-art of 3D concrete printing is studied, considering the printed shapes, material behaviour, applied forces and the choice of print technique and strategy. The study shows that the different components involved in this construction method are clearly connected, but the relationships are generally unknown. A research model is developed to evaluate these relations in a smart, efficient way. Because of the great quantity of variables involved, a parametric approach is adopted for this thesis, allowing for a high adaptability of the parameters involved and the relationships between them. This however results in a vast amount of combinations and variations, unable to provide useful solutions, let alone optima, without an extensive amount of (trial-and-error) calculations and time. For this reason the optimization technique of Simulated Annealing (SA) is incorporated in the research model. SA is a problem-solving strategy which allows for a quick evaluation of an unknown domain of parameters, escaping local optima during the iterations and eventually converging to a global optimum.

The parametric relationships between the different 3D printing components are realized in Grasshopper, a graphical algorithm editor, which comes as a plugin for Rhinoceros. Additionally, programming language Python is used to write the required algorithms. Together they act as the input for Finite Element Method (FEM) software Abaqus, which is used to study the 3D printed concrete wall in a structural way. The FEM model allows the printing behaviour to be studied, as it includes orthotropic behaviour which is a typical result of the layered production method. Considering the developments of new printable materials and shapes, non-linear elastic material properties and sandwich cross sections are also incorporated in the FEM model.

Once an efficient link between parametric input, structural analysis and optimization method is established, the 3D printing parameters are included in the research model. The printing speed is chosen as the to-be-studied variable, as this printer property influences two important material characteristics: the bond strength between the printed layers and the overall strength development of the concrete. The relationship between printing time gap (i.e. the time it takes to print a new layer on top of the previous one) and bond strength is applied in varying forms, showing an improvement in interaction between layers as the printer speed increases. Additionally, the strength development is implemented by a concrete maturity method, including tensile and compressive strength development based on (printing) time and temperature.

The impact of 3D concrete printing is finally shown for varying print strategies, i.e. speed, layer size and printing environment (in-situ or prefab). The analyses of the research model show that the loading capacity initially increases along with a higher print speed, as the bond between layers strengthens. However, at higher print speeds the overall strength development of the printed element becomes governing, and the capacity decreases. A reduction of layer height is beneficial for the overall strength

development, but does strongly increase the total construction time. The choice of printing environment clearly influences the end result as well, as the low temperatures of in-situ printing result in much slower strength development compared to controlled, prefab printing.

These simple analyses on a printed structural wall have proven that the influence of 3D printing must not be underestimated and that the printing strategy has to be taken into account during each step from early design to construction. Additional research will have to be carried out aimed at gaining more insight in the components linked to 3D printing. By studying new printable materials and optimizing shapes including the typical properties of 3D printed concrete, the potential of this promising technique can be realised in practice. Using a research model as discussed in this thesis, and expanding it with newly available data, will guide future developments on 3D concrete printing and support its successful implementation in the building industry.

TABLE OF CONTENTS

PREFACE	I
ABSTRACT	III
TABLE OF CONTENTS.....	VII
1. Introduction	1
1.1. 3D Concrete Printing: State-of-the-Art	1
1.1.1. Contour Crafting	1
1.1.2. Concrete Printing	3
1.1.3. D-Shape	3
1.2. Research Goal and Methodology	5
1.3. Thesis Outline	6
2. 3D Printing Research Model	7
2.1. Research Method	7
2.2. FEM Implementation of Printed Structures	8
2.2.1. Orthotropic Behaviour	8
2.2.2. Non-Linear Elastic Materials.....	9
2.2.3. Sandwich Elements	9
2.2.4. Aging Properties.....	10
2.3. Programming Environment	10
3. Optimization Algorithm	11
3.1. General Optimization Methods	11
3.2. Simulated Annealing.....	12
4. 3D Printing Parameters.....	17
4.1. Printing Strategy	17
4.2. Model Input.....	19
5. The Impact of 3D Concrete Printing: Conclusions and Recommendations	23
5.1. Optimization Analyses	23
5.2. Conclusions.....	25
5.3. Recommendations.....	26
6. Bibliography	29

Annex A. 3D Concrete Printing: State-of-the-Art	33
A.1. Additive Manufacturing.....	33
A.1.1. SLA – Stereolithography Apparatus	35
A.1.2. SLS – Selective Laser Sintering.....	35
A.1.3. LOM - Laminated Object Manufacturing.....	35
A.1.4. FDM - Fused Deposition Modelling	36
A.1.5. SGC – Solid Ground Curing.....	37
A.1.6. MJM – Multi-Jet Modelling	37
A.2. 3D Concrete Printing.....	38
A.2.1. Contour Crafting.....	39
A.2.2. D-Shape.....	40
A.2.3. Concrete Printing	42
A.2.4. Yingchuang	45
A.2.5. TotalKustom	46
A.2.6. CyBe Additive Industries	47
A.2.7. BetAbram	47
A.2.8. Emerging Objects.....	48
A.2.9. The Mediated Matter Group	48
A.2.10. IAAC - Minibuilders.....	49
A.2.11. WASP	50
Annex B. FEM Analyses	51
B.1. Loading In Plane	51
B.2. Loading Out of Plane.....	54
B.3. Orthotropic Behaviour	55
B.4. Non-Linear Elastic Materials	56
B.5. Sandwich build-up cross sections.....	58
Annex C. Grasshopper model	61
C.1. Input Modules	62
C.2. Optimization Algorithm	64
C.3. Toggle Abaqus.....	65
C.4. Python scripts	66
C.5. (Graphical) Output post processing.....	66
C.6. (Textual) Optimization results.....	67
Annex D. Python script.....	69

TABLE OF CONTENTS

1. Introduction

Structural engineers nowadays are confronted with a shift of mind in the building industry. The desire for a more sustainable built environment is growing in the western society, which is a call that has to be answered from every discipline involved in the construction cycle. The field of structural design and engineering has found a way to contribute to this transformation: structural optimization. Techniques have been developed over the past years, allowing for designs which carry and transfer loads in an efficient way. While these methods guide the structural engineer in an early design stage, their implementation in practice is restricted so far due to the limitations of traditional production techniques. This new way of designing requires a new way of constructing, which can only be achieved if the building industry embraces the strange and keeps an eye on new and upcoming techniques.

3D Printing of concrete structures is one of these on-going high-tech developments in today's construction technology. The advantages are clear: high speed construction, no need of formwork, less heavy labour and most of all a great increase of freedom to design. This technique allows for mass customization, as it does not require every (structural) element of a building to be identical, for matters of speed or costs. Besides, as the printer only prints where it is desired, solid and massive structures are no longer required. 3D Printing creates new shapes in an efficient manner, answering to the call for a more sustainable built environment.

1.1. 3D Concrete Printing: State-of-the-Art

Progress in 3D printing developments goes rapidly: costs drop, new printable materials are being added on a regular basis and even multiple materials can be printed at once. The technique has shown its potential in a wide range of disciplines, varying from the medical world to the food industry and from aerospace engineering to household uses. The current state of the technique even allows people to have a 3D printer at their homes for an affordable price.

It comes as no surprise that the building industry aims to apply 3D printing on a larger scale. A handful of companies and institutes spread around the world have already been showing the prospects of concrete printing. A brief overview of the state-of-the-art is given in this chapter. A more elaborate study on the varying concrete printing techniques is included in Annex A. Additionally; a summary of commercial 3D printing methods is given in the same annex.

1.1.1. Contour Crafting

The origins of 3D concrete printing lie far back, as shown by Contour Crafting (CC), one of the oldest still existing techniques. The first publications on the technique by Khoshnevis (University of Southern California) can be found in 1998 and much progress has been made since. Contour Crafting is a method of layered manufacturing, using polymer, ceramic slurry, cement, and a variety of other materials and mixes to build large scale objects with smooth surface finish. The smoothness of the extrusion is achieved by constraining the extruded flow in the vertical and horizontal direction to trowel surfaces. The orientation of the side trowel is dynamically controlled to conform to the slope of surface features. CC is also capable of using a variety of materials with large aggregates and additives like reinforcement

fibres (Khoshnevis, Hwang, Yao, & Yeh, 2006). Figure 1.1-right shows how the nozzle may consist of multiple outlets, i.e. one for each side, and others for the inner (core) of a wall structure. Co-extrusion of multiple materials is also possible. By deflecting the nozzle, non-orthogonal surfaces such as domes and vaults can be created (Khoshnevis, 2004).

Little has been published about the material composition and properties of CC concrete, but it is stated that the mixture contains a plasticizer to increase the workability and has a small particle size to accommodate for the nozzle diameter. Experiments have shown a mean compressive strength equal to $18,9 \text{ N/mm}^2$, which is achieved in a few hours of curing (Hwang & Khoshnevis, 2005).

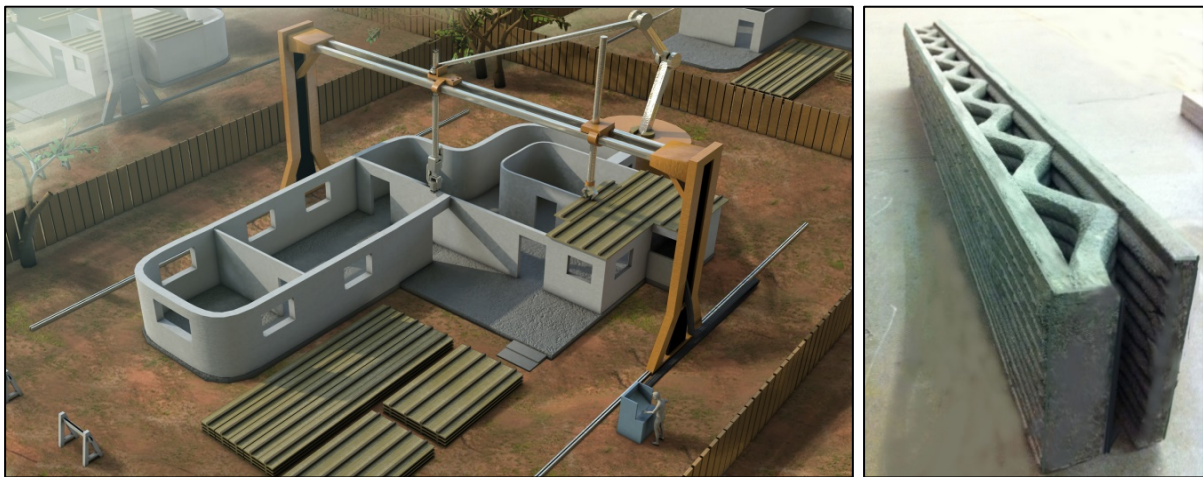


Figure 1.1 - Contour Crafting by University of Southern California (contourcrafting.org, 2014)

Techniques similar to Contour Crafting have been developed over the past few years, showing interesting results coming from diverse regions of the world. The American contractor Andrey Rudenko has founded TotalKustom (TK), a company that is able to print structures comparable to CC, but with a much smaller layer height (*Figure 1.2 left*). Similar developments are seen in China, where Yingchuang has constructed large parts of 3D printed buildings and assembles them on site. Their huge concrete printer (150 x 10 x 6.6 m) uses construction waste materials mixed with cement and glass fibre reinforcement (3Ders.org, 2014). Besides, *Figure 1.2-right* shows the application of fibre mesh reinforcement being applied in between layers.



Figure 1.2 – 3D printed walls by TotalKustom (left) and Yingchuang (right) (totalkustom.com, 2014), (Yingchuang, 2015)

A third party that has adopted the CC technique is CyBe Additive Industries, founded in the Netherlands. This company uses a type of mortar that reaches a bearable strength within minutes. CyBe experiments by attaching a print head to a regular robot-arm, allowing for high diversity in print speed and strategy (Anderson, 2015). Besides, the Slovakian company BetAbram aims to bring Contour Crafting like printers to the market, by developing varying sizes of printers and printable concrete.

1.1.2. Concrete Printing

Concrete Printing (CP), a technique developed at the Department of Civil and Building Engineering of the Loughborough University, is similar to CC in their extrusion based construction process. This technique however, has a smaller resolution of deposition to achieve higher 3-dimensional freedom, as it allows greater control of internal and external geometries (*Figure 1.3 right*). One of the by-products of this process is the ribbed surface finish, as the resulting surface is heavily dependent on the layer thickness (Lim, et al., 2011).

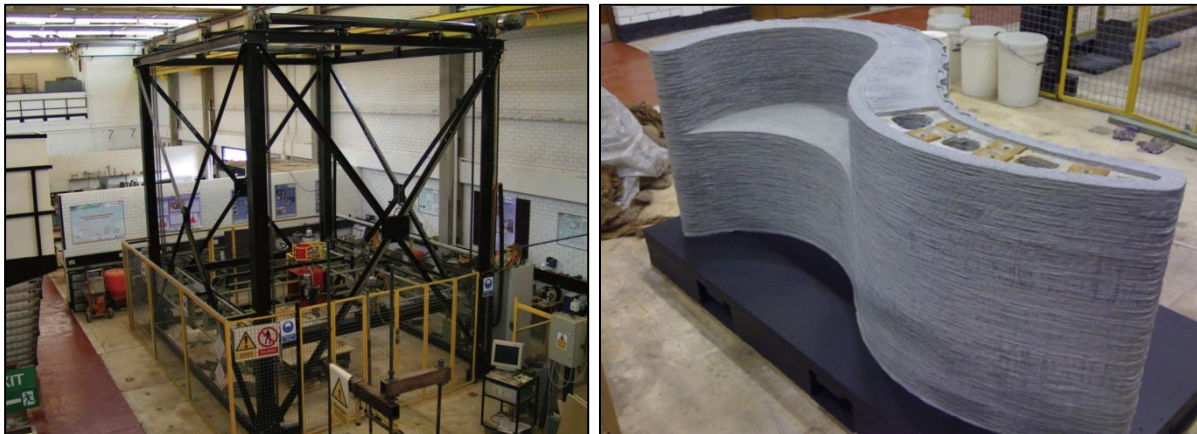


Figure 1.3 - Concrete Printing by the University of Loughborough (Lim, et al., 2011)

A high-performance printing concrete has been developed for the CP method, based on not merely target hardened properties (compressive strength of 100 MPa and flexural strength of 10 MPa at 28 days), but also workability, extrudability and buildability requirements. A fibre reinforced mixture has been designed, containing superplasticiser and retarder to increase workability and strength (Le, et al., 2011). Varying tests on printed concrete have been carried out, summarized in Annex A.

1.1.3. D-Shape

D-Shape is a 3D printing process, developed by Enrico Dini, which uses powder deposition selectively hardened by locally applying a binder material. Layers of sand are laid down to the desired thickness and compacted. A printing head composed of 300 nozzles, mounted on a gantry aluminium frame as seen in Figure 1.4, is then moved over the printing area and deposits the binder where the part is to be solid. Once completed, the part is dug out of the loose powder bed (Lim, Buswell, Le, Austin, Gibb, & Thorpe, 2012). The unhardened sand acts as a temporary support for the layers above, which allows for shapes that cannot be created by a single-material layer extrusion (*Figure 1.4 - D-Shape by Enrico*

Dini (Figure 1.4 right). A disadvantage of this technique however, is that the sand has to be spread and compacted for each layer. Once the element is completed, all the unused sand has to be removed.



Figure 1.4 - D-Shape by Enrico Dini (dinitech.it, 2014)

A similar technique is applied by Emerging Objects (EO), a subsidiary of Rael San Fratello Architects. EO uses a fibre reinforced cement mixture with small-sized aggregates, applying adhesives to improve workability of the mix. This printing method uses two types of binder: an alcohol-based binder, which is a water-soluble synthetic polymer that has high adhesive and mixing properties and high tensile strength. It will help the mix cure more rapidly and cause it to be denser and have greater flexural strength. A secondary binder is added to further strengthen the material, by both hydrating the material and joining the fibres to the concrete mixture. The result is a hybrid concrete polymer (Rael & San Fratello, 2011).

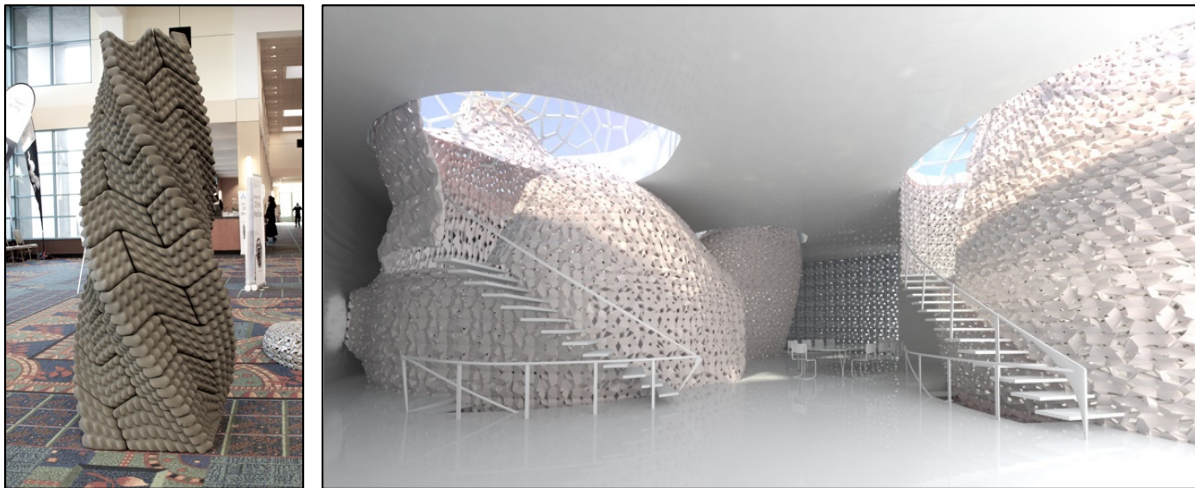


Figure 1.5 – Quake Column and 3D Printed House concept by Emerging Objects (emergingobjects.com, 2014)

1.2. Research Goal and Methodology

The state-of-the-art of concrete printing showed that the pioneering companies have been developing their printing technique and scaling up the printed objects. As these objects get larger, they generate more (media) attention, but also raise questions. The printed elements are after all often similar to the walls that have been printed by the initial techniques like Contour Crafting. A diverse application of concrete printing in the built environment is still limited.

This can be explained by lack of knowledge on the components involved in 3D concrete printing. The printed shape, the printable material (concrete), the forces working on the object and the printing technique itself are clearly connected, but their relationship are generally unknown (*Figure 1.6*). When a design has been made, the object is created with a trial-and-error attained printing strategy. However, once the design changes or new materials become available, the process has to start all over again and the printer settings have to be varied in an exhaustive way to find these new, proper attributes. With the high amount of variables involved, the time required increases dramatically and may slow down the development of the concrete printing technique.

This graduation thesis aims to develop a method to explore the relationships between the components involved in 3D concrete printing in a smart way. By creating an efficient link between these components, the influence of the 3D printing process can be shown for changing variables, without a time-consuming trial-and-error approach. This method will guide future research and experiments on the concrete printing technique.

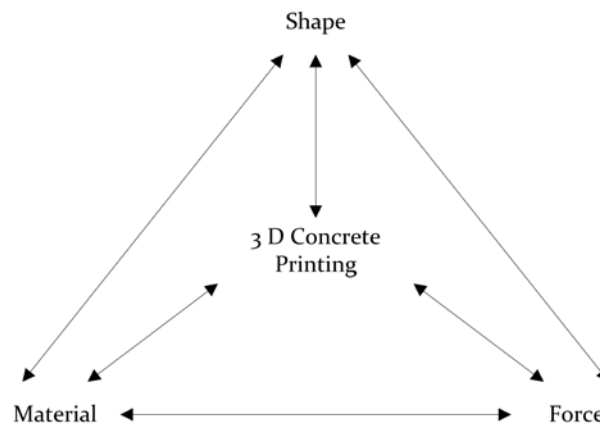


Figure 1.6 - 3D Concrete printing components

The number of variables involved in concrete printing and their value range results in a vast amount of combinations and variations, unable to provide useful solutions, let alone optimum settings, without an extensive amount of calculations and time. A smart optimization algorithm is essential, incorporated in a parametric model such that a high adaptability of the properties and the relationships between them can be achieved. Besides, the starting point can be a designed shape, but may also be material properties or a printing strategy. For this reason, a Reversed Engineering approach is required, allowing printing to be studied from multiple perspectives.

3D Printing allows for complex designs, like shown in *Figure 1.7-left*. The behaviour of these structures can't be analysed, much less optimized, if the underlying basic relationships are not understood. This thesis restricts itself in the shape component: a structural shear wall will be considered, assumed to be orthogonal and loaded in- and out of plane (*Figure 1.7 right*). Furthermore, the study on current state of

concrete printing has showed little available data on printed material properties. Thus, at certain points throughout this thesis assumptions are made in line with the expected behaviour of 3D printed concrete, and marked as such.



Figure 1.7 – Freeform to-be-printed wall (left) and simplified printed shear wall as studied in this thesis (right)

1.3. Thesis Outline

The work done during the graduation project is presented in this thesis, which started off with a summary of the state-of-the-art of concrete printing in this chapter. The summary is part of a more elaborate study on the concrete printing technique, which can be found in Annex A. In addition, general 3D printing methods are described and compared in this annex.

The research model that has been designed to study the relationships between the different concrete printing components, and vary their properties, is discussed in the second chapter of this thesis. This includes the general approach, along with a more in-depth paragraph on the FEM implementation of printed structures. The structural behaviour is benchmarked to theory, of which an extended overview is found in Annex B. The chapter concludes with a discussion on the programming environment used in the model.

The third chapter discusses the techniques available to find optimum solutions by applying smart algorithms. In specific, the theory behind the simulated annealing algorithm is explained and the required adaptations to incorporate it into this research are shown.

Chapter four focuses on how the printing strategy influences the properties of the printed element, specifically aimed at printing speed. The fifth and final chapter uses this to show the impact of 3D printing, by discussing the results of optimization analyses using varying printing strategies. This is the basis for an in-depth discussion on the future developments of 3D concrete printing and recommendations for additional topics to be studied.

Screenshots and a brief user-manual for the parametric model in Grasshopper are found in Annex C. Additionally, the Python script that controls Abaqus and contains the optimization algorithm is included in Annex D.

2. 3D Printing Research Model

The first chapter discussed the necessity for a method to study the concrete printing technique. This chapter will show how a research model has been constructed for this purpose. It will elaborate on the choice of the models' components and how they are related to each other.

2.1. Research Method

The structure of the research model can be depicted as seen in Figure 2.1. The core of the model is a structural analysis [1]. This analysis module does not restrict itself to predefined shapes, materials or loading types. Because of the large amount of variables used for this analysis, and the corresponding extensive output, the structural analysis module is placed in a parametric environment [2]. The input is highly adaptable and expendable with new data, while the output remains clear and understandable in a graphical way.

Due to the comprehensive input component, finding the required output may require a high amount of luck, computational time, or both. It is thus desirable to find the input settings in a smarter way, by varying the input based on the evaluation of the results. This is achieved by applying an optimization loop [3].

Finally, the model is extended with a variational loop and used to evaluate the influence of the 3D printing technique [4]. The printing properties are varied in a stepwise manner, entering the optimization module for each step.

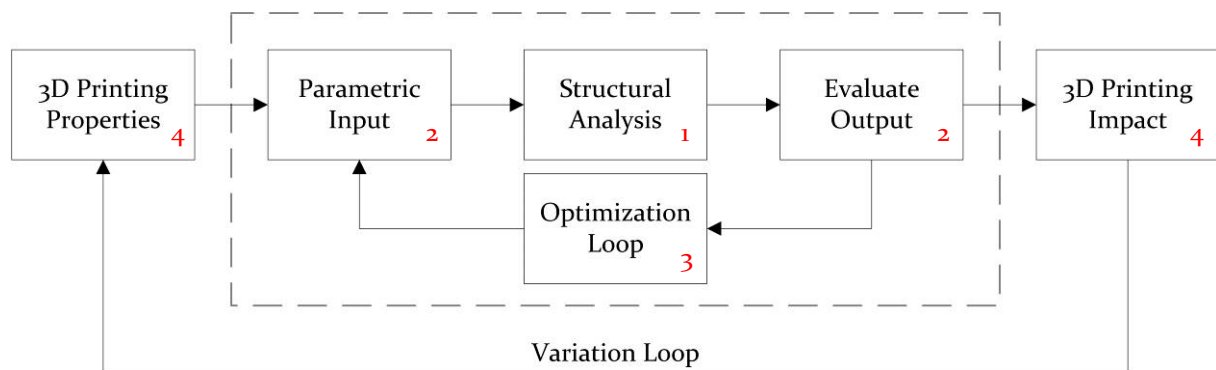


Figure 2.1 – Research model

Printable material, applied forces, printed shapes and print technique (Figure 1.6) are all connected in the parametric environment. As the starting point of the research on printing can be any of these components, the model is constructed based on a Reversed Engineering approach. Each of them can be selected as a variable, fixing the others at predefined values. The chosen component is then optimized, but may just as easily be swapped with a variable from a different component, without the need to reconstruct the entire model. This allows for quick evaluation and variation of parameters, whose behaviour is still unknown.

2.2. FEM Implementation of Printed Structures

The structural analysis component of the research model is realised in FEM software Abaqus. Abaqus does not restrict itself to certain shapes as it uses advanced meshing techniques and has an extensive element type library which allows for varying materials properties and analyses. This makes the FEM analysis a very adaptable, yet reliable component in the research model.

The studied orthogonal wall is assumed to be part of a structural framework (*Figure 2.2 left*), which results in an in- and out of plane loading applied on the wall. The wall is taken out of its framework (*Figure 2.2 right*) and modelled in Abaqus, using conventional shell elements which incorporate both shell and plate behaviour. The loads from the framework are applied as a line load Q_L and bending moment M_Q on the top edge, along with a (wind-) pressure P on the face of the wall.

The following printing behaviour is incorporated in Abaqus, discussed stepwise in this paragraph:

- Orthotropic behaviour
- Non-linear elastic materials
- Sandwich elements
- Aging properties

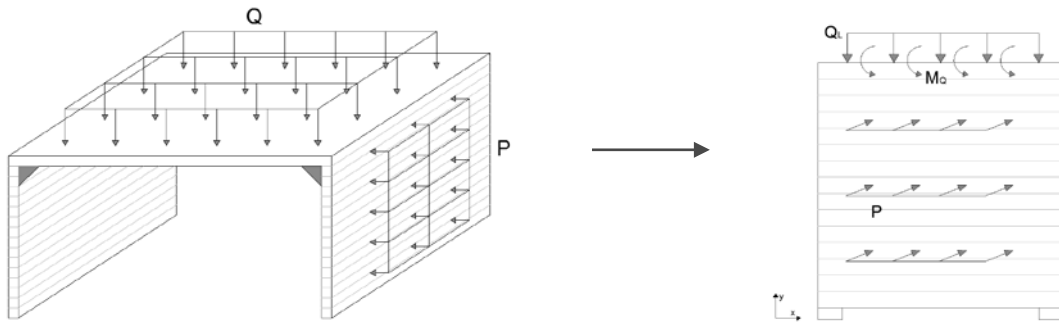


Figure 2.2 – Structural framework (left) and simplification (right) of the studied printed wall

2.2.1. Orthotropic Behaviour

The printing technique results in a layered build-up of the wall: properties from layer to layer will vary, compared to in-layer behaviour. Hence, a certain orthotropic behaviour must be achieved in the model. As the studied object is an orthogonal wall, the mechanical properties in x direction will be different from those in y direction.

Abaqus allows orthotropy to be defined for shell elements by giving the elastic and shear moduli in these two principle directions. This results in dissimilar stiffness behaviour. Because of the bond between layers, it is desired to also have different strengths (i.e. failure stresses) in two directions. The limits of Abaqus restrict the implementation of the required orthotropic material behaviour caused by printing, without the need to write a user defined material model. A different approach is therefore implemented. Once an analysis has been carried out, a small loop algorithm checks the occurring stresses in each direction for each element of the structure. It writes the occurring maximum stresses for both compression and tension in two directions and evaluates if these values exceed the predefined orthotropic strength properties. In case these stresses do exceed the limit, the user is notified of the occurring failure and its location.

2.2.2. Non-Linear Elastic Materials

Considering future developments of 3D printing, the use of new types of printable concrete is not unlikely. Traditional types of concrete have a low tensile capacity and the application of reinforcement by 3D printing is far from being self-evident. The incorporation of fibre reinforcement may be an interesting alternative, as it increases the ductility of the material (softening) and may even provide a higher tensile capacity (hardening), recognized from Figure 2.3. These graphs show the load (vertical axis) versus deflection (horizontal axis) with post cracking behaviour of a concrete beam loaded by bending. Once cracking is initiated, plain concrete quickly fails as depicted by the dashed line. Applying fibre reinforced concrete (FRC) results in a post-cracking behaviour where the deformation capacity increases as the fibre content grows: the material has become more ductile.

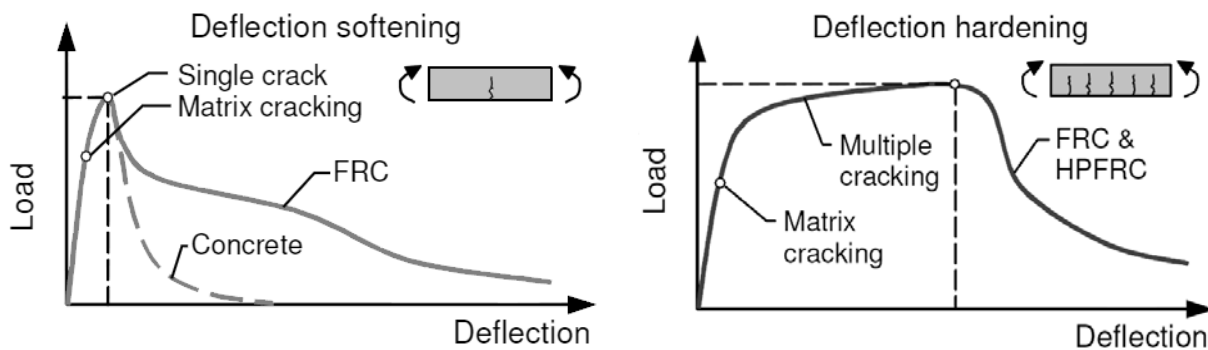


Figure 2.3 – Flexural behaviour of fibre reinforced concrete (Löfgren, 2005)

Thus, to study new types of printable concrete, the FEM model has to be able to incorporate non-linear elastic (NLE) material behaviour. This can be achieved by using a Smeared Cracking approach, which is a concrete model of Abaqus that includes the presence of cracks by affecting the stresses and material stiffness in the analysed element. The stress-strain diagram of concrete in both tension and compression can be given, which allows for a variation of fibre content or concrete types to be studied with this model.

However, when considering this simplified orthogonal wall it is clear that no redistribution of forces is possible, as it is a statically determined system. Without high amounts of fibre reinforcement (and thus hardening), there is little to no extra plastic capacity once the linear-elastic limit is reached. This is a consequence of the choice made in the early stage of the graduation research, to focus on a simplified orthogonal wall. For this thesis, the impact of 3D printing will be discussed based on solely linear elastic properties.

2.2.3. Sandwich Elements

It can be beneficial to print sandwich-like structures, which have an efficient mass-stiffness ratio. This cross section build-up is therefore also incorporated in the model, but not yet studied for this thesis. The advantage of sandwich structures over solid cast walls has already been proven for orthogonal shapes and is not the goal of this study. When future research focuses on freeform printed structures, the use of sandwich cross sections will be of more interest and can be incorporated in a research model like presented in this thesis.

2.2.4. Aging Properties

The aging properties of printed concrete are likely to differ from traditional cast concrete. However, the structural wall for this research is considered as 'wished-in-place'. This means that the wall is analysed as if it was printed with a certain strategy, without simulating the layered construction process. The aging behaviour of concrete is therefore not modelled in Abaqus, but incorporated in the mechanical properties that are applied to the wall.

2.3. Programming Environment

A parametric setup is desirable for the research model to quickly vary properties and evaluate their relations. In an early stage, this led to the choice of Grasshopper (GH), a graphical algorithm editor, which comes as a plugin for Rhinoceros. The large amount of parameters can be categorized in material properties, geometry, and loadings and boundary conditions. All of these are modelled in Grasshopper, of which an example can be seen in Figure 2.4. An extended overview of the GH model can be found in Annex C.

The parametric model allows the user to select LE or NLE material behaviour, and a homogeneous or sandwich cross section, by means of a drop down menu. Orthotropic properties are automatically taken into account. Besides, the user has a choice in type and location of supports, based on the assumed framework. The structural engineer will have to decide if these can be modelled as fixed or clamped, and if the lateral loading is transferred to shear walls (left/right support) or floors (top/bottom support).

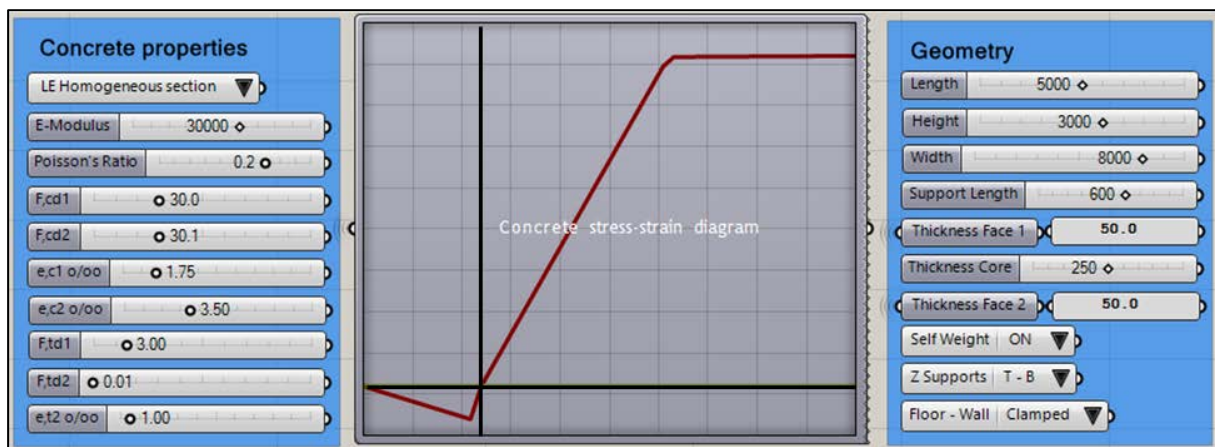


Figure 2.4 – Example of parametric input module in Grasshopper

Grasshopper is a propagation-based system, which restricts cyclic algorithms (i.e. 'loops') without additional plugins. Both the optimization algorithm and the variational loop of the printing properties are therefore written in programming language Python, as the limits of GH were reached.

Additionally, the Python language can be used to control Abaqus. A script is entirely generated in Grasshopper, including all parameters, optimization targets and output requests. This script is then simply sent to Abaqus, which executes both the structural analysis and the optimization loop. In real time, results are sent back to GH, allowing the user to keep track of the progress and optimized results. These results consist of the (maximum) occurring stresses and deformations of the analysed wall, along with their nodal location. Due to the sharp edge between wall and support, some peak stresses may still arise, even though the stiffness of the supports is taken as close to zero. The Python script is extended with an option to filter out these peak stresses, based on a user defined area close to the supports.

3. Optimization Algorithm

The previous chapter has shown how the FEM analysis provides the research model with stresses and deformations, for varying shapes, material behaviour or loading types. This chapter will explain the algorithm that uses these calculated values to search towards efficient combinations of parameters, without having to carry out an extensive amount of calculations.

3.1. General Optimization Methods

When presented with a problem to be studied, that has an extensive amount of variable parameters, the first thing to come to mind is to start varying the parameter values until all settings have been analysed. The results are then evaluated and the setting which comes closest to the goal is chosen as the optimum one. This so called ‘exhaustive search’ doesn’t require a complex algorithm: cut the domain of the input parameter in n pieces, then analyse and evaluate those n times. But as soon as the number of variables increases, the possible combinations and thus required computational time grows drastically.

A smarter searching algorithm is desirable. Instead of carrying out all calculations and then evaluating and choosing the best result, the algorithm should guide the selected input based on previously acquired results. Once a certain value has been evaluated, a new one is chosen from the neighbourhood of the previous value. If this new point is better, it is chosen as the new starting point. Otherwise, a new point from the neighbourhood is selected. This method repeats itself until no further improvement is achieved. This method is known as the ‘hill climbing algorithm’ and even though it is an improvement compared to the exhaustive search, it still has several downsides. It usually terminates at local optima and the algorithm gives no information on how much this local optimum deviates from the global optimum (*Figure 3.1*). Besides, the optimum found greatly depends on the initial configuration of the neighbourhood, i.e. the chosen initial domain (Michalewicz & Fogel, 2000).



Figure 3.1 – Example of a hill climbing algorithm ending at a local optimum

As mentioned before, in case of 3D concrete printing there is still much unclear. When evaluating a chosen parameter and searching for a certain goal, it is initially unknown how the range of the parameters’ values is related to the optimum value. Moreover, the evaluation function, i.e. the way the chosen values relate to the optimum (the fitness of each solution), is usually strongly non-linear and may contain local optima. Fortunately, optimization algorithms have been improved over the past years. Methods have been developed that take these issues into account. The simulated annealing algorithm will be shown to be very suitable when studying concrete printing.

3.2. Simulated Annealing

The simulated annealing (SA) algorithm was inspired from thermodynamics and metal work. Annealing involves heating and cooling a material, altering its properties by changing its structure on molecular level. Once the material is cooled down, this new structure is locked in, along with its newly obtained properties. If the temperature is dropped too quickly, irregularities may occur which are trapped in the newly created structure as well.

The SA algorithm has a fictitious temperature variable, similar to this heating and cooling process. This variable starts high, and the slowly ‘cools down’ as the algorithm runs. At high temperature the algorithm is often allowed to accept worse solutions than the currently best one, more or less similar to a random search. It is therefore able to jump out of local optimums in the early stage of execution, like depicted in Figure 3.2. The chance of accepting worse solutions is reduced as the temperature drops, i.e. as the algorithm has completed more iterations. This allows the algorithm to narrow the search space, focusing on what is hopefully a global optimum solution (Jacobson, 2013).



Figure 3.2 – Example of a simulated annealing algorithm jumping out of a local optimum

Figure 3.3 compares a local search method of hill-climbing with the SA algorithm. Note that both algorithms are strongly simplified for comparison. The first difference is the end of the procedure: a termination condition has to be satisfied for SA, while the hill-climbing method requires an improvement to be found, or else aborts. Secondly, the “improve?(x , T)” function doesn’t necessarily have to return a better solution; it just returns an accepted one. Finally, as mentioned before, the parameter T for temperature is incorporated in the algorithm, being updated for each iteration in the algorithm (Michalewicz & Fogel, 2000).

<pre> procedure local search begin x = some initial starting point in S while improve(x) \neq ‘no’ do x = improve(x) return(x) end </pre>	<pre> procedure simulated annealing begin x = some initial starting point in S while not termination-condition do x = improve?(x, T) update(T) return(x) end </pre>
--	--

Figure 3.3 – The difference between a local search and SA algorithm (Michalewicz & Fogel, 2000)

The procedure of simulated annealing as used in the research model is given in Figure 3.4. Each function of the algorithm will be explained stepwise based on this flowchart.

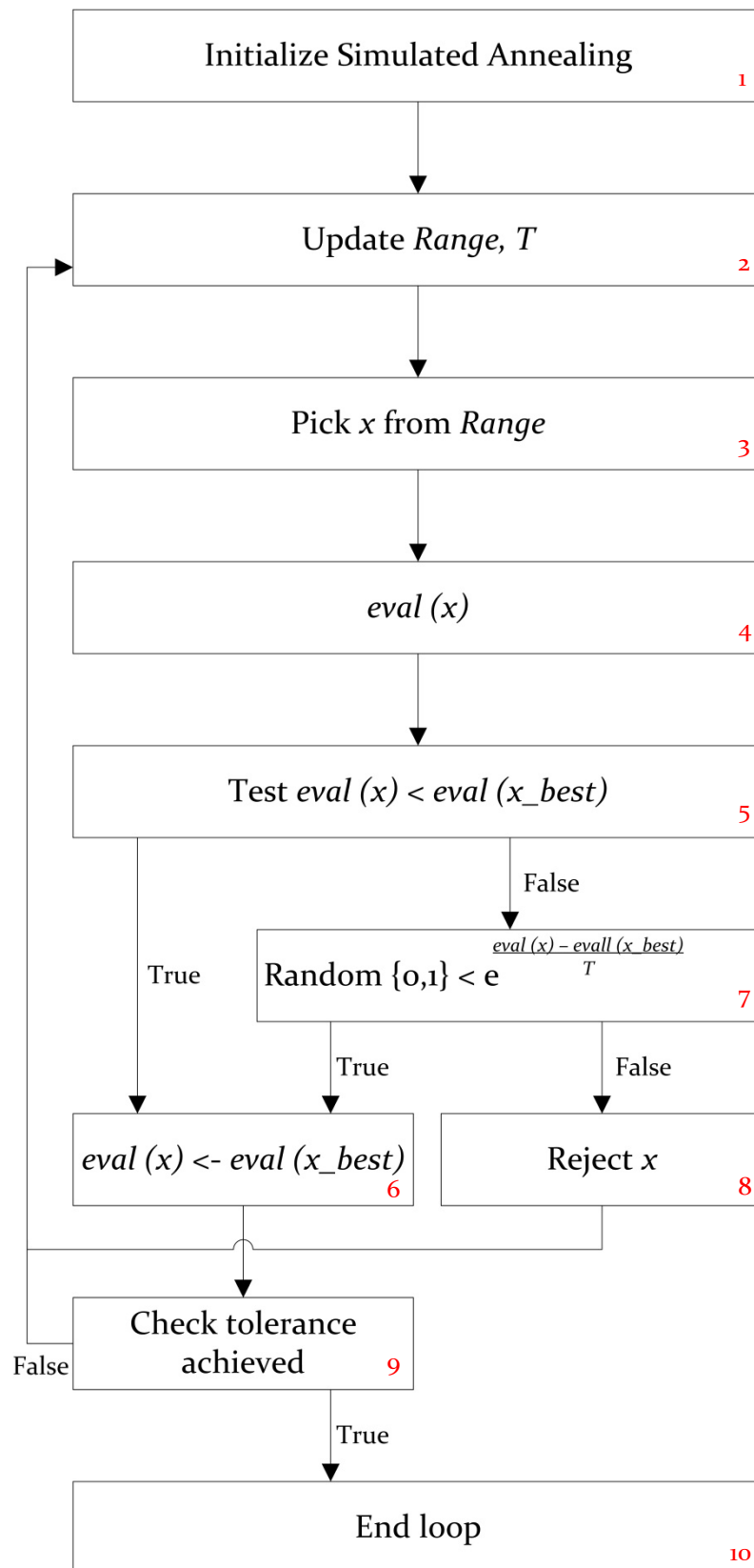


Figure 3.4 – Structure of simulated annealing

As the simulated annealing algorithm gets initiated [1], the variable and goal of the optimization loop have to be fixed, along with a certain number of parameters. The target module in the Grasshopper model allows the users to choose the parameters of interest. The variable to be optimized can be the loading on a wall (in- and out of plane), geometry, and cross-section or material properties. As this study is aimed at single-target optimization to prevent an over complexity of the model, a choice in goal has to be made. This can be either compressive or tensile stresses, in orthogonal directions, or displacements in- and out of plane. An example can be seen in Figure 3.5.

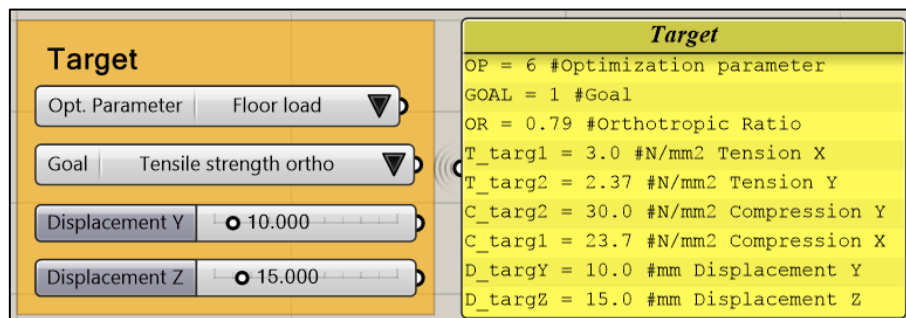


Figure 3.5 – Target module in Grasshopper

Next, the algorithm parameters are chosen (Figure 3.6). The maximum amount of iterations is fixed, and so is the initial domain and temperature [2]. The change of domain has to be determined, of which the influence will be discussed later on in this paragraph. Finally, the tolerance is set. Once the algorithm has found a solution that is within this range of accuracy, it terminates the loop.

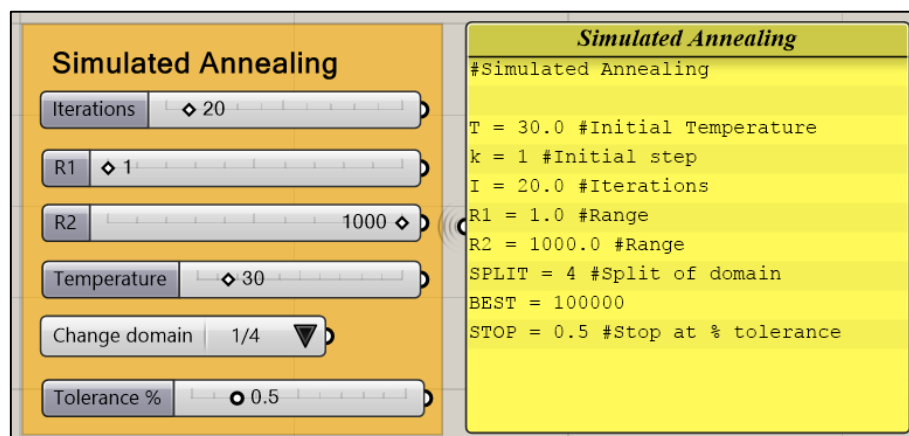


Figure 3.6 – Simulated Annealing module in Grasshopper

The algorithm is now initialized and chooses a random point x within the domain $\{R_1, R_2\}$ [3]. The value x is then evaluated by the function $eval(x)$, which corresponds to a structural analysis in Abaqus [4]. This analysis will show the maximum occurring stresses or deformations and compares them with the predefined goal. The result of this evaluation function is compared to the currently best solution [5]. If the occurring result is indeed better, $eval(x)$ becomes the currently best solution $eval(x_{best})$ [6].

Up to this point, the algorithm has followed a procedure similar to hill-climbing methods. However, the next steps will show that SA continues in a different way.

When $eval(x) < eval(x_{best})$ returns a *False* statement, a new function is called upon. This part of the algorithm determines if a worse solution is accepted or not, based on the difference between the current

and best solution, and the newly introduced parameter T . The probability of acceptance is chosen as a random value in the domain $\{0, 1\}$ [7]. This number is compared with the evaluated point in Python as follows:

```
p = math.exp((eval(x) - eval(x_best)) / T)
if random.random() < p:
    eval(x_best) = eval(x)
```

Once this statement returns *True*, the value is accepted as $eval(x_best)$ [6]. When it returns *False*, the solution is discarded [8].

After these steps, the solution being either better or worse, the algorithm will return to its starting point and repeat the same steps. Before doing so however, it checks a termination condition; in this case the required tolerance of the solution [9]. If the tolerance is achieved, the algorithm ends, showing the result, the corresponding best value of x and the tolerance percentage of the solution [10].

If the tolerance is not achieved, the temperature T is updated before restarting the algorithm. As mentioned before, this parameter influences the probability of accepting a worse solution. The nature of this update function can be varied: both the order of the function and the size of steps may vary between analyses. Subsequently, the choice of function strongly influences the probability of accepting worse solutions. In case the analysed problem is largely unknown or contains a high amount of local optima, it may be desired to slowly decrease the temperature during the run. On the other hand, when the problem is well-known, the temperature function may drop rapidly such that the algorithm converges quickly to the optimum solution.

For the implementation in the research model used in this thesis, the temperature function is taken as a linear decreasing function. Each step k in temperature drop is therefore equal, but their size can be varied by selecting the total number of iterations I . The resulting temperature function in Python language is shown below.

```
def update_temperature(T, k):
    return T_int - (k * (T_int / I))
```

To study the concrete printing technique an additional functionality has been added to the algorithm. The behaviour of the parameters of interest and the relationship between them is generally unknown. For this reason, the initial domain $\{R_1, R_2\}$ is taken very large at first, to prevent a too small choice of range that doesn't incorporate the optimum solution. However, as the algorithm runs, the problem gradually becomes clearer. Keeping the domain large would result in a time consuming optimization run, which negates the benefit of this smart algorithm. Thus, at a user defined point in the loop, the algorithm will narrow the domain $\{R_3, R_4\}$ step by step based on the current best solutions. This allows the algorithm to converge to an optimum solution much faster, even when the problem is initially unknown:

```
if k <= I/(SPLIT):
    return range(R1, R2)
elif k > I/(SPLIT):
    return range(R3, R4)
```

The values of R_3 and R_4 are based on the best and second best known solution, which continuously update during the run, as the figures below illustrate. The horizontal axis concerns the domain, in which a parameter x may be chosen. The vertical axis lists the fitness of each solution $eval(x)$. The optimum solution corresponds with an unknown optimum variable $x_{optimum}$: the target of the algorithm.

Figure 3.7 shows that if this expected optimum is somewhere in between the currently best two solutions (x_1 and x_2), i.e. the best evaluation value is below the target and the second best is above or vice versa, the range will be straightforwardly reduced.

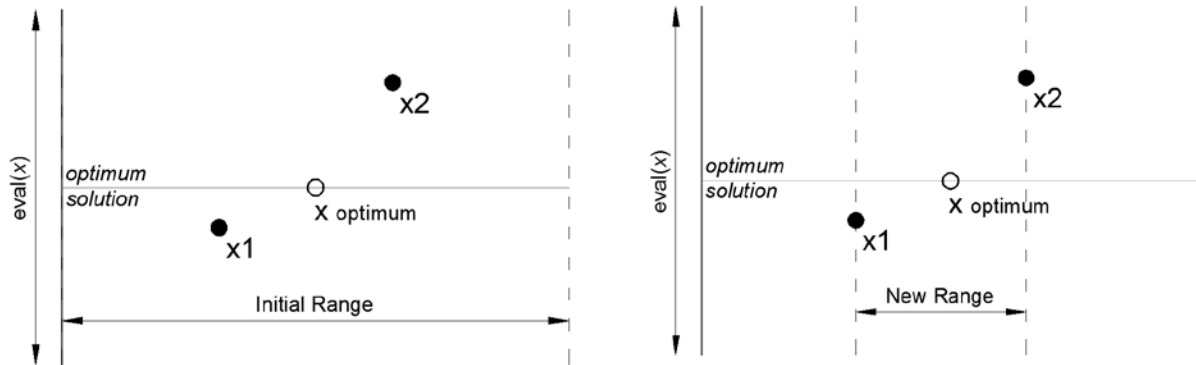


Figure 3.7 – Reduction of range, using the two best known evaluation values on two sides of the optimum

However, if the initial range was chosen too small and both values are on one side of the target, the new range is less obvious. Figure 3.8 shows how the algorithm uses the distances between the two best values (x_1 and x_2), to compute the trend of the solutions. This can then be used to estimate the distance (Δx) between the currently best value and the optimum one. However, as this assumes the problem to be linear, the range is taken as twice this distance ($2\Delta x$) to incorporate for non-linear optimization problems.

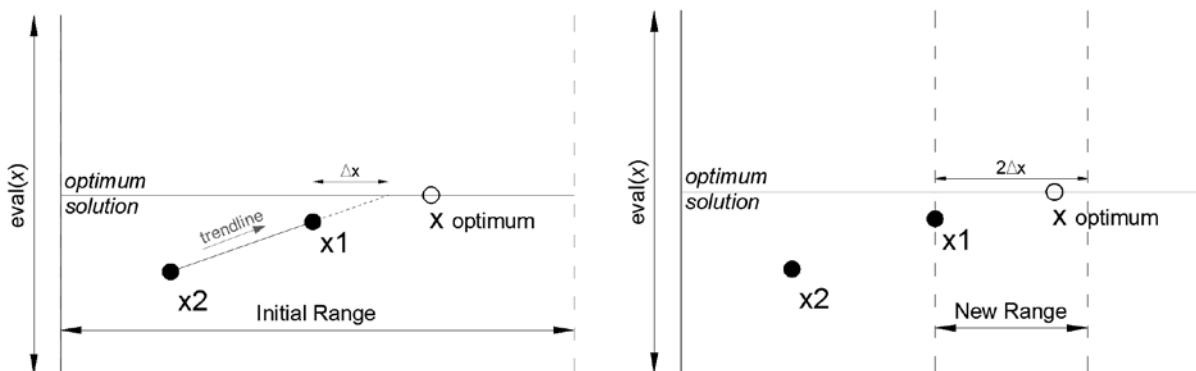


Figure 3.8 – Reduction of range, using the two best known evaluation values on one side of the optimum

It has to be noted that when using optimization algorithms, the balance between computational time and accuracy comes into play. Besides, as seen when evaluating the SA algorithm, accepting solutions is partially based on chance. Especially in case of largely unknown problems, as is this case for this study, either much iterations and exact solutions, or a quick evaluation and somewhat inaccurate solutions may be the result. For this early stage of research the latter is preferred.

4. 3D Printing Parameters

The previous chapter has shown how the research model uses an optimization algorithm to find suitable solutions, even with a large number of unknown variables. However, some basic printing parameters have yet to be included, so that by varying them the influence of 3D printing can be analysed and presented. This chapter will show which printing parameters have been chosen to analyse and how they are incorporated in the research model.

4.1. Printing Strategy

Despite the differences in the concrete printing techniques, finally they all end up with a layered end result. The bond strength between these layers depends on the printing time gap, i.e. the time it takes for one layer to be printed on top of the previous one. Research at the Loughborough University has shown that the bond strength decreases as the time gap rises. Figure 4.1 shows this time gap up to 7 days on the horizontal axis and the corresponding bond strength between layers on the vertical axis. Considering bond strength it can be stated that higher printing speeds are preferred, to achieve better mechanical properties.

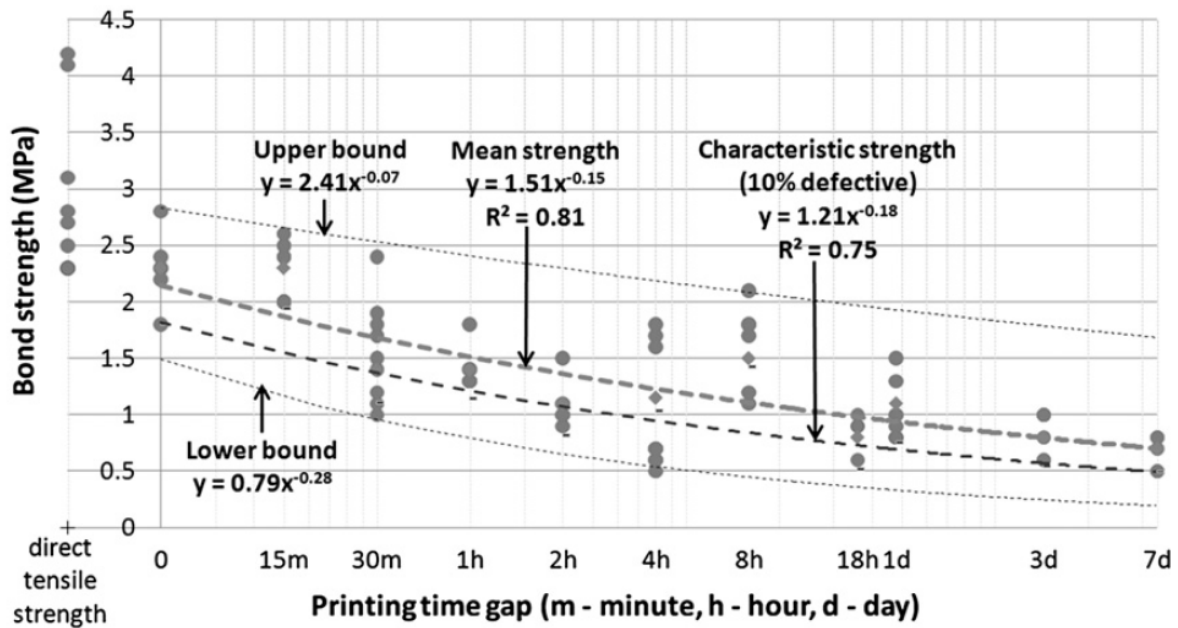


Figure 4.1 – Bond strength between layers vs. Printing time gap (Le, et al., 2011)

Printed objects (either structural elements or entire buildings) will be loaded much sooner than traditionally cast structures. The strength development in time is therefore interesting, as it defines a relation between printing strategy and the minimum time at which a certain loading can be applied.

The chosen printing speed, together with the layer size, determines how long it will take to complete the printed object. The height of the element or building is divided in a number of layers, based on the print nozzle dimension. The total printing time is then found by multiplying the printing time gap by the amount of layers. This time can be used to calculate the maturity of concrete, which is used to compute the developed strength at a certain time after printing.

Mathematical functions have been developed based on the product of time and temperature (i.e. maturity), such that the strength of concrete can be determined. These strength-maturity relations are often defined by a logarithm equation, first proposed by Plowman (1956):

$$f_c = a + b \log(M)$$

where,

f_c = concrete compressive strength [N/mm²]

M = Maturity index

a, b = constants based on cement type and water-cement ratio of mixture

For the research model used in this thesis, the maturity index is computed by a function based on the Arrhenius equation. The key parameter in this equation is the apparent activation energy E , which describes the effect of temperature on the rate of strength development (Carino & Lew, 2001):

$$M = \sum_0^t e^{\frac{-E}{R} \cdot \left(\frac{1}{T} - \frac{1}{T_r} \right)} \cdot \Delta t$$

where,

M = the equivalent age (Maturity index) at the reference temperature and time,

E = apparent activation energy [J/mol],

R = universal gas constant, 8,314 J/mol.K,

T = average absolute temperature of the concrete [K], and

T_r = absolute reference temperature, 293 K.

The activation energy is linked to the temperature at which the reaction takes place. A high temperature result in less energy required to active the reactions, which is formulated by Freiesleben-Hansen and Pedersen (1977) as follows:

$$E = 33500 + 1470(293 - T), \text{ if } T < 293K$$

$$E = 33500, \text{ if } T \geq 293K$$

If the compressive strength based on the concrete maturity is known, and the relation between compressive and tensile strength development is presented, the tensile strength at a certain maturity can be computed as well. Many equations have been proposed for this reason, which generally take the following form (Ros & Shima, 2013):

$$f_{spt} = k \cdot (f_c)^n$$

where,

f_{spt} = split tensile strength [N/mm²],

f_c = compressive strength [N/mm²], and

k, n = non-dimensional coefficients

The coefficient n is smaller than 1, which results in a slightly faster development of tensile strength than the compressive equivalent. The same behaviour is recognized in Figure 4.2, showing the results of a study by Lew and Reichard (1978) on the early age mechanical properties of concrete.

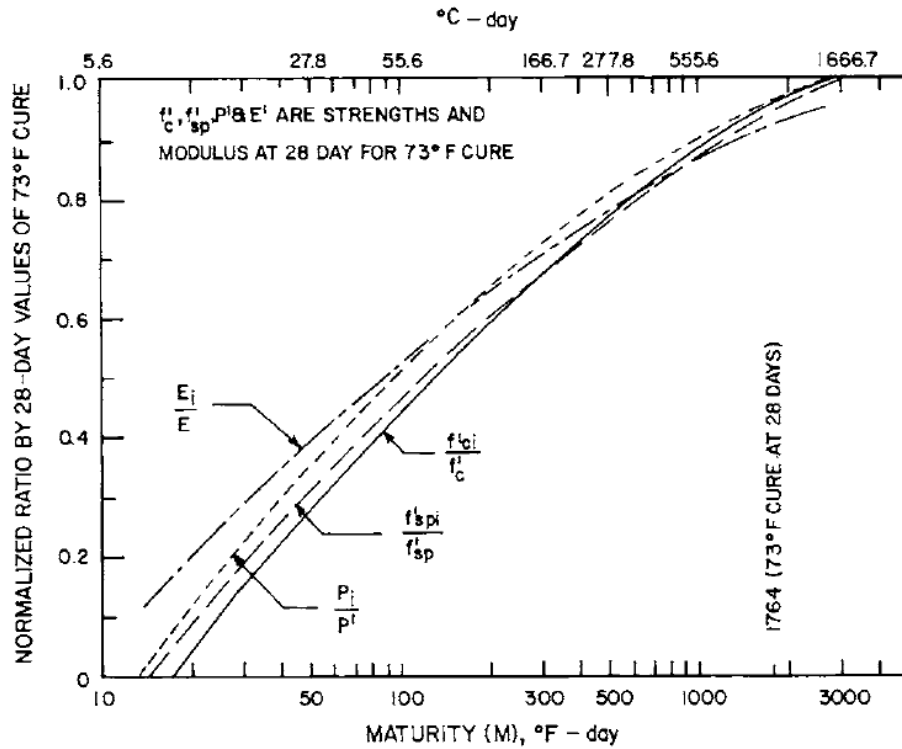


Figure 4.2 – Concrete maturity plotted versus strength and stiffness development for a concrete mixture with CEM I (Lew & Reichard, 1978)

Considering 3D printing it is clear that for both the compressive and tensile strength development, high printer speeds are not always preferred. Simply put, the faster the printed element is constructed, the less development of strength will be achieved once it is completed.

4.2. Model Input

Little data on mechanical behaviour of printed concrete has been presented. But as this thesis aims to give understanding on the influence of 3D printing, some relations will have to be assumed.

Another issue at hand becomes clear when considering the time span of the studied element. The structural wall that is analysed will be printed in a matter of hours, depending on the choice of print strategy (speed and layer height). Yet, most available data on strength development is published based on days or weeks. For this reason some assumptions have been made, based on the expected behaviour of printable concrete.

The strength development, usually related to 28 day strength, is now scaled down to a full development in circa 24 hours. The result is a logarithmic function for both the normalized compressive and tensile strength development, based on concrete maturity M , recognized from Figure 4.3-left. The vertical axis shows a normalized strength development from 0 to 1, where 1 corresponds to the maximum strength as defined by the user. The horizontal axis contains the maturity hours from 0 to 24 hours, based on a reference temperature of 20 degrees.

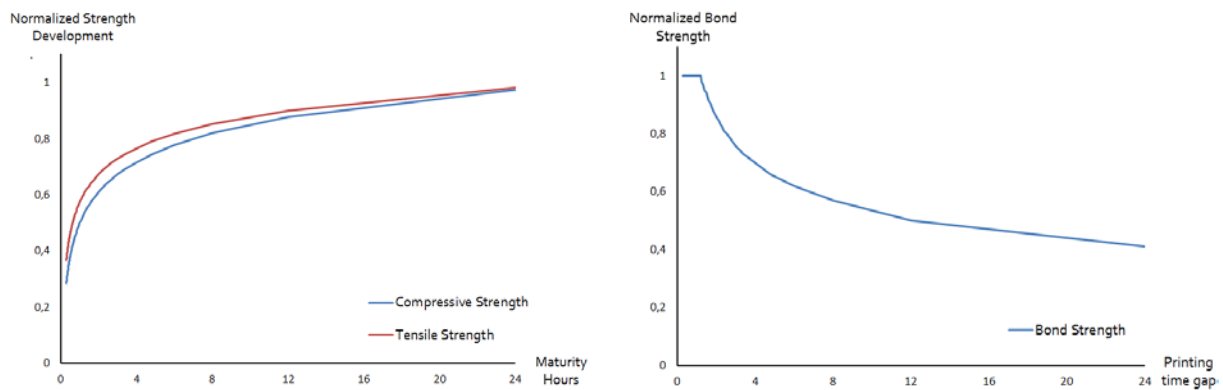


Figure 4.3 – Overall strength development (left) and bond strength between layers (right) versus time

The bond strength between layers is modelled by a function similar to the results found at the Loughborough University. The bond can be plotted versus the printing time gap, using a non-integer order polynomial function. Similar to the maturity function, it is normalized based on a 24 hour time span on the horizontal axis (Figure 4.3 right). The actual strength between layers is then found by multiplying the normalized bond strength ratio (vertical axis) by the current developed overall strength. A short plateau is recognized at small printing time gaps, as the bond strength is assumed not to reduce significantly at very high print speeds.

These functions are included in the Python script that is used to run the optimization algorithm. In both functions, the main variable parameter is printing speed as it is used to calculate the printing time gap and concrete maturity. The temperature that is used to calculate the maturity is taken as a constant value, based on the printing environment. In reality, this value will not be constant and strongly influenced by the curing process of the concrete.

All of these printer properties are included in the parametric Grasshopper model, where the maximum print speed, layer size and temperature can be fixed, see Figure 4.4. The amount of evaluation steps are then chosen, which determines in how many steps above functions are studied. For each step an optimization loop is carried out in Abaqus. The model is constructed in such a way, that each of these print properties may be chosen as the evaluated parameter.

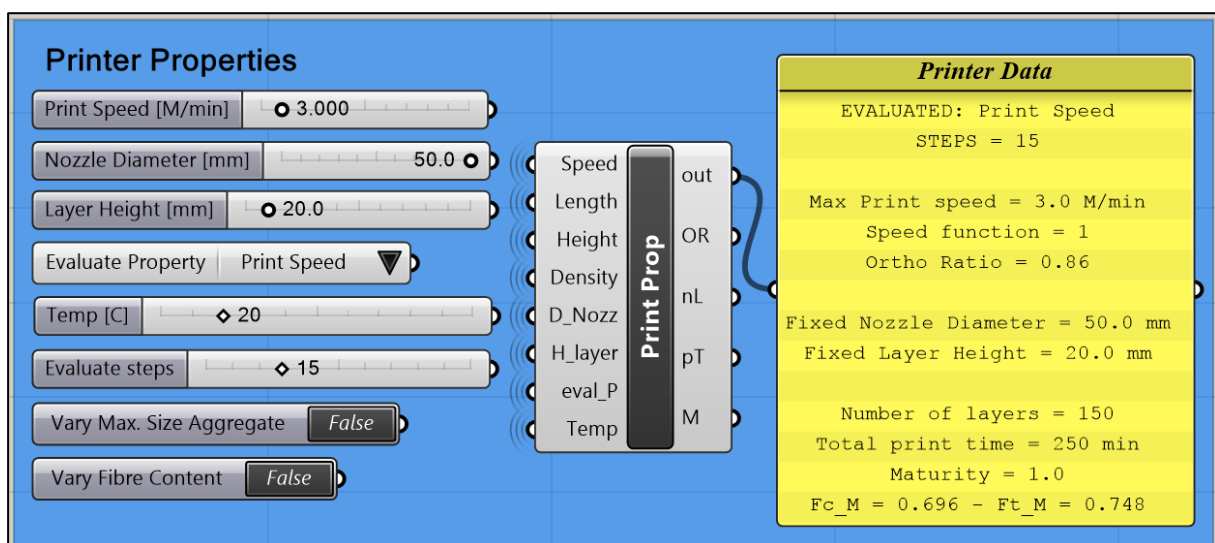


Figure 4.4 – Printer Properties module in Grasshopper

It is noted that it is assumed that the printed material is able to carry its own weight, and that of the layers above. Research has shown that this is not always the case, depending on the buildability (i.e. shear strength) of the concrete mixture (*Figure 4.5*). This behaviour may result in a maximum allowable print speed, to prevent the fresh layers to arrive too soon on the old ones. This however, is not incorporated in the research model of this graduation thesis.



Figure 4.5 – Printed layers collapsed by their own weight (Le, et al., 2011)

5. The Impact of 3D Concrete Printing: Conclusions and Recommendations

3D Printing of concrete structures is in an early stage of development: the potential of this innovative technique has been shown, but there is little research to support the method. This results in a time consuming and inefficient trial-and-error approach when one or more of the components involved in 3D printing are changed: shape, material properties, applied forces or print strategy.

The previous chapters presented a method that tackles this issue: a research model has been developed, which allows the printing properties and the relationship between the involved parameters to be evaluated, and optimized. This final chapter will show the impact of 3D concrete printing, by running optimization analyses with varying printer properties.

5.1. Optimization Analyses

The analysed printed wall is assumed to be part of a simple framework, consisting of 2 printed walls supporting a floor. One wall is thus loaded by a constant pressure out of plane (e.g. wind loading) and a variable floor load. The latter results in both a line load and a bending moment, applied at the top edge of the wall. This floor load will be the parameter that is optimized during the loop. The other properties, like geometry and boundary conditions, are fixed. The analyses are carried out using linear elastic material behaviour.

The most interesting printer property to vary for this structural element is the print speed, as it influences both the bond strength between the layers and the total strength development of the wall. The domain of the speed is set at 0 to 20 m/min and divided in 20 steps to be evaluated. Each of these steps will result in one optimized parameter value, plotted in the following graphs.

To compare print strategies, the evaluation of print speed is done for 5 different layer heights, going from 10 to 50 mm. Besides, an evaluation of different printing environments is carried out: in-situ printing (5 degrees) and controlled, 'prefab' printing (40 degrees). These are compared to the reference temperature of 20 degrees.

Because of the assumptions that have been made for this analysis, all values are normalized based on the results of the reference printing strategy: 50 mm layer height at a temperature of 20 degrees. This allows for an easy comparison of results, without getting the false impression that these exact values can be taken into practice.

Figure 5.1 and 5.2 show the results of the optimization analyses. The vertical axis shows the normalized loading capacity of the structure, which corresponds to a maximum allowable floor load Q . The horizontal axis concerns the normalized printer speed.

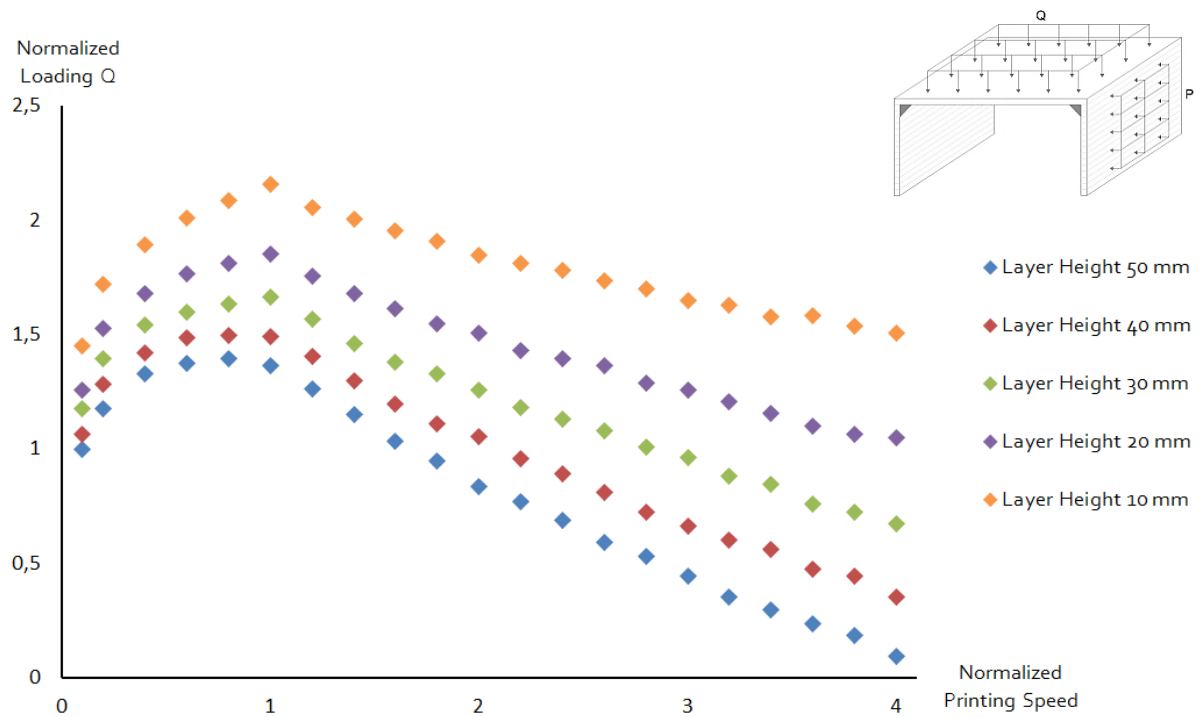


Figure 5.1 – Optimization results of printing speed versus loading capacity for varying layer height

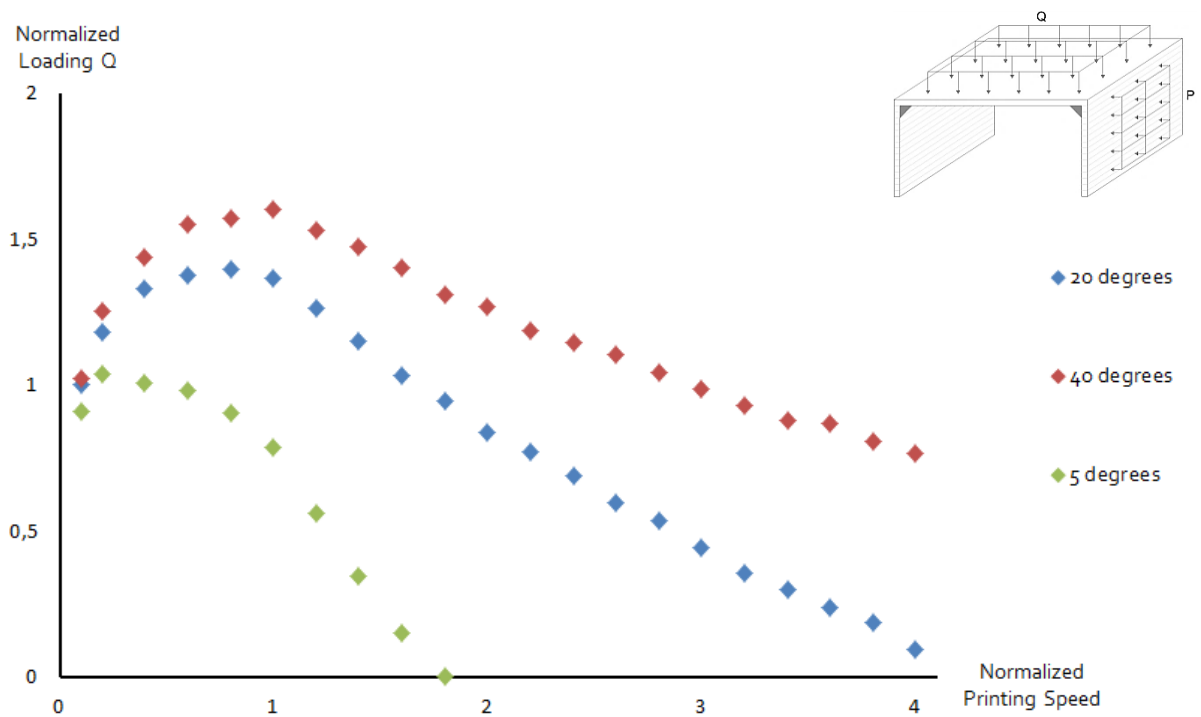


Figure 5.2 – Optimization results of printing speed versus loading capacity for varying temperature

5.2. Conclusions

The results of the optimization loops clearly show the impact of printing strategy on the loading capacity of the structural wall. At low printer speeds the bond strength between layers is governing, which gradually improves when the printer speeds up. However, once the printer reaches a certain speed, the loading capacity starts decreasing at higher speeds. As the wall is finished quicker and thus loaded earlier, its maturity is in an early stage. In other words, it might not always be sensible to print as fast as the printer allows, as the required strength may not yet have been developed. This same issue can also result in unexpected print paths. It might be beneficial to firstly print the load bearing elements, followed by the non-structural parts. This way, the strength development of the load bearing structure is ahead when higher loads have to be applied (for example, when a floor slab is laid down in the construction process).

The first graph shows that as the layer height gets smaller, the overall loading capacity increases. This effect is mostly recognized in the second part of the graph, where maturity is the governing system. At a constant speed, the layer size does not affect the bond strength between layers, but does strongly influence the total printing time and thus the strength development of the wall. Simply put, reducing the layer height is beneficial for the loading capacity of the wall. However, considering the process of automation that 3D printing provides, the overall printing time is also of interest. Reducing the layer height increases the total duration of construction, which is not taken as a target in this research but may be very important in practice.

The second graph displays that the construction environment has to be taken into account during the early design stage of the to-be-printed structure. When printing takes place in-situ (simulated by a temperature of 5 degrees), the strength development is much slower than in a controlled, prefab environment (i.e. 40 degrees). The capacity of the wall is shown to be much lower in the first situation at a certain print speed, compared to controlled environment printing.

These simple analyses on a printed structural wall have proven that the influence of 3D printing must not be underestimated and that the printing strategy has to be taken into account during each step from early design to construction.

The research model has proven to be able to evaluate the varying components involved in 3D concrete printing in a smart way. The modular and parametric environment are set up such, that they can easily be replaced or edited by fresh data on existing or newly developed print techniques, as the following years will provide much more (experimental) research on 3D concrete printing, specifically at the Eindhoven University of Technology. Accordingly, the following and final paragraph of this thesis will give some recommendations based on the findings of this graduation project.

5.3. Recommendations

This first chapter showed a graphical representation of the components involved in 3D concrete printing (*Figure 5.3*). This final paragraph will use it as a basis for recommendations on future research on each of these components.

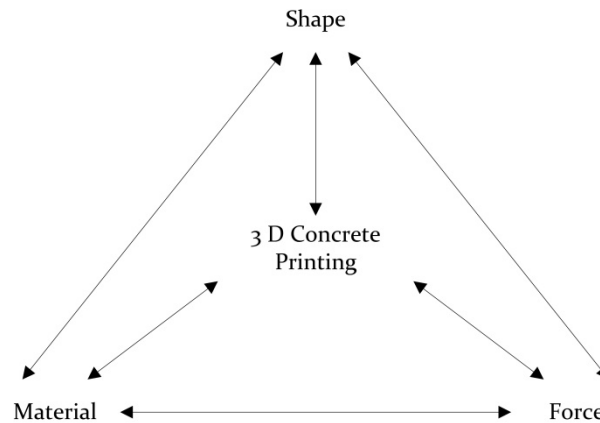


Figure 5.3 - 3D Concrete printing components

It is clear that printable concrete differs from its traditionally used equivalent. This does not only concern mechanical properties, like bond strength and overall strength development, but also the printing behaviour. The mixture has to be sufficient pump-able and extrude-able at first, but then has to be stiff enough to quickly carry its own weight and that of the following layers. During this graduation research these properties have been assumed to be fulfilled, but in reality this behaviour is just as unknown as the mechanical properties. Experimental research will have to be carried out to find the relation between mixture design, mechanical properties and printing behaviour of concrete.

This also raises questions about the mixture components. The state-of-the-art of concrete printing shows that most material is best described as ‘mortar’. Studies will have to show the range of aggregate size that is printable, and its influence on the mechanical properties. This way the mixture does not restrict itself to cement and smaller particles, but may also incorporate large size aggregates which can be harvested from recycled concrete. This is another step towards the call for a more sustainable built environment. Besides, the printed objects are either unreinforced, or reinforcement is applied manually. An interesting alternative may be found in the development of fibre reinforced mixtures, increasing the ductility of the printed concrete.

These new materials also call for new shapes to be printed. The printing technique offers a great freedom in design, which allows taking the typical properties of concrete (high compressive and low tensile strength) into account. The designs can then be taken a step further, by using printable concrete properties as a starting point for structural optimization techniques on element or cross-sectional level. These optimization methods have been developed in the past years, but they usually do not take the influence of the chosen construction method into account. If these techniques can be extended by the typical properties of 3D printing, it will result in highly efficient structures that can be created without the restrictions of traditional construction methods.

The structural optimization of printed structures can take place on multiple levels, as the printed element can have an efficient cross section (*Figure 5.4 left*) or a non-solid topology on element scale. Besides, when openings or out of plane curves are to be printed, a temporary material can be printed

along (Figure 5.4 middle). This (biodegradable) material may be removed and re-used afterwards, or incorporated into the design as insulation. The design and construction method truly becomes sustainable if this secondary material is structurally optimized as well (Figure 5.4 right).

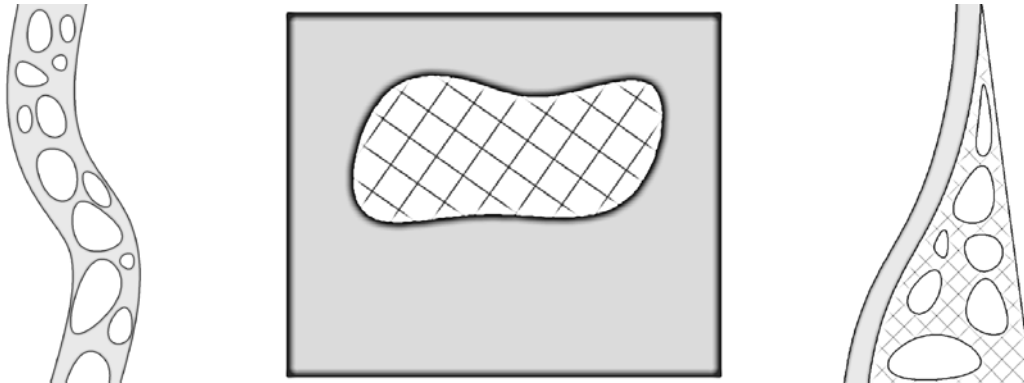


Figure 5.4 – Conceptual optimization of printed concrete structures and its support material

During this research the printed wall has been considered as ‘wished in place’ and then loaded by external forces. In other words, the wall has been analysed as if it was printed with a certain printing strategy, resulting in specific properties once completed. In practice however, the wall will be created layer by layer. To properly comprehend what happens during this printing process, and thus understand why the element has certain properties when completed, printed elements should also be analysed in this way. By modelling and analysing the process using a so called state parameter concept, the aging properties of each layer and the bond between layers can be determined. Loads can then gradually be applied and increased, corresponding with reality. This state parameter concept can be extended with the state of the environment, not only including a constant atmospheric temperature but also taking the internal heat production of concrete into account.

The printing strategy of the element considered has been quite simple, as it was printed using a constant speed, layer height and path. This simplified study has made clear that the chosen print strategy strongly influences the (mechanical) properties of the printed structure. Future research should therefore be aimed at more dynamic printing strategies. This may concern printing faster or slower where certain strengths are required, but can also mean a variable layer size to approach curved shapes or openings. It calls for an advanced printing head that allows flow control by varying its diameter, pressure and speed, to print inner structures or sharp curves. Likewise, the print strategy of an entire building plan is of interest. The printing order of load-bearing and non-structural elements in a building can be included in the strategy to improve the process of automation that 3D printing brings as a construction method.

These varying topics of research will help the development of the concrete printing technique, such that it can be taken into practice as a highly efficient design and construct method. This type of fundamental research can also be used to guide the new building legislation and standards that have to be developed. These laws should not slow down the development of the technique, but the safety of the future user may not be compromised: a suitable balance will have to be found when exploring the frontiers of 3D printing. The research model that is presented in this thesis has shown a way to support the technique. It is strongly recommended that future research will be carried out to extend and renew such models, to support the promising technique of 3D concrete printing, and apply it on a large scale in the building industry.

6. Bibliography

- 3Ders.org. (2014). *10 completely 3D printed houses appear in Shanghai, built under a day*. Retrieved June 2014, from 3Ders.org.
- 3Ders.org. (2014). *New photos of 10 'green' 3D-printed houses in Shanghai, built in 24 hours*. Retrieved from 3Ders.org.
- Alec. (2014, December). *Slovenian construction pioneers BetAbram share footage of their 3D house printers in action*. Retrieved February 2015, from 3ders.org.
- Allen, H. (1969). *Analysis and Design of Structural Sandwich Panels*. London: Pergamon.
- Anderson, S. (2014, November). *LU and Skanska Sign Collaborative Agreement to Commercialize 3D Concrete Printing Robot*. Retrieved February 2015, from 3dprint.com.
- Anderson, S. (2015). *Concrete Plans: CyBe's Berry Hendriks Describes Plans to 3D Print with Mortar*. Retrieved January 2015, from 3Dprint.com.
- Beaman, J., Barlow, J., Bourell, D., Crawford, R., Marcus, H., & McAlea, K. (1997). *Solid freeform fabrication: A new direction in manufacturing*. Dordrecht: Kluwer Academic Publishers.
- Beranek, W. (1970). Nieuwe berekeningsmethoden voor rechthoekige platen in de Betonvoorschriften 1970. *Cement*, 50-59.
- Carino, N., & Lew, H. (2001). The Maturity Method: From Theory to Application. *Proceedings of the 2001 Structures Congress & Exposition*.
- Cesaretti, G., Dini, E., De Kestelier, X., Colla, V., & Pambaguian, L. (2014). Building components for an outpost on the Lunar soil by means of a novel 3D printing technology. *Acta Astronautica*, 93, 430-450.
- Ciraud, P. A. (1972). *Patent No. 2263777*. France.
- Dini, E. (2013). Large Scale 3D Printing: from Deep Sea to the Moon'. *Low-cost 3D printing for science, education & sustainable development*, 127-132.
- Freiesleben Hansen, P., & Pedersen, J. (1977). Maturity Computer for Controlled Curing and Hardening of Concrete. *Nordisk Betong*, 19-34.
- Grimonprez, B., van Helvoort, R., Bastiaens, R., & Mistiaen, B. (2007). Bestaande technieken, nieuwe ontwikkelingen – Rapid Prototyping. *Kunststof en Rubber*(5).
- Hopkinson, N., & Dickens, P. (2003). Analysis of rapid manufacturing – using layer manufacturing processes for production, Proceedings of Institution of Mechanical Engineers Part C. *Journal of Mechanical Engineering Science*, 217(1), 31-40.
- Hwang, D., & Khoshnevis, B. (2005). An Innovative Construction Process-Contour Crafting (CC). *ISARC 2005 - September 11-14*.
- IAAC. (2014). *Minibuilders*. Retrieved June 2014, from iaac.net.

- Jacobson, L. (2013). *Simulated Annealing for beginners*. Retrieved January 2015, from theprojectspot.com.
- Khoshnevis, B. (2004). Houses of the Future – Construction by Contour Crafting Building Houses for Everyone. *University of Southern California Urban Initiative Public Policy Briefing*.
- Khoshnevis, B., & Zhang, J. (2012). Extraterrestrial Constructing Using Contour Crafting.
- Khoshnevis, B., Hwang, D., Yao, K., & Yeh, Z. (2006). Mega-scale fabrication by contour crafting. *Int. J. Industrial and Systems Engineering*, 1(3), 301-320.
- Krassenstein, E. (2014, August). *First Entirely 3D Printed Estate is Coming to NY, Including a 3D Printed 2400 Sqft House, Pool & More*. Retrieved February 2015, from 3Dprint.com.
- Krassenstein, E. (2014, September). *WASP Plans to Demonstrate New 6 Meter Tall 3D House Printer This Week: Will 3D print houses in developing countries next*. Retrieved February 2015, from 3dprint.com.
- Krassenstein, E. (2015, January). *3D Printed Full Size Replicas of Stonehenge, Atlantis, and More? One Engineer Has a Grand Plan*. Retrieved February 2015, from 3dprint.com.
- Le, T., Austin, S., Lim, S., Buswell, R., Law, R., Gibb, A., et al. (2011). Hardened properties of high-performance printing concrete. *Cement and Concrete Research*, 42, 558-566.
- Le, T., Austin, S., Lim, S., Buswell, R., Law, R., Gibb, A., et al. (2011). Mix design and fresh properties for high-performance printing concrete. *Materials and Structures*, 1221-1232.
- Leonhardt, & Walther. (1966). *Wandartige Träger*. Ernst.
- Lew, H., & Reichard, T. (1978). Mechanical Properties of Concrete at Early Ages. *ACI Journal*, 533-542.
- Lim, S., Buswell, R., Le, T., Austin, S., Gibb, A., & Thorpe, T. (2012). Developments in construction-scale additive manufacturing processes. *Automation in Construction*, 21, 262-268.
- Lim, S., Buswell, R., Le, T., Wackrow, R., Austin, S., Gibb, A., et al. (2011). Development of a viable concrete printing process. *28th International Symposium on Automation and Robotics in Construction (ISARC2011)*, 665-670.
- Löfgren, I. (2005). *Fibre-reinforced Concrete for Industrial Construction*. Göteborg: Chalmers University of Technology.
- Michalewicz, Z., & Fogel, D. (2000). *How to Solve It: Modern Heuristics*. Berlin: Springer.
- Oxman, N., Duro-Royo, J., & Keating, S. (2014). Towards Robotic Swarm Printing. *Architectural Design*, 108-115.
- Oxman, N., Keating, S., & Tsai, E. (2011). Functionally Graded Rapid Prototyping, Innovative Developments in Virtual and Physical Prototyping'. *Proceedings of the 5th International Conference on Advanced Research in Virtual and Rapid Prototyping*.
- Plowman, J. (1956). Maturity and Strength of Concrete. *Magazine of Concrete Research*, 13-22.
- Rael, R., & San Fratello, R. (2011). Developing Concrete Polymer Building Components for 3D Printing. *Integration through computation - Acadia 2011 Proceedings*.

- Ros, & Shima. (2013). Relationship between splitting tensile strength and compressive strength of concrete at early ages with different types of cement and curing temperature histories. *Proceedings of the Japan Concrete Institute*.
- Srinivast, S., & Rao, A. (1970). Bending, Vibration and Buckling of Simply Supported Thick Orthotropic Rectangular Plates and Laminates. *International Journal Solids Structures*, 1463-1481.
- Swainson, W. K. (1977). *Patent No. 4,041,476*. United States of America.
- Wolfs, R. (2014). *3D Printing of Concrete Structures Literature Review*. Eindhoven: Eindhoven University of Technology.
- Yingchuang. (2015). *Yingchuang.com*. Retrieved February 2015, from yhbm.com.
- Abaqus Analysis User's Manual 6.13*. (2013). Providence: Dassault Systèmes Simulia Corp.
- 3dprintcanalhouse.com*. (2014).
- buildfreeform.com*. (2014).
- contourcrafting.org*. (2014).
- dinitech.it*. (2014).
- emergingobjects.com*. (2014).
- totalkustom.com*. (2014).

Annex A. 3D Concrete Printing: State-of-the-Art

This annex gives an overview of the state-of-the-art of 3D concrete printing, containing experimental data and showcases of institutes spread around the world. The data in this annex is based on the literature review that has been carried out in the initial phase of this graduation project. However, as concrete printing undergoes daily developments, that review is partly out dated and therefor extended with all available on concrete printing up to the moment of writing. The annex starts off with a brief introduction on 3D printing and a general overview of (commercial) 3D printing methods. The second part contains an in-depth discussion on the 3D concrete printing techniques being applied worldwide.

A.1. Additive Manufacturing

Beamen et al. (1997) state that a distinction can be made between in three manufacturing techniques: subtractive fabrication, net-shape fabrication and additive fabrication. A good example of the first technique is CNC (Computer Numerical Control) Milling: removing material from an initial block or bar of raw material. Net-shape fabrication is based on reshaping raw materials (i.e. powder, melt, plaster). Examples of this technique are injection moulding or investment casting. Finally, additive fabrication is based on selectively adding material (like welding) to create more complex objects.

While all of these methods have been developed and automated in the past, the manufacturing costs and time are still mainly based on the geometric complexity and production quantity. A cost analysis is performed for a small (lever) and medium (cover) sized part (*Figure A.1*). The traditional manufacturing technique (injection moulding) is compared with layered additive processes and shows that the traditional process is relatively uneconomic, until high quantities are reached (Hopkinson & Dickens, 2003).

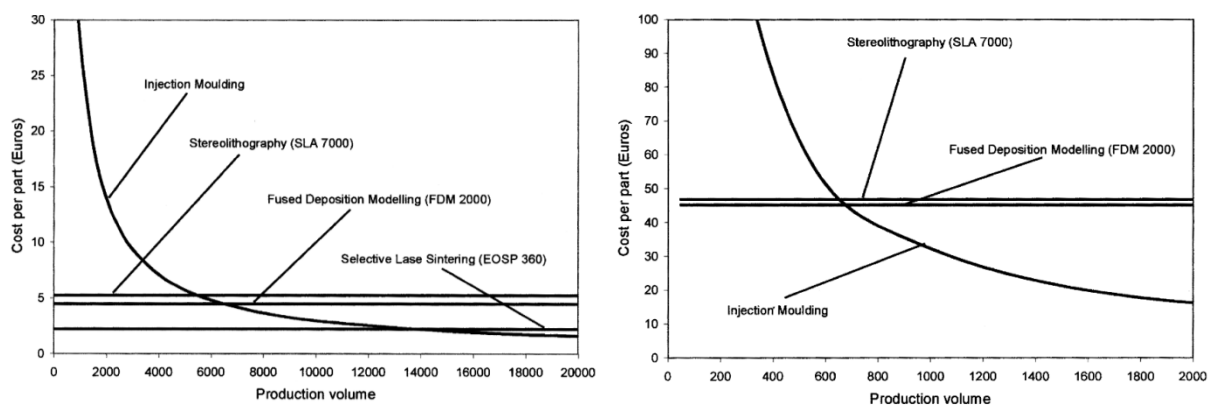


Figure A.1 - Cost comparison for different fabrication processes (Hopkinson & Dickens, 2003)

3D Printing, or additive manufacturing (AM), does not belong to the three fabrication processes discussed above. It is a fairly new technique with different execution approaches. All of them are based on a layer-wise method: a 3D computer model is sliced in horizontal layers, which are reproduced by a machine, layer by layer. This technique is not only suitable for a wide range of shapes, but also allows the production of low quantity parts to be economic, as seen in *Figure A.1*. Different additive techniques that have been developed throughout the years will be discussed below.

The roots of additive manufacturing may be found in the 19th century, where its application is seen in topography and photosculpture. Slicing simultaneously taken photos around an object allowed an artisan to carve a solid sculpture of the figure, as first carried out by the Frenchman François Willème in 1860. Blather used a layered method to create moulds for topographical relief maps in 1890, by cutting wax plates on contour lines and stacking them on top of each other (Beaman, et al., 1997).

While these techniques allowed for new ways to create solid shapes, most of this work was still done manually. It was not until the late 20th century that the first attempts were made to automate additive fabrication processes. Swainson presented a way by selective three-dimensional polymerization of a photosensitive polymer at the intersection of two laser beams in 1968 (Swainson, 1977).

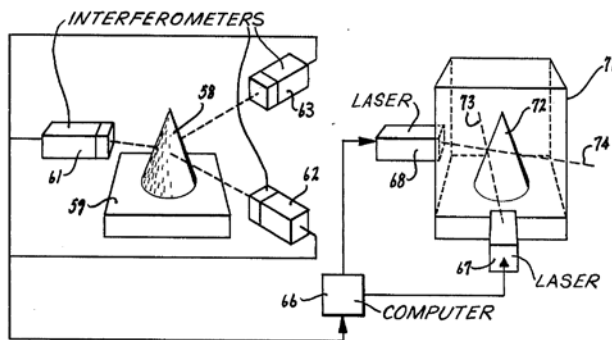


Figure A.2 – Selective laser technique (Swainson, 1977)

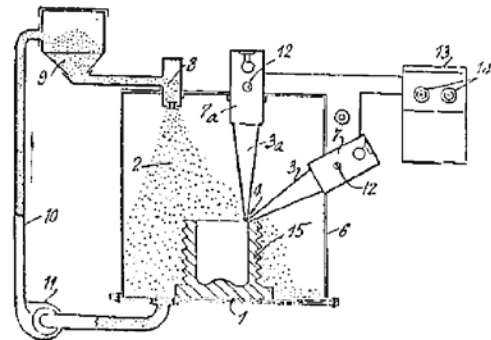


Fig A.3 – Powder based technique (Ciraud, 1972)

A powder based process followed in 1971 by Ciraud. Particles from varying materials with meltable properties are positioned and then locally heated by laser, electron beam or plasma beam (Ciraud, 1972).

Much development followed during the next years, both on academic and commercial field. This research led to a range of different additive techniques, which will be discussed and compared below. A distinction can be made between multiple commercial techniques, which are compared by Grimonprez et al. (2007). Their findings on the most common AM processes will be discussed; their images are used to illustrate the techniques in the following paragraphs.

A.1.1. SLA – Stereolithography Apparatus

Stereolithography uses an ultraviolet (UV) laser to solidify a thin layer of photocurable resin. This resin is positioned in a bath which can be moved in vertical direction. The laser initiates a curing reaction in the liquid polymer. When each spot of a layer has been solidified, the bath is lowered and a fresh layer of photocurable resin is positioned on top of the previous one. The laser is activated again and the new layer will bond to the preceding one. This process is repeated until the entire object is completed. This stepwise fabrication results in a visual layer wise result, which may be improved by sandblasting or scouring. The outcome is usually brittle and not very strong. To prevent overhanging parts of the object to float away during the process, temporary support structures may have to be printed and removed afterwards

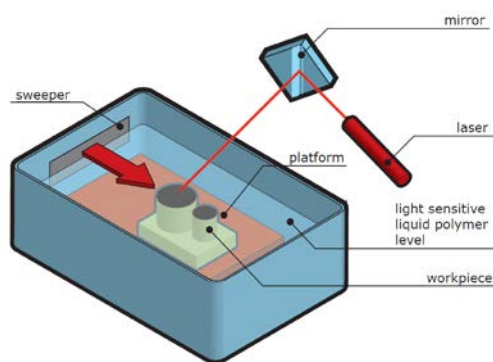


Figure A.4 - SLA – Stereolithography Apparatus

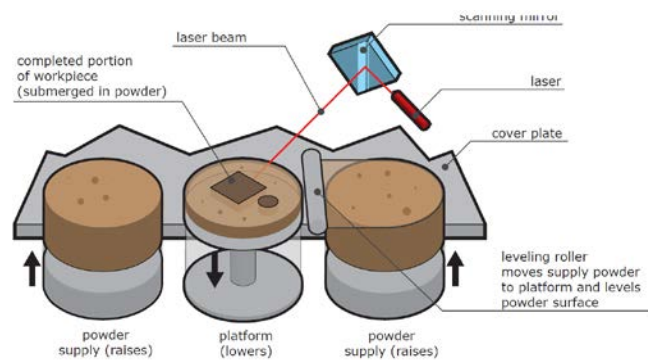


Figure A.5 - SLS - Selective Laser Sintering

A.1.2. SLS – Selective Laser Sintering

Selective laser sintering uses layers of polyamide or polycarbonate powder. Each layer is locally heated by a CO₂-laser, which results in local sintering or melting of the material and bonding to the layer below. After each layer, a roller applies a new sheet of powder on top of the old one. As the printed object is fully enclosed by powder during fabrication, no additional support structure is required. The result has a relatively rough finish, but is stronger compared to SLA.

A.1.3. LOM - Laminated Object Manufacturing

Laminated Object Manufacturing is a fabrication technique based on gluing layers of paper, PVC or even metal. For each layer a laser cuts out the 2D-contour of a sheet coated with melting glue. The material is then unrolled over the production table and glued to the previous layer by a roller. The material in the unwanted zones is removed by a laser or knife. When completed, the production table is lowered and the process is repeated for the next layer. The technique is less suitable for decent surface finish and is sensitive for damage in case of complicated geometries.

A.1.4. FDM - Fused Deposition Modelling

Fused deposition modelling is an extrusion based fabrication technique. A heated nozzle moves over a production table in X/Y direction, extruding a thin wire of thermoplastic polymer. The extrusion hardens immediately. After completion of one layer, the table lowers in z-direction and the next layer is applied. As there is no supporting material available compared to the other techniques, gravity may cause the object to fail during production. Printing an additional material as (temporary) support structure may solve this issue.

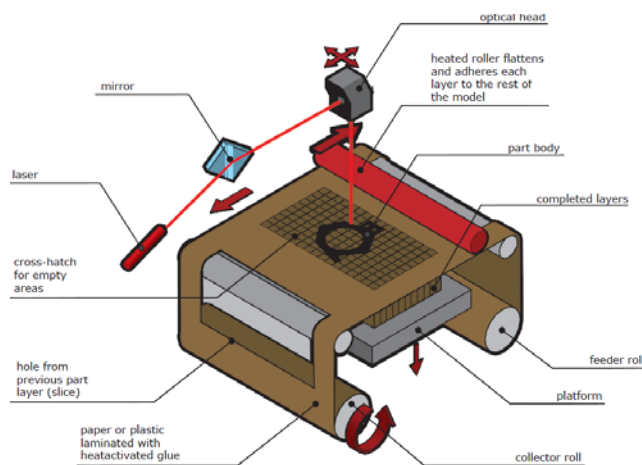


Figure A.6 - LOM - Laminated Object Manufacturing

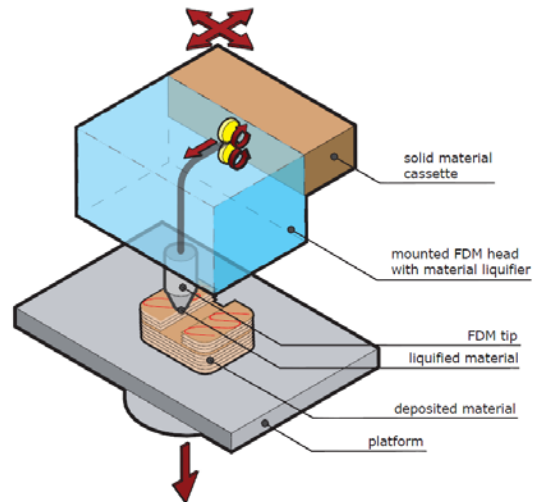


Figure A.7 - FDM - Fused Deposition Modelling

The Dutch company DUS Architects uses the FDM technique to 3D print a canal house in Amsterdam. They have up scaled a FDM printer to a so-called 'Kamermaker' and use it to print building bricks that can be linked into the final shape (Figure A.8.).



Figure A.8 - Kamermaker by DUS Architects (3dprintcanalhouse.com, 2014)

A.1.5. SGC – Solid Ground Curing

Solid Ground curing uses a photosensitive resin layer, comparable with SLA. SGC however hardens the layer in one go, by applying a mask for the non-hardening areas. A new mask is created for each layer, thereby selectively exposing the material to a powerful UV-lamp. The mask is made out of a transparent plate covered by toner, comparable to copying machines and laser printers. After exposing, the remaining liquid is sucked away and the layer is once again exposed to the UV-light for additional hardening. Finally, the layer is flattened to create a smooth surface for the next layer. A similar mask-based process is developed by Sintermask, which is called Selective Mask Sintering (SMS) (Grimponez, et al., 2007).

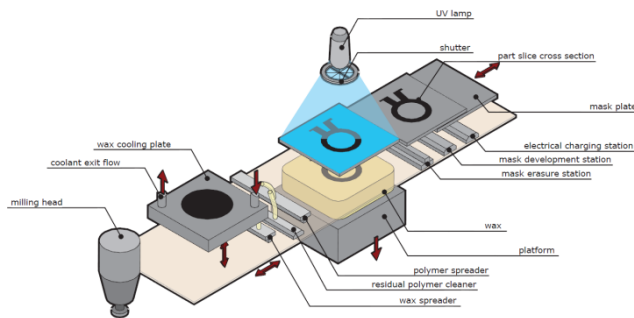


Figure A.9 – SGC - Solid Ground Curing

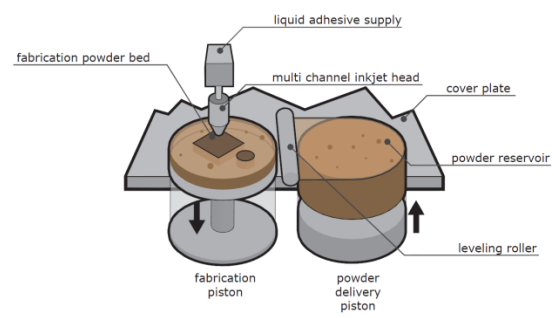


Figure A.10 – MJM – Multi-Jet Modelling

A.1.6. MJM – Multi-Jet Modelling

Multi-Jet Modeling uses a powder-binder combination. Each layer consists of freshly deposited powder, which is then selectively sprayed by binder by an inkjet mechanism. The result is fragile and may be strengthened by dipping in wax or other material. The benefit however, is the high resolution that can be achieved with the relatively small print head jets.

A.2. 3D Concrete Printing

The building industry has followed the developments on 3D printing techniques discussed in the previous part and started to apply them on a larger scale. Mainly the printing of concrete and cementitious materials has lately gained much interest in the field of architecture and construction. Unsurprisingly, varying techniques have been developed during the past years.

An overview of different construction-scale 3D printing techniques is given by Lim et al. (2012). At their time of writing Contour Crafting (University of Southern California), D-Shape (Enrico Dini) and Concrete Printing (Loughborough University) were the main companies in the concrete printing world. Today, however, new players have entered the field, combining the proven printing techniques with new features. This chapter aims to discuss these different concrete printing techniques and compares their main pros, cons and areas of application.



Figure A.11 – Contour Crafting (Top left), D-Shape (Top right) and Concrete Printing (Bottom)

(contourcrafting.org, 2014), (dinitech.it, 2014), (buildfreeform.com, 2014)

A.2.1. Contour Crafting

Contour Crafting (CC) is one of the oldest still existing concrete printing techniques. The first publications on the technique by Khoshnevis (University of Southern California) can be found in 1998 and much progress has been made since. Contour Crafting is a method of layered manufacturing, using polymer, ceramic slurry, cement, and a variety of other materials and mixes to build large scale objects with smooth surface finish. The smoothness of the extrusion is achieved by constraining the extruded flow in the vertical and horizontal direction to trowel surfaces. The orientation of the side trowel is dynamically controlled to conform to the slope of surface features. CC is also capable of using a variety of materials with large aggregates and additives like reinforcement fibres (Khoshnevis, Hwang, Yao, & Yeh, 2006). The nozzle may consist of multiple outlets, i.e. one for each side, and others for the inner (core) of a wall structure (*Figure A.12 right*). This way a co-extrusion of multiple materials is also possible. By deflecting the nozzle, non-orthogonal surfaces such as domes and vaults can be created (Khoshnevis, 2004).

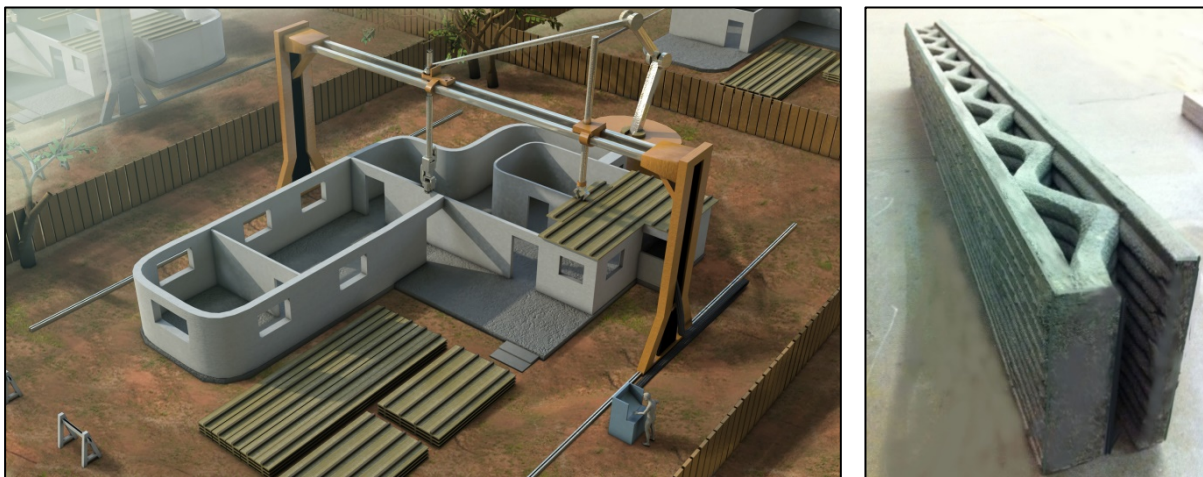


Figure A.12 – Contour Crafting by University of Southern California (contourcrafting.org, 2014)

The research done at the University of Southern California is carried out in three phases (Khoshnevis, 2004):

Phase I aims at developing the basic CC technology to print single-residence structures in one go. A gantry system carries the nozzle system and robotic arms move on two parallel lanes on the construction site (*Figure A.12 left*). Adding an additional support-beam picking and positioning arm could produce conventional structures (at openings). Besides, integrating automatic embedment of reinforcement is studied as well.

Phase II consists of expanding the system to a larger community and multi-residence structures. This includes an automated system for beam installation, plumbing, electrical and communication wiring, painting and tiling. This way apartment buildings, hospitals, schools and government buildings may be realised.

Finally, phase III will pursue adaptation of Contour Crafting as the construction technique of entire communities. Sensory systems and information technologies will be included for real-time inspection and feedback and project management for the new technology is created for effective implantation.

The research group in California has achieved a suitable concrete mixture based on trial and error experiments. It contains the following components:

- | | |
|-------------------------------------|-----|
| - Type II Hydraulic Portland Cement | 37% |
| - Sand | 41% |
| - Plasticizer | 3% |
| - Water | 19% |

Unsurprisingly, the mixture contains a plasticizer to increase the workability and has a small particle size to accommodate for the nozzle diameter. The water/cement ratio equals 0,5 and experiments shown a mean compressive strength equal to 18,9 N/mm². The mixture is designed to cure to maximum strength in a few hours (Hwang & Khoshnevis, 2005). In multiple publications and interviews it is mentioned that the contour crafting technique may use fibre reinforced concrete, but both the type and percentage is unknown. No other data is available on the concrete used by CC.

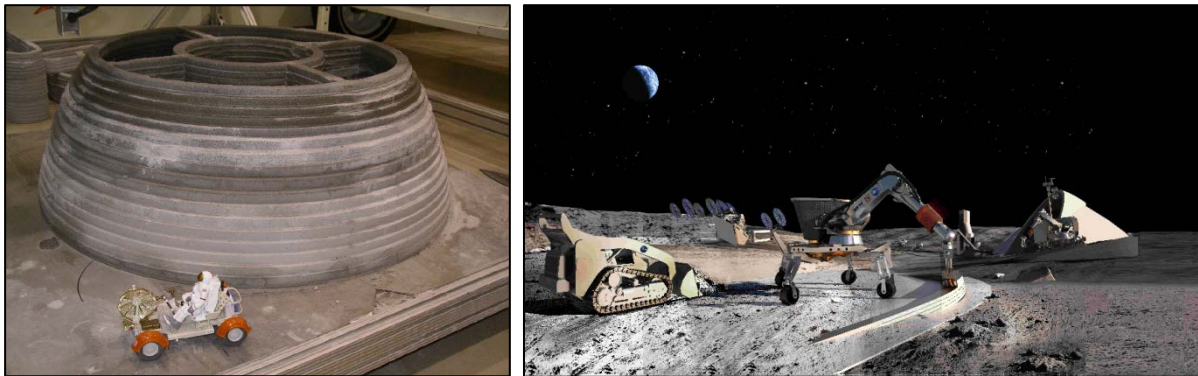


Figure A.13 – Lunar Contour Crafting experiments (left) and impression (right) (Khoshnevis & Zhang, 2012)

Contour Crafting has gained interest and funding of space administration NASA. Human beings are to return to Moon and land on Mars in the future, but most proposals for construction have been based on transporting structural elements from Earth and assembling them at their destination. This is an expensive and infeasible approach on a larger scale. Instead, the use of in-situ material for 3D printing is suggested. CC aims to use a combination of sulphur concrete and sintered moon soil, so called regolith. Both can easily be obtained in-situ, saving transportation costs. Promising experiments have been carried out, showing the benefits of Contour Crafting as a lunar construction method (Figure A.13.). The need for human intervention is minimized, thereby reducing food and oxygen supply, as well as required safety measures. (Khoshnevis & Zhang, 2012)

A.2.2. D-Shape

D-Shape is a 3D printing process, developed by Enrico Dini, which uses powder deposition selectively hardened by locally applying a binder, much in the same way as Multi-Jet Modelling (see chapter A.1.6.). Layers of sand are laid down to the desired thickness and compacted. A printing head composed of 300 nozzles, mounted on a gantry aluminium frame, is then moved over the printing area and deposits the binder where the part is to be solid. Once completed, the part is dug out of the loose powder bed (Lim et al., 2012).

D-Shape's technology is used to print buildings or building blocks. The first may be achieved by producing directly on site, while the second can be conceived either 'off site' or 'beside site'. This may result in a set of building blocks that can be assembled together. The to-be-printed granular material has to be preliminary mixed with pulverised metal oxide, which will later react with the applied liquid binder (Cesaretti, Dini, De Kestelier, Colla, & Pambaguian, 2014).



Figure A.14 – D-Shape by Enrico Dini (dinitech.it, 2014)

Similar to CC, D-Shape has found funding in a space administration to develop their technique: ESA. D-Shape is also interested in using regolith to harvest and print lunar outpost structures. Preliminary studies have been carried out on efficient (structural) design, material properties, vacuum tests and even on a robotic version of the D-Shape printer (Cesaretti, et al., 2014). The design of the outpost' structure has been carried out by London's Fosters+Partners, resulting in an igloo-like structure, with an external shield of printed regolith. This shield has a so called closed-foam structure: internal cavities giving suitable thermal insulation and micrometeoroid shielding properties, while minimizing material use (Figure A.15.). The bottom layer diameter is about 10m, with a total height of around 5m (Dini, 2013).

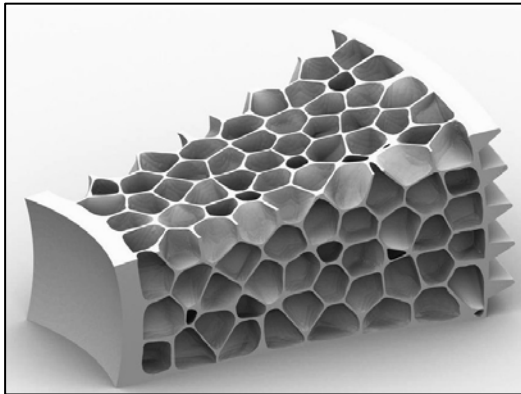


Figure A.15 – Lunar outpost wall structure (Cesaretti, et al., 2014)



Figure A.16 – 5 x 5 x 5 m D-Shape printer (Krassenstein, 2014)

Recently, D-Shape has announced collaboration with KUSHNER studios in 3D printing a large estate in New York, including a swimming pool, using an up scaled printer of 5 x 5 x 5 meters. It will be printed in container size units of 10 by 5 meters, using locally sourced building materials. Local rock can be harvested and crushed, combined with the bonding agent to create the printed material (Krassenstein, 2014).

A.2.3. Concrete Printing

Concrete Printing (CP) is an extrusion based process, similar to Contour Crafting. This technique however, has a smaller resolution of deposition to achieve higher 3-dimensional freedom, as it allows greater control of internal and external geometries. One of the by-products of this process is the ribbed surface finish, as the resulting surface is heavily dependent on the layer thickness (Lim, et al., 2012).

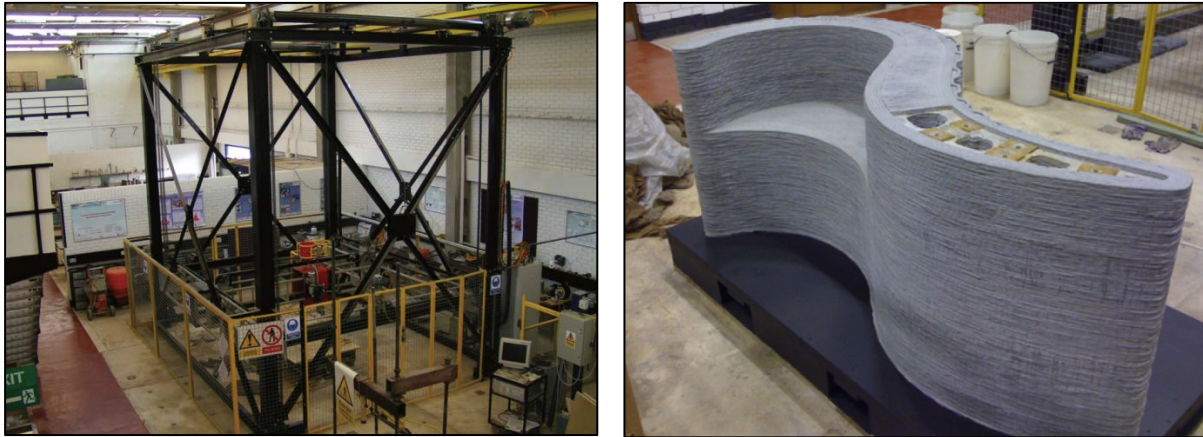


Figure A.17 – Concrete Printing by the University of Loughborough (Lim, et al., 2011)

The Department of Civil and Building Engineering at the Loughborough University has developed a high performance printing concrete for the Concrete Printing technique. Extensive material research and performance tests have been carried out, aimed at achieving high strength properties (100 MPa in compression and flexural strength of 12 MPa at 28 days), as the layered structure of 3D printed components is likely to be significantly weaker than conventional in situ and prefabricated concrete. Additionally, workability, extrudability and buildability requirements were taking into account during the design of the mixture. A fibre reinforced mixture has been designed, containing superplasticiser and retarder to increase workability and strength (Le, et al., 2011). Varying tests on printed concrete have been carried out and compared to standard mould cast concrete; reference is made to ‘Mix design and fresh properties for high-performance’ and ‘Hardened properties of high-performance printing concrete’ by Le et al (2011) for their results. These findings will be discussed briefly below.

Because of the high printing resolution a 2 mm maximum size sand was selected. Additionally, cement CEM I 52.5, fly ash and undensified silica fume formed the binder component. The mixtures enclosed 12/0.18 mm length/diameter polypropylene micro fibre reinforcement. Five mixtures with different proportions were designed and evaluated (Figure A.18). The optimum mix was considered to be the one with the lowest content of binder, containing the recommended dosage of fibres from the supplier (i.e. 1.2 kg/m³), while remaining sufficiently printable and gain the target strengths (Le, et al., 2011).

Material	Mix proportions (kg/m ³)				
	Mix 1	Mix 2	Mix 3	Mix 4	Mix 5
Sand	1612	1485	1362	1241	1123
Cement	376	446	513	579	643
Fly ash	107	127	147	165	184
Silica fume	54	64	73	83	92
Water	150	178	205	232	257

Figure A.18 – Mix proportions of trial mixes for Concrete Printing (Le, et al., 2011)

Mixture 4 was found to be the optimum, with a 3:2 sand-binder ratio and 1.2 kg/m^3 fibres. The binder consists of 70% cement, 20% fly ash and 10% silica fume. This mix had a water-binder ratio of 0.26 and contained a superplasticiser and retarder with dosages of 1 and 0.5%, by weight of binder, such that the open time extended up to 100 min.

With a fixed mixture composition the research team created a test setup with mould-cast controls and specimens extracted from printed elements (*Figure A.19*). These were used to determine the key properties of printed concrete, namely density, compressive strength, flexural strength, tensile bond strength and drying shrinkage. The tests clearly show the difference between conventionally cast and in-situ printed states, considering the anisotropy caused by the printing process (Le, et al., 2011).

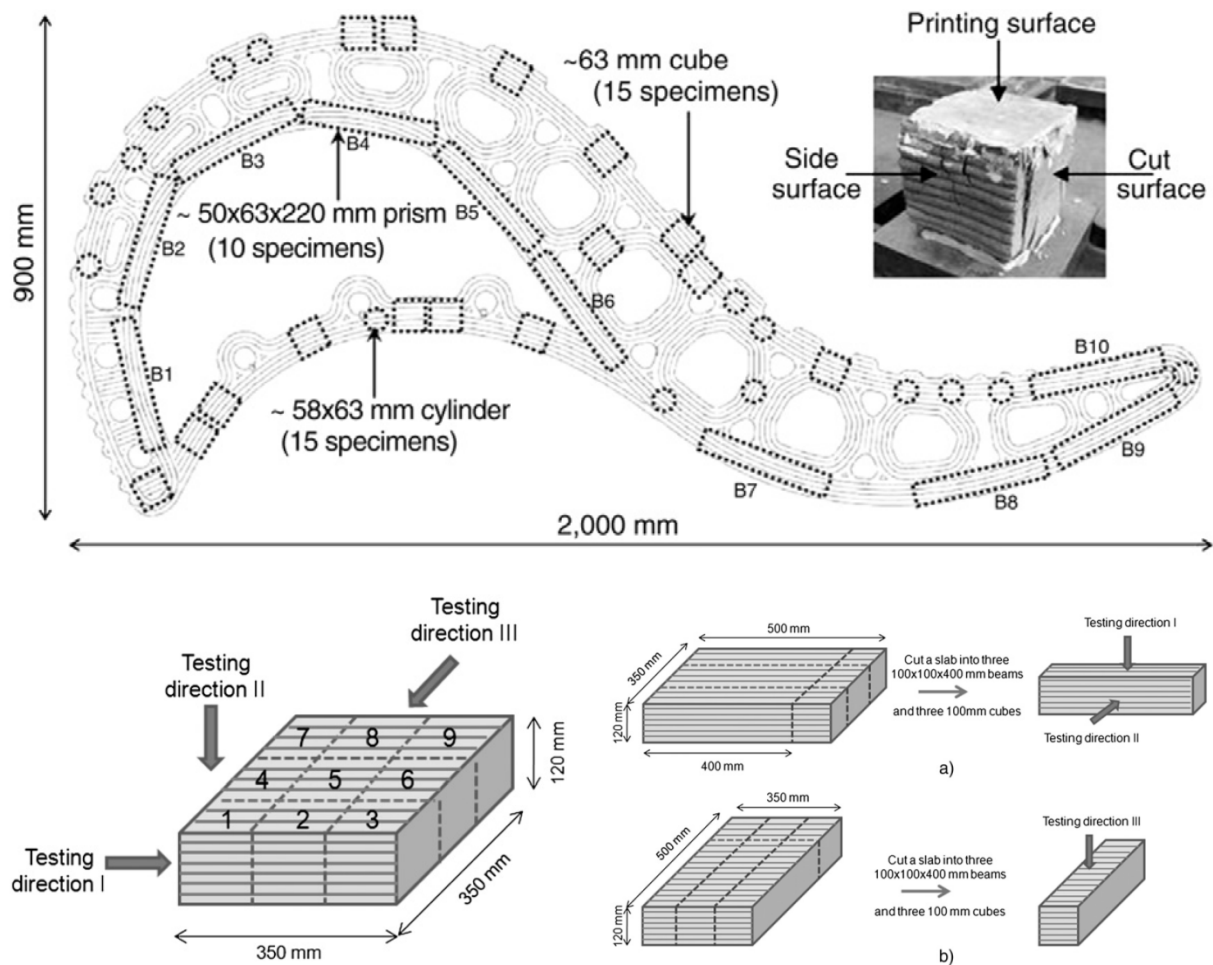


Figure A.19 – Positions of extracted specimens in printed element: curved wall (top) and slabs (bottom) (Le, et al., 2011)

The average density of the printed specimens was 2350 kg/m^3 , which is a little higher than the mould-cast: 2250 kg/m^3 . This may be due to the vibrating of the concrete printer, combined with the pressure that is provided by the pipe and pump system during extrusion.

The specimens extracted from the curved wall show a reduction of 30% in compressive strength compared to the mould-cast controls (107 MPa). Those extracted from the straight-line printed slabs showed only 5-15% loss in strength (91-102 MPa), depending on the orientation of the load relative to the layers. As expected, the lowest strengths occurred when the loading was parallel to the layers. Applying

a retarder significantly reduced the strength at early ages, i.e. 1 and 7 days, while accelerator improved the compressive strength considerably. Both effects disappeared at later ages.

The flexural strength of the printed aligned was significantly higher (13-16 MPa) than the mould-cast control (11 MPa), when tension was aligned with the layers. This may be due to the fact that the maximum tensile stress occurs at the bottom of the specimen, where the area is most likely well-compacted. As expected, the flexural strength was considerably reduced (by up to 36%) when loaded perpendicular to the layers: 7 MPa.

The tensile bond strength was studied for specimens with varying time gap between printed layers. All specimens with a time gap over 15 minutes failed at the interface, while the specimens of 0 and 15 minutes failed in the material. As expected, a clear reduction in bond strength is found when increasing the time gap (*Figure A.20*). The trend shows that characteristic bond strengths of 0.8, 1.0 and 1.2 MPa will be achieved at time gaps of 8, 3 and 1 h respectively.

As expected, the concrete cured in water shrank the least, followed by damp hessian and then a climatic chamber (20 °C and 50% relative humidity). The shrinkage in all three conditions was notably lower compared to sprayed mortars, which are considered to be low shrinkage materials. The optimum particle grading, low water/binder ratio and fly ash addition thus appeared to be beneficial in reducing shrinkage of the printed concrete (Le, et al., 2011).

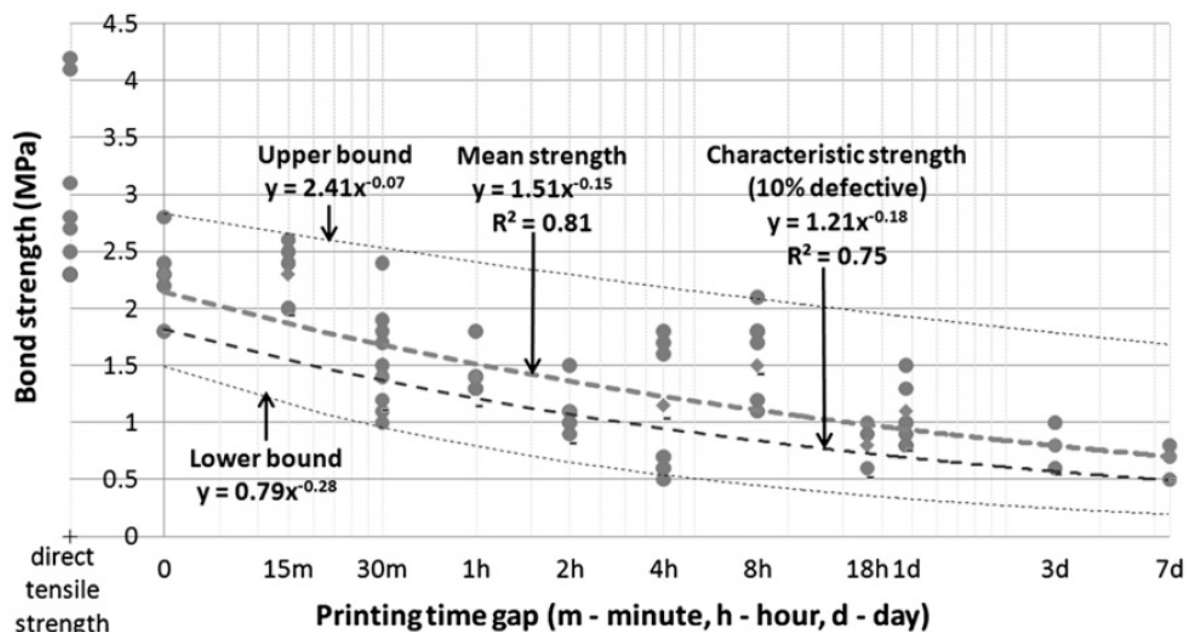


Figure A.20 – Tensile bond strength with printing time gap (Le, et al., 2011)

The Loughborough University has recently announced collaboration with construction company Skanska, to further develop their laboratory printer and commercialize it (Anderson, LU and Skanska Sign Collaborative Agreement to Commercialize 3D Concrete Printing Robot, 2014).

A.2.4. Yingchuang

The Chinese company Yingchuang uses a 150(L) x 10(W) x 6.6(H) m printer to print large scale building components at high speed in their factories. The method is similar to Contour Crafting: an inner and outer leaf is printed, followed by a zigzag shaped inner structure. Pictures show reinforcement being placed in between layers during printing. With this technique, the company has realised multiple houses, a five-story apartment block and a 1,100 square meter mansion. The mixture used contains glass fibre, steel, cement, hardening agents and recycled construction waste materials (Yingchuang, 2015).



Figure A.21 – Yingchuang printing process including reinforcement (Yingchuang, 2015)

The company has a different approach when it comes to logistics. Where CC aims to print an entire building at the construction site, Yingchuang prints large pieces of a building in a factory, transports them and finally assembles them at the construction site.



Figure A.22 – Top: Transport and assembly of printed components, Bottom: printed wall element (Yingchuang, 2015)

A.2.5. TotalKustom

While most 3D concrete printing progress has been made by companies, one man in Minnesota has been working on a printing technique in his garage in Minnesota. Contractor Andrey Rudenko has developed a printing technique, similar to Contour Crafting, but with a much smaller layer height (5mm). He believes that, in contradiction to Yingchuang, a cheap house built in a small time span is not the right target. Homes of good quality, which will take longer to build than cheaper buildings, will be more beneficial when including aspects like plumbing, insulation and electrical aspects in the construction process (www.3dprint.com, 2014).



Fig A.23 – 3D Concrete printing by TotalKustom (totalkustom.com, 2014)

Rudenko upscaled his printer size and founded TotalKustom, which has a wide range of targets. It aims to print zero-energy houses, replicas of ancient relics, a village of concrete houses and even develop a 3D printer for the moon (Krassenstein, 2015). Rudenko has shown the potential of his technique by 3D printing a scale model castle in his backyard. The ‘freeform’ tower components have been printed on the ground and the lifted on top of the walls of the castle, but Rudenko aims to print all future buildings in one go.



Fig A.24 – 3D Printed castle by TotalKustom (totalkustom.com, 2014)

A.2.6. CyBe Additive Industries

A third party that has adopted the CC technique is CyBe Additive Industries, founded in the Netherlands. This company uses a type of mortar that reaches a bearable strength within minutes. CyBe experiments by attaching a print head to a regular robot-arm, allowing for high diversity in print speed and strategy (Anderson, 2015).



Fig A.25 – Robot arm (left) and 3D printed wall (right) by CyBe Additive Industries (Anderson, 2015)

A.2.7. BetAbram

Slovenian company BetAbram is developing 3D concrete printers for commercial use. Varying sizes of printers will be up for sale and the company has showed their use by 3D printing a concrete staircase. Additionally they are researching a printable concrete, which may be purchased along with all required printer equipment (Alec, 2014).

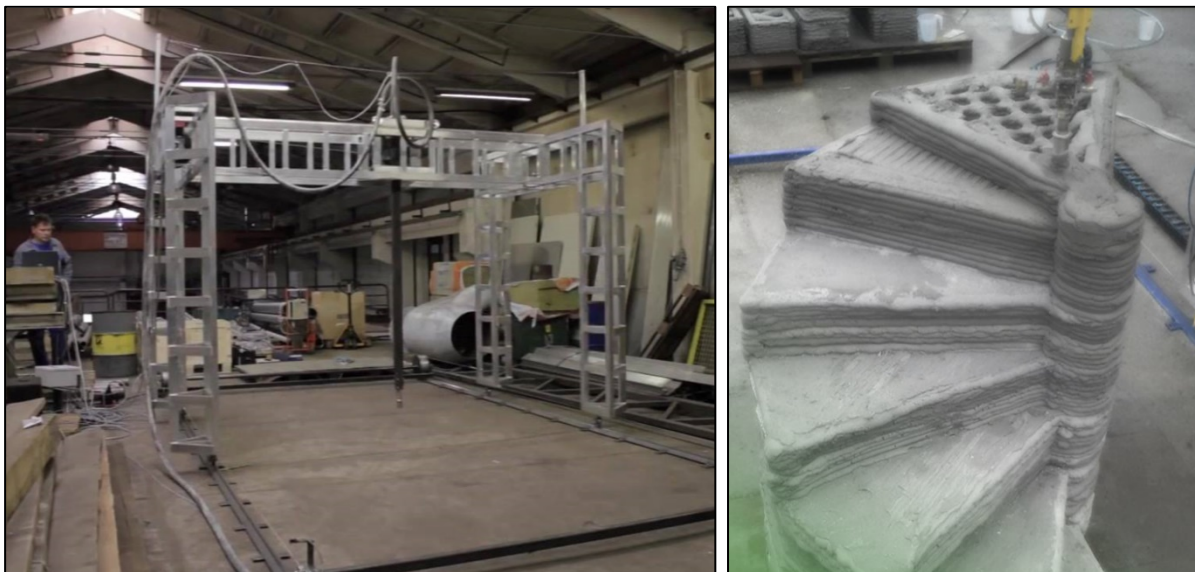


Fig A.26 – BetAbram's 3D printer(left) and a printed concrete staircase (right) (Alec, 2014)

A.2.8. Emerging Objects

A similar technique to D-Shape is applied by Emerging Objects (EO), a subsidiary of Rael San Fratello Architects. EO uses a fibre reinforced cement mixture with small-sized aggregates, applying adhesives to improve workability of the mix. This printing method uses two types of binder: an alcohol-based binder, which is a water-soluble synthetic polymer that has high adhesive and mixing properties and high tensile strength. It will help the mix cure more rapidly and cause it to be denser and have greater flexural strength. A secondary binder is added to further strengthen the material, by both hydrating the material and joining the fibres to the concrete mixture. The result is a hybrid concrete polymer (Rael & San Fratello, 2011).

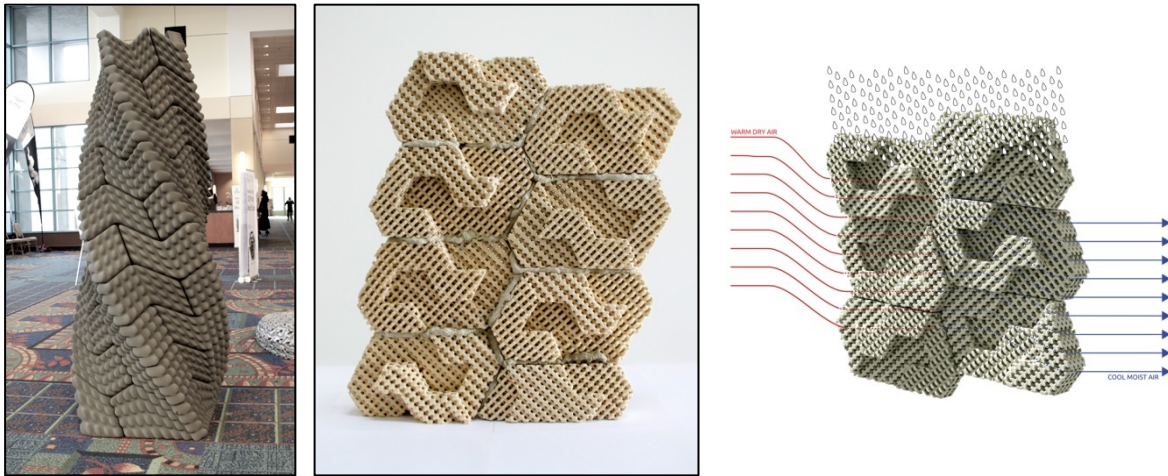


Fig A.27 – Quake Column (left) and Cool Brick (middle and right) by Emerging Objects (emergingobjects.com, 2014)

Emerging Objects has printed varying building components using their technique. Examples included the cement polymer Quake Column, which difuses the dynamic forces of an earthquake throught the interlocking components of the column, and the ceramic Cool Brick, which absorbs water and lets air pass trough to passively cool a room (Figure A.27). These shapes are possible because, similar to Dini's technique, the dry unused powder acts as a support structure during the printing process and can be recycled afterwards.

A.2.9. The Mediated Matter Group

The Mediated Matter group at MIT, led by Neri Oxman, is studying varying innovative printing techniques.

One of their topics is printing of concrete with varying density, so called 'Functionally Graded Rapid Prototyping'. Graded materials have a spatially varying composition or microstructure, which can be found throughout nature. The research group suggests a similar property for elements fabricated by rapid fabrication. A cement and concrete foam have been developed, where the density is controlled through an admixture of aluminium powder and lime, which react to produce hydrogen gas bubbles (Figure A.28 left). The experiments were carried out with varying radial density, showing promising mechanical results: a graded cylinder beam under bending stress can have 9% less mass than a solid cylindrical beam of the same dimensions and support the same load (Oxman, Keating, & Tsai, 2011).

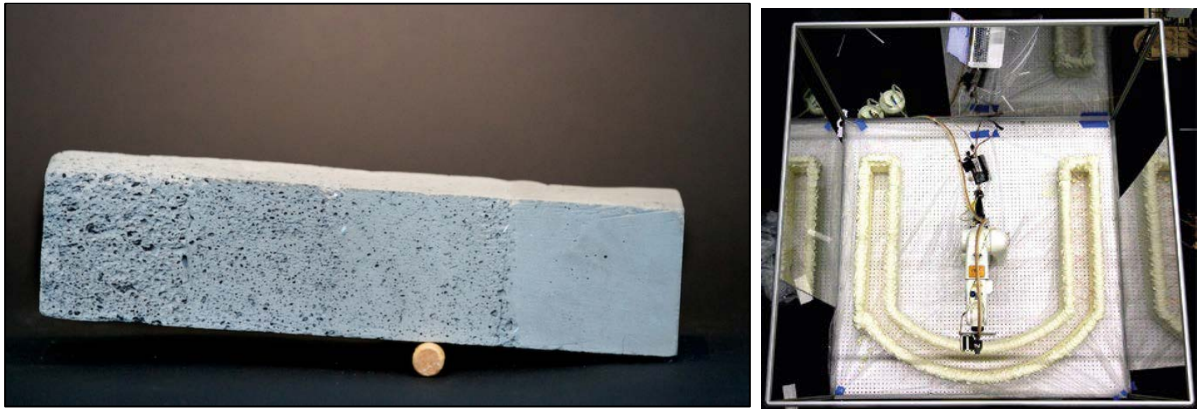


Fig A.28 – Functionally Graded Concrete (left) and Print-in-Place foam by MIT (Oxman, Duro-Royo, & Keating, 2014)

A second development at MIT is the so-called Print-in-Place technology, which prints a fast curing polyurethane foam that may be used as formwork for either temporal support or permanent insulation (Figure A.28 right). The Mediated Matter Group is also studying the application of swarm printing, which aims to overcome the limits of gantry systems by using a collective of cooperative small robots (Oxman, Duro-Royo, & Keating, 2014).

A.2.10. IAAC - Minibuilders

The Institute for Advanced Architecture of Catalonia (IAAC) has developed printing technique that falls in line with the swarm printing theory of MIT: minibuilders. This method uses a family of three small robots that are capable of printing (concrete) structures of any size together.

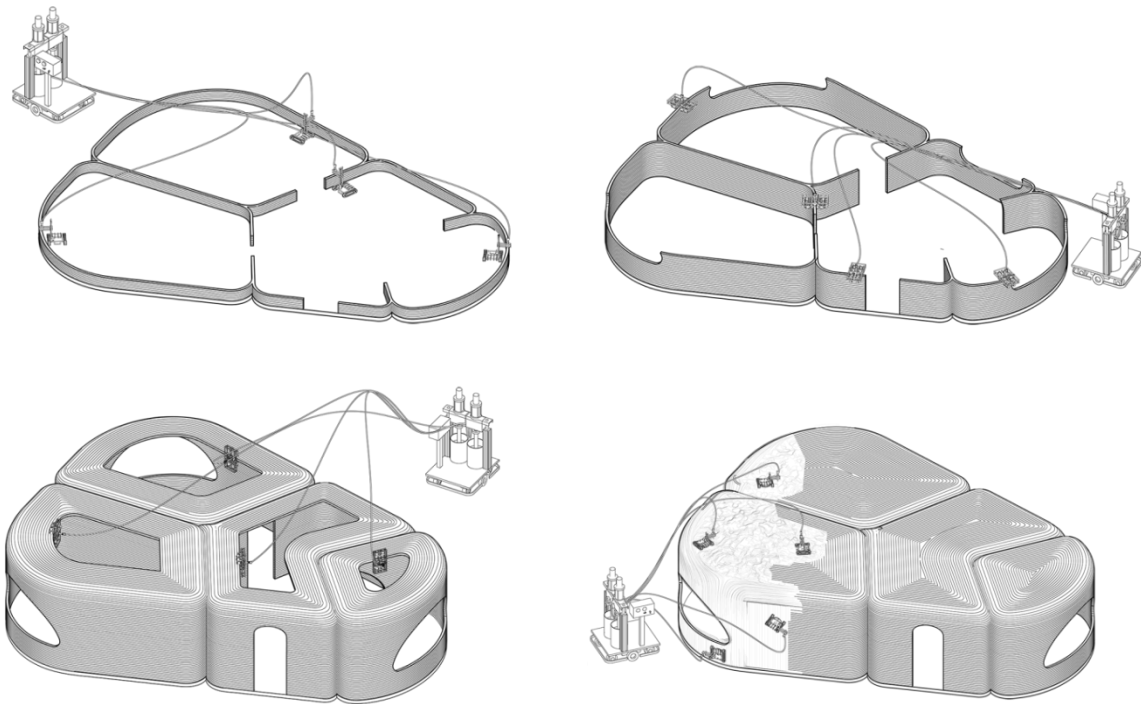


Fig A.29 – Minibuilders construction steps (IAAC, 2014)

One or more Foundation Robots create the footprint of the structure, followed by Grip Robots that are clamped onto the footprint and extend the structure, finding support on the previously printed layers. They climb up and use the fast curing speed to allow for horizontal printing. This way, ceilings and windows/door lintels may be printed. Finally, a Vacuum Robot moves over the printed structure and reinforces it by applying additional layers. These layers do not have to be parallel, but can be free form and only applied where the structure requires them (IAAC, 2014). While this technique is still in development, the principle is promising as it may counter one of the main points of attention of traditional additive manufacturing: the anisotropic behaviour that is caused by the layered build-up of the printed elements.

A.2.11. WASP

The final printing method of this annex is not officially a type of concrete printing, as it uses a mixture of clay, sand and mud. WASP (World's Advanced Saving Project) aims to 3D print shelter in third world countries. They use a 6 meter tall printer that can use local soil to print large structures. The Italian company has showed their concept by printing varying scale models of houses (*Figure A.30*), and now hope to sell their printers for commercial use. The acquired funds can then be used to apply the technique to print homes in countries in need (Krassenstein, 2014).



Fig A.30 – WASP 3D printing clay structures (Krassenstein, 2014)

Annex B. FEM Analyses

This annex contains benchmarks on the varying types of loading and material models that have been used to structurally analyse printed concrete structures. It starts off with in-and out of plane loading, following by a discussion on orthotropic behaviour caused by the layered production method. The final two paragraphs include new materials (NLE models) and new shapes (sandwich cross sections).

Abaqus provides conventional and continuum shell elements to model the structural wall (*Figure B.1*). The first type discretizes a body by defining the thickness as a cross section property of the shell. The latter uses a three-dimensional body and determines its thickness by the element nodal geometry. As this research assumes a structural wall with a constant thickness, conventional shells provide sufficient options to study the wall and are thus chosen for this purpose.

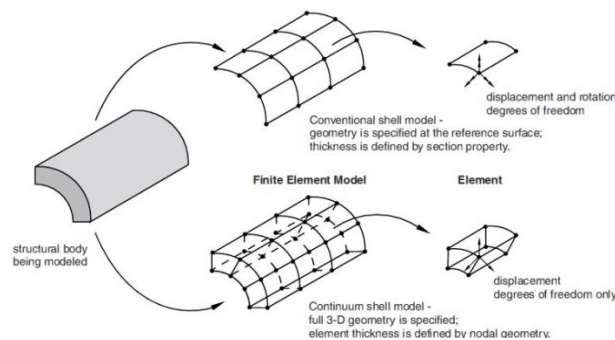


Figure B.1 – Abaqus conventional versus continuum shell element (Abaqus user manual 6.13, 2013)

Abaqus has a shell element library with elements for varying purposes, of which S8R (8 nodal, stress/displacement shell with reduced integration) is chosen for this study. This thick, conventional shell element has been tested and was able to fulfil all required material properties and/or loading conditions discussed in the follow paragraphs. For a homogeneous section the total number of section points is defined by the number of integration points through the thickness, using the default of five for Simpson's rule of integration. In case of non-linear elastic material behaviour (*paragraph B.4.*) the number of points is increased to 9.

B.1. Loading In Plane

The wall has to be loaded in plane, which is achieved by applying a (unit) line load on the top edge of the shell and supporting it by a roller and hinge at the bottom. These supports are spread over a length of $l/10$ on an element with negligible stiffness, to reduce peak stresses near the supports. The shell is then meshed with evenly distributed quadratic elements (

Figure B.2). The behaviour of the wall is benchmarked to analytic data of Leonhardt and Walther (1966), for a similar structure with varying *length / height* ratios.

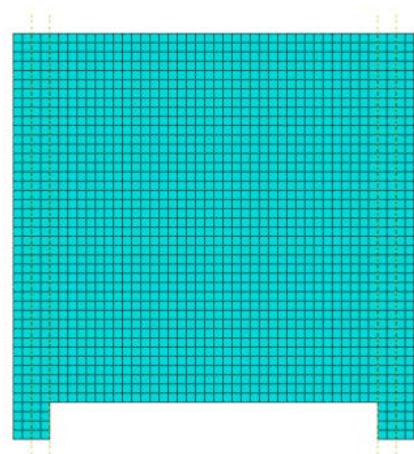


Figure B.2 – Abaqus representation of the printed wall

The occurring stresses in horizontal direction σ_x are plotted over the height of the wall, at midspan in Figure B.3. The shape of the stress pattern found in Abaqus agrees with the analytical solution, for varying length/height ratios:

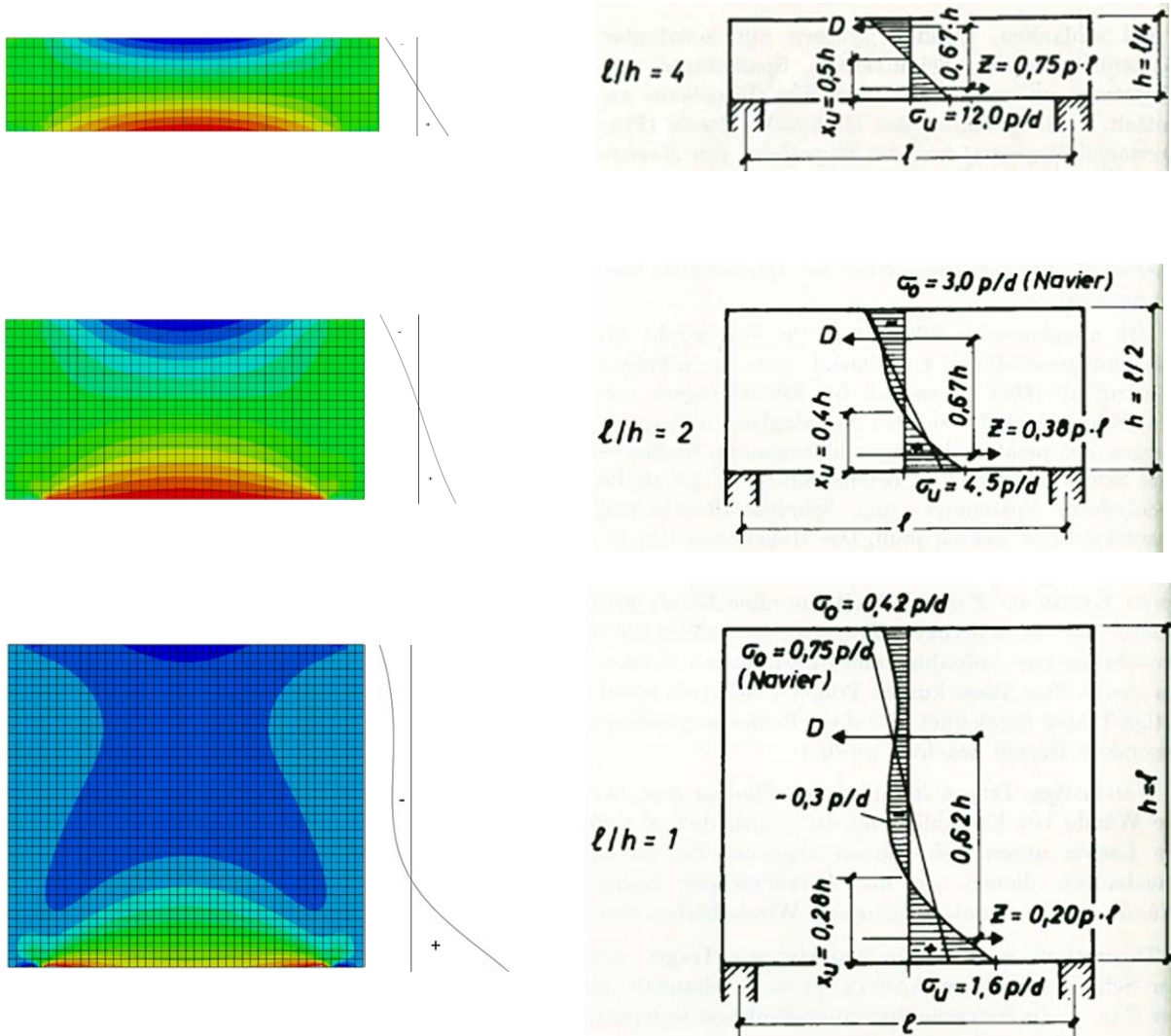


Figure B.3 – Horizontal stresses σ_x plotted at midspan (left) and compared to Leonhardt and Walther (right) for varying l/h ratios

A more detailed comparison between the analytical solution by Leonhardt and Walther and the Abaqus model is given below, for $length/height = 1,0$.

Figure B.4 shows the horizontal stresses going from compression to tension on the x-axis, over the height of the wall on the y-axis. The red line gives the results at midspan by Abaqus; the solid black line is the analytical solution. Additionally, the stresses in vertical direction σ_y are plotted in the same manner in Figure B.5. Here the red line contains stresses in Abaqus at midspan, while the blue line are the stresses closer to the support, at $0,15 \ell$. Again, the solid black line is the analytical solution.

The resulting stresses at midspan given by Abaqus correspond well to the analytical solution. The deviations of the results in Abaqus compared to the analytical results occur mainly at the introduction of loading (top edge) and close to the supports. This can be explained by the discretization of the structural system in Abaqus, which leads to small errors.

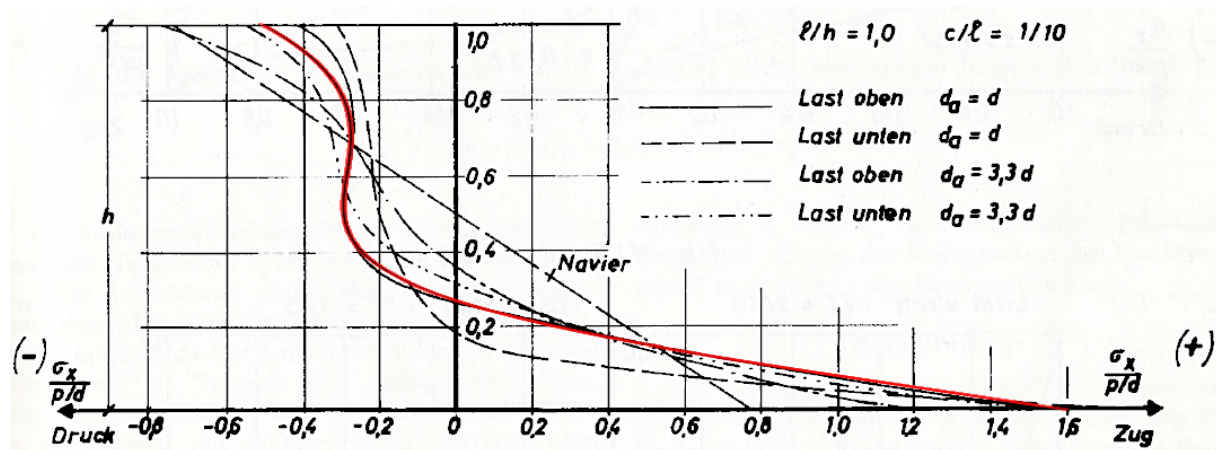


Figure B.4 – Horizontal stresses σ_x plotted at midspan (red) compared to Leonhardt and Walther for $l/h = 1,0$ and loaded at the top edge

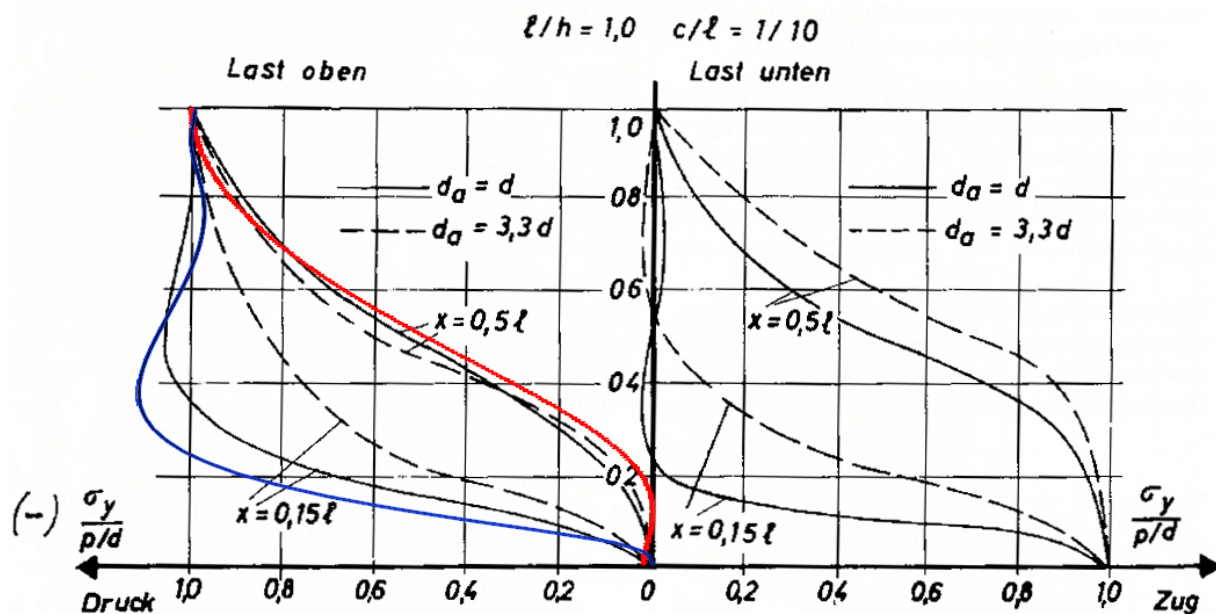


Figure B.5 – Vertical stresses σ_y plotted at midspan (red) and close to the support (blue) compared to Leonhardt and Walther for $l/h = 1,0$ and loaded at the top edge

B.2. Loading Out of Plane

Loading the wall out of plane is initially achieved by either applying a pressure to one face of the shell or a bending moment to its top edge. Once this behaviour is proven to be accurate, the analyses used in the thesis may consist of a combination of a bending moment and surface pressure.

Figure B.6 shows how the bending moment patterns in consecutively x-, y- and xy-direction match between theory (top) and Abaqus (bottom), for a simply supported plate (left) and all clamped plate (right).

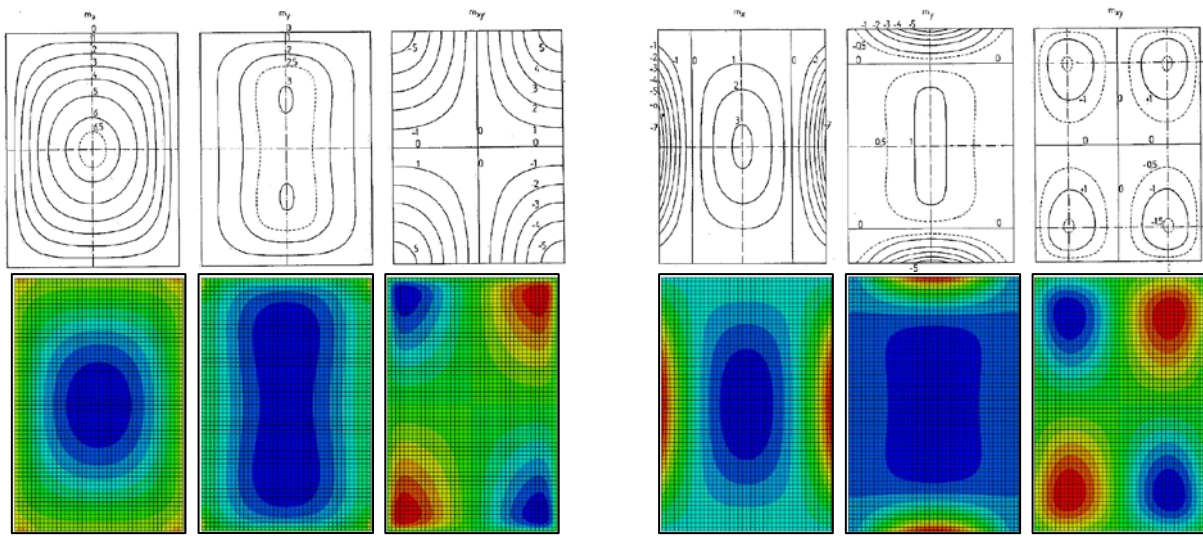


Figure B.6 – Bending moment patterns in a simply supported (left) and clamped (right) slab due to loading out of plane in Abaqus (bottom) compared to theory (top) (Beranek, 1970)

The corresponding bending moment values for both types of plates are compared in the table below, which show a good correlation between Abaqus and Kirchhoff-Love theory of pure bending.

		Normalized bending moment M_x at midspan			Normalized bending moment M_y at midspan			Normalized bending moment M_x at support			Normalized bending moment M_y at support		
	b/a	Exact	Abaqus	%	Exact	Abaqus	%	Exact	Abaqus	%	Exact	Abaqus	%
Simple Support	1	41	40,4	1,5	41	40,4	1,5	-	-		-	-	
	1,4	72	71,7	0,4	31,5	32,8	4,1	-	-		-	-	
	2,0	100	100,1	0,1	26	28,2	8,5	-	-		-	-	
Fixed Support	1	17,8	17,8	0	17,8	17,8	0	51	49,9	2,1	51	49,9	2,1
	1,4	32	31,4	1,9	12	12,3	2,5	72	71,7	0,4	55	55,2	0,4
	2,0	41	40,0	2,4	10	9,3	7	82	82,6	0,7	53	55,3	4,3

B.3. Orthotropic Behaviour

As 3D printing is an additive production technique, the mechanical properties between layers may be different from the properties in a single layer. In other words, as the studied object is an orthogonal wall, the properties in x direction will be different from those in y direction.

Abaqus allows orthotropy to be defined for shell elements by giving the elastic and shear moduli in the two principle directions: $E_1, E_2, G_{12}, G_{13}, G_{23}$. Additionally, the poisons ratio ν_{12} must be entered which is used to calculate ν_{21} , as the following relation holds: $\nu_{21} = (E_2/E_1) \cdot \nu_{12}$.

Orthotropic behaviour is often benchmarked using so called Argonite crystals, which have the following typical properties:

$$\begin{aligned} E_y/E_x &= 0,543103 & E_z/E_x &= 0,530172 \\ E_{xy}/E_x &= 0,23319 & E_{xz}/E_x &= 0,010776 \\ E_{yz}/E_x &= 0,098276 & G_{xy}/E_x &= 0,262931 \\ G_{xz}/E_x &= 0,159914 & G_{yz}/E_x &= 0,26681 \end{aligned}$$

Simply supported plates with multiple width/length/height ($b/a/h$) ratios are studied in Abaqus using Argonite crystals properties and the occurring deformations and stresses due to out of plane loading are compared to theory. The results are summarized below:

		Normalized displacement ($Q_{11} \cdot u$) / ($h \cdot q$)			Normalized stress x σ_x/q			Normalized stress y σ_y/q		
b / a	h / a	Exact	Abaqus	Error %	Exact	Abaqus	Error %	Exact	Abaqus	Error %
2	0,05	21542,0	21893,8	1,63	262,67	266,0	1,27	79,545	82,37	3,55
	0,10	1408,5	1452,94	3,16	65,975	67,25	1,93	20,204	21,13	4,58
	0,14	382,23	403,63	5,60	33,862	34,57	2,09	10,312	11,05	7,16
1	0,05	10,443	10930,7	4,67	144,31	149,2	3,39	87,080	89,57	2,86
	0,10	688,57	750,354	8,97	36,021	38,31	6,35	22,210	23,41	5,40
	0,14	191,07	213,327	11,65	18,346	19,79	7,87	11,615	12,40	6,76
0,5	0,05	2048,7	2156,92	5,28	40,657	42,05	3,43	54,279	56,51	4,11
	0,10	139,08	152,807	9,87	10,025	10,67	6,43	13,888	14,88	7,14
	0,14	39,790	44,4707	11,76	5,0364	5,438	7,97	7,2794	7,873	8,15

q : normal stress on top surface ($z = 0$); u : deflection of central point ($X = Y = Z = 0,5$); σ_x and σ_y normal stresses at centre top surface ($X = Y = 0,5$; $Z = 0$); $Q_{11} = E_x / (1 - \nu_{12} \cdot \nu_{21})$

The results in Abaqus correspond well to the exact solution given by Srinivast and Rao (1970), but the error increases as the plate thickness grows. Besides, only the orthotropic stiffness properties are given using this material model. Because of the bond between layers, it is desired to also have different strengths (i.e. failure stresses) in two directions.

The limits of Abaqus restrict the implementation of the required orthotropic material behaviour caused by printing, without the need to write a user defined material model. A different approach is therefore implemented in the Python script that guides the Abaqus analyses and post processing. Once an analysis has been carried out, a small loop algorithm checks the occurring stresses in each direction for each element of the structure. It writes the occurring maximum stresses for both compression and tension in two directions and evaluates if these values exceed the predefined orthotropic strength properties. In case these stresses do exceed the limit, the user is notified of the occurring failure and its location.

B.4. Non-Linear Elastic Materials

The use of non-linear elastic material behaviour is achieved by applying a Smeared Cracking model in Abaqus. This allows the incorporation of fibre reinforcement, by giving the tension stiffening properties of the concrete. The behaviour in compression is modelled as elastic-plastic (*Figure B.7 left*). The relationship between compression and tension is given by the failure ratios (*Figure B.7 right*).

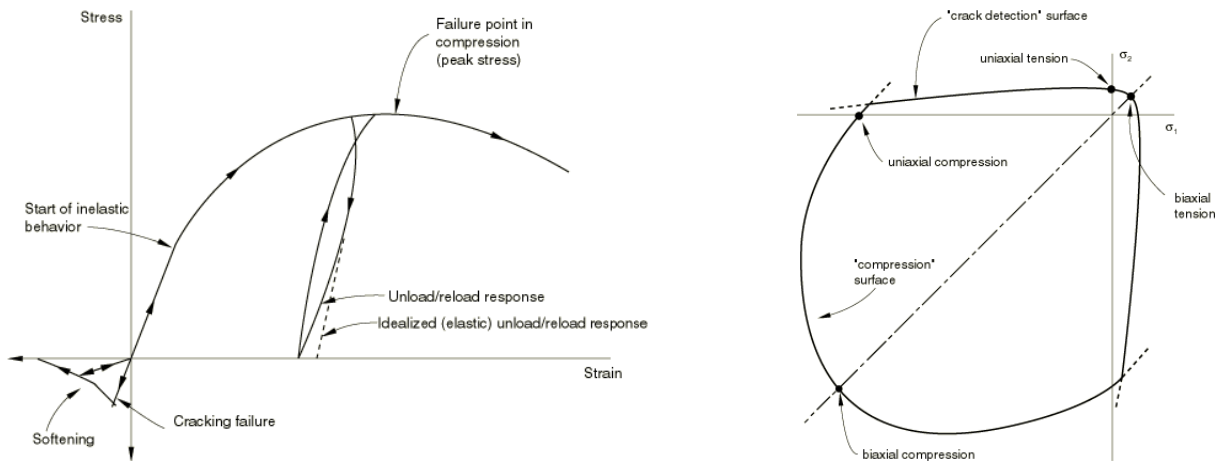


Figure B.7 – Stress strain diagram of concrete (left) and failure ratios (right) in Abaqus (Abaqus user manual 6.13, 2013)

The use of NLE material model in Abaqus is demonstrated below. Study has shown that tension stiffening is only beneficial if redistribution of forces is possible, in other words: when the structure is statically undetermined.

This is recognized from Figure B.9. The top shows a simply supported beam with a line load on top. Once the maximum load is reached, 'cracking' initiates. However, as there is no way to redistribute the forces, the beam fails shortly after.



Figure B.8 –Statically determined beam before and after 'cracking'

Figure B.9 shows a beam on 3 supports (half, because of symmetry). Here, redistribution of forces is clearly recognized from before cracking (left) and after (right). The highest bending moment initially is located above the middle support. Once the cracking load has been reached and the allowable stress above this support starts decreasing, the bending moment at midspan increases. Once the maximum stresses have been reached here as well, the beam fails.



Figure B.9 –Statically undetermined beam before (left) and after (red) 'cracking'

Both beams have been analysed with tension stiffening behaviour as often seen with lower fibre contents, i.e. softening.

However, if this content would be strongly increased and stiffening can be achieved, both the statically determined and undetermined system has a higher loading capacity. As the compressive stresses are still in the linear elastic branch, the cross section under tension gradually grows, until its 'fully plastic' in tension. Once the maximum allowable strain has been reached, the analysis terminates.

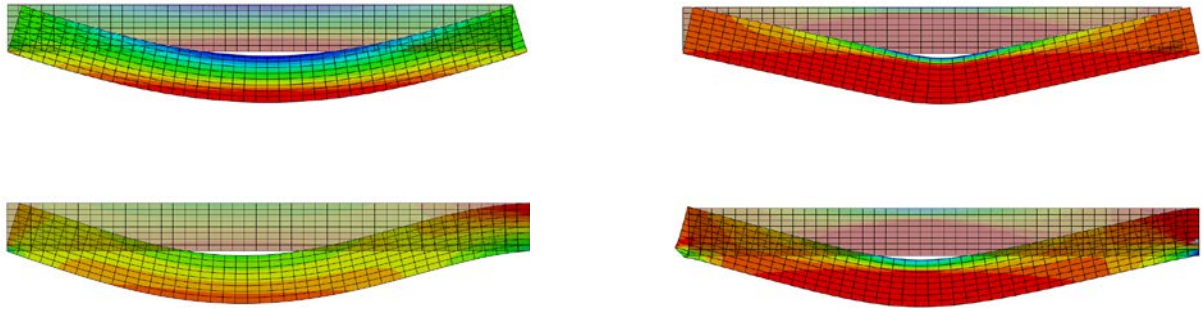


Figure B.10 –Statically determined (top) and undetermined (bottom) beam and before and after 'cracking' with high fibre contents

When considering the structural wall that is analysed in this thesis, a comparison can be made with the statically determined beam. As there is no redistribution of forces possible in the wall, it fails shortly after the cracking failure load is reached. Only if high fibre contents are applied, and a semi-plastic behaviour is achieved in tension, the load bearing capacity is increased. This goes hand in hand with large occurring deformations, which have to be kept in mind if such a system is applied in practice. It would be more beneficial to apply lower fibre contents in statically undetermined systems, such that redistribution of forces is possible.

B.5. Sandwich build-up cross sections

The Contour Crafting print technique and its equals are often linked to a specific type of wall: one with an inner- and outer leaf, along with a sinus-shaped inner structure. While this may be beneficial for structural reasons, i.e. de lateral bending stiffness, study has shown that it is not the most efficient cross section concerning thermal stresses (Wolfs, 2014). This might explain why often reinforcement is applied in between the layers of printed concrete.

A solution may be found in 3D printing of sandwich build-up cross section. These typical cross sections do have the advantage of an efficient mass-stiffness ratio, but also prevent the thermal bridging between inner- and outer leaf.

The ability to analyse sandwich structures is therefore implemented in the research model. This is done by applying a composite shell section to the wall in Abaqus, consisting of a thin concrete inner and outer leaf (faces), along with a solid core material in between. The user can define the material orientation for each layer, but for this study the core and faces are assumed to be isotropic. The behaviour in Abaqus is benchmarked to the theory of sandwich beams by Allen (1969).

A beam consisting of shell elements is loaded with a uniform pressure load and simply supported on its sides. A sandwich cross section is applied to the beam with a constant core of 100 mm, along with faces increasing from 5 to 100 mm. The table below shows how the displacement and stresses found by Abaqus correspond to the theoretical solutions by Allen.

Thickness faces	Displacement at midspan			Bending Stress in face edge			Bending Stress in core edge		
	Exact	Abaqus	Error %	Exact	Abaqus	Error %	Exact	Abaqus	Error %
5	9844	9897	0,54	6235	6241	0,10	0,18	0,18	0,00
10	4485	4500	0,33	3099	3096	0,10	0,086	0,086	0,00
20	1885	1879	0,32	1519	1508	0,72	0,036	0,036	0,00
50	483	468	3,11	555,5	537	3,33	0,009	0,009	0,00
100	136	127	6,62	234,4	217	7,42	0,0026	0,0024	7,69

The stresses in the faces and core are found using the ordinary bending theory for beams, adapted to a composite cross section. The resulting composite bending stiffness is equal to:

$$D = E_f \cdot \frac{bt^3}{6} + E_f \cdot \frac{btd^2}{2} + E_c \cdot \frac{bc^3}{12},$$

Here E_f and E_c are the bending stiffness of respectively the sandwich faces and core, and the other variables are geometrical properties as recognized from Figure B.11.

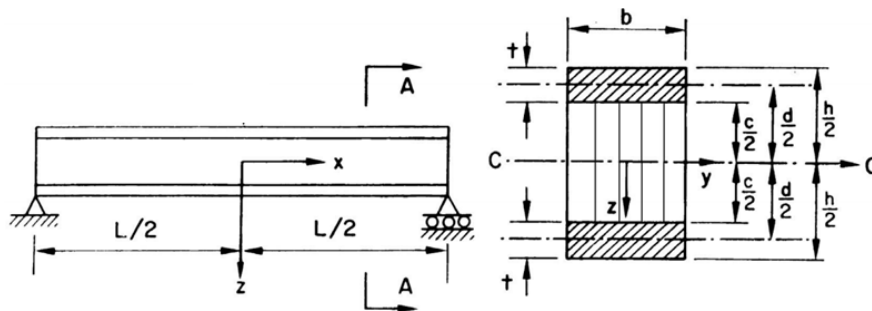


Figure B.11 – Dimension of sandwich beam (left) and section (right) (Allen, 1969)

In case of thin faces (i.e. when $3(d/t)^2 > 100$ is true), the first and third term of the composite bending stiffness equation can be neglected and the displacement and bending stresses are computed as follows (Allen, 1969):

$$D = E_f \cdot \frac{btd^2}{2}$$

$$w = -\frac{5}{384} \frac{qL^3}{EI}, \quad \sigma_f = \frac{Mz}{D} \cdot E_f, \quad \sigma_z = \frac{Mz}{D} \cdot E_c$$

The increased error between the results found by Abaqus and the sandwich beam theory is explained by the increasing influence of local bending of the faces. Sandwich sections with thick sections are not considered in this thesis, but must be kept in mind when the rule $3(d/t)^2 > 100$ is not fulfilled.

Annex C. Grasshopper model

This annex will show the different modules written in Grasshopper, of which an overview is given below. The model is set up in such a way, that each of these 'blocks' may be taken out and replaced by a new one. In other words, if a different structure is to be analysed, or if new material properties are known, this can be easily changed. Additionally, if another optimization algorithm has to be applied, or if the structural analysis has to be carried out in a different software package, the model allows this without a big overhaul of the system.

Blue = Printer properties and input parameters

Orange = Optimization algorithm

Red = Abaqus commands

White = Pre-set Python scripts

Green = (Graphical) Output post processing

Yellow = (Textual) Optimization results



Fig C.1 – Graphical overview of Grasshopper model

C.1. Input Modules

The printer input module allows the user to choose the varying print strategy properties of speed, nozzle size (i.e. layer size) and the temperature of the created object. By choosing one of these to evaluate, the others are fixed, while the chosen property is cut in a user defined number of steps. Additionally, the model is able to vary the fibre content or max aggregate size, but this is not included in the studies of this thesis. The printing parameters are used to calculate the developed concrete strength based on concrete maturity and bond strength. The functions describing these mechanical properties may be changed as well (multiple functions are predefined), and are plotted in the module.

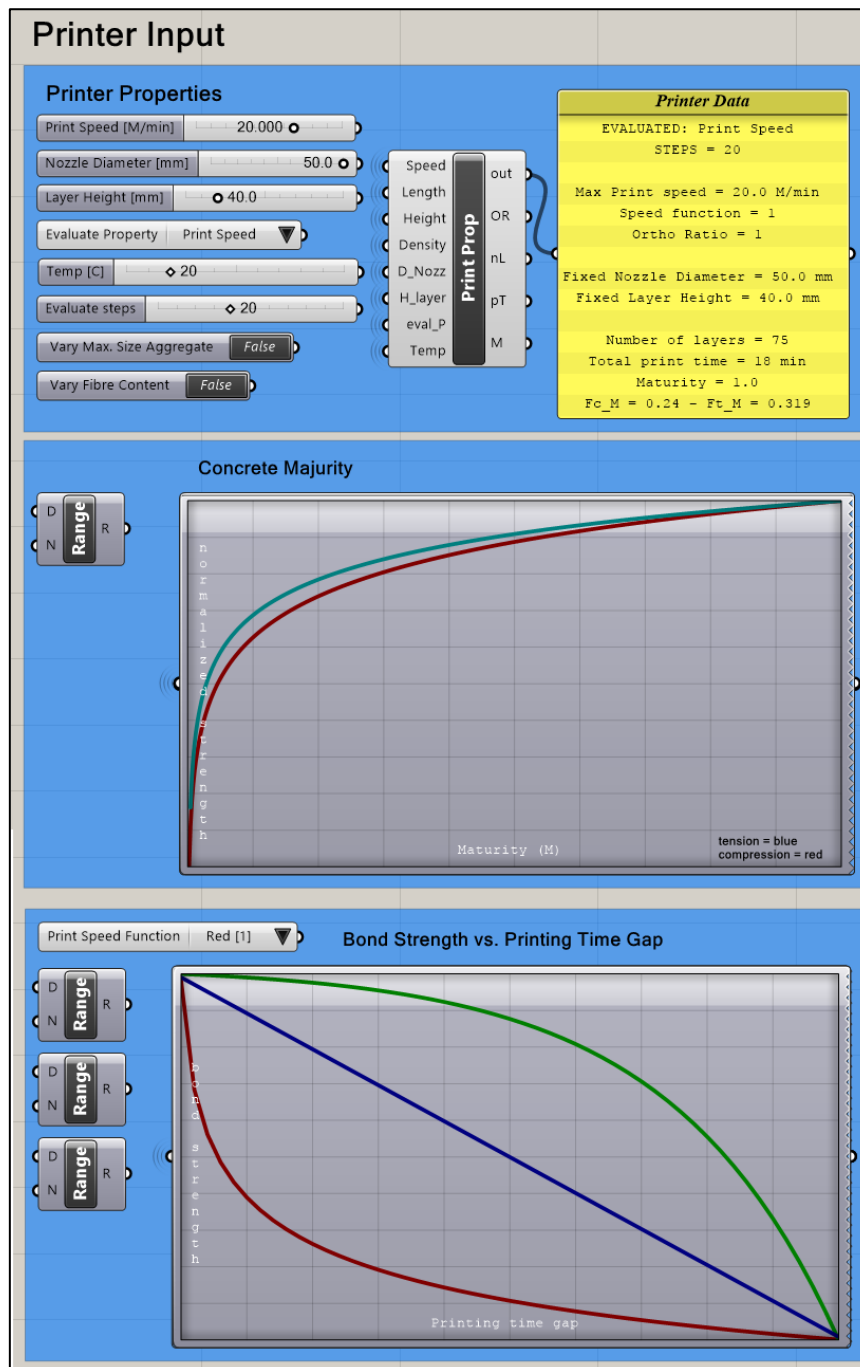


Fig C.2 – Printer input module

The second input module contains all other parameters. First the concrete model is chosen. This can be either linear elastic or non-linear elastic material behaviour, along with a homogeneous or sandwich cross section. The concrete stress-strain diagram can then be given. In case a LE model is chosen, the post cracking behaviour and plasticity is not taken into account; only the failure loads are used. If a sandwich cross section is chosen, the core properties can be given, else these parameters are ignored.

This module also allows the user to define the geometry of the analysed wall, along with the boundary conditions (hinged or clamped, left-right or top-bottom support) and the loading. Additionally, the self-weight can be turned on or off.

Input Parameters

Concrete properties

LE Homogeneous section ▾

E-Modulus 30000 ▾

Poisson's Ratio 0.2 ▾

F_{cd1} 30.0 ▾

F_{cd2} 30.1 ▾

e_{c1} o/oo 1.75 ▾

e_{c2} o/oo 3.50 ▾

F_{td1} 3.00 ▾

F_{td2} 0.01 ▾

e_{t2} o/oo 1.00 ▾

Sandwich core properties

E-Modulus Core 100 ▾

G-Modulus Core 20000 ▾

Poisson's Core 0.2 ▾

Geometry

Length 5000 ▾

Height 3000 ▾

Width 8000 ▾

Support Length 600 ▾

Thickness Face 1 50.0 ▾

Thickness Core 250 ▾

Thickness Face 2 50.0 ▾

BCs and Loading

F-load [kN/m²] 2.00 ▾

L-load [kN/m] 20.0 ▾

W-load [kN/m²] 0.30 ▾

Density 2500 ▾

Self Weight ON ▾

Z Supports T - B ▾

Floor - Wall Clamped ▾

FEM properties

Mesh size 3 ▾

Steps 300 ▾

Concrete stress-strain diagram

Input Parameters

```
#Concrete properties
E = 30000.0 #Youngs Modulus
v = 0.2 #Poissons Ratio
Fcd1 = 30.0 #Fc:rep1
Fcd2 = 30.1 #Fc:rep2
ecd1 = 1.75 #ec:rep1 o/oo
ecd2 = 3.5 #ec:rep2 o/oo
Ftd1 = 3.0 #Ft:rep1
Ftd2 = 0.01 #Ft:rep2
etd2 = 1.0 #et:rep2 o/oo
FR = 1.16, 0.1, 1.28, 0.33
D = 2500.0 #Density

#Sandwich core properties
Ec = 100.0 #Youngs Modulus Core
Gc = 20000.0 #Shear Modulus Core
vc = 0.2 #Poissons Ratio Core

#Geometry
L = 5000.0 #Length
H = 3000.0 #Height
B = 8000.0 #Width
Ls = 600.0 #Support length
tf1 = 50.0 #Thickness Face1
tc = 250.0 #Thickness Core
tf2 = 50.0 #Thickness Face2

#BCs and Loading
Q = 2.0 #Floor Load N/mm2
P = 0.0003 #Wind Pressure N/mm2
Gr = -9.81 #Gravity
SUP = 1 #Z-Support Location
FW = 1 #Floor-Wall Connection

#FEM Properties
M = 3.0 #Mesh size
STEP = 300 #Number of Steps
```

Fig C.3 – Input parameters

C.2. Optimization Algorithm

The optimization module allows the optimized parameter to be chosen, which may be geometry, loading or material properties, but can just as easily be expended with another parameter. The goal is then selected, being either compressive or tensile strength in x or y direction, or displacements in- or out of plane. In case stresses are picked, the material properties are automatically taken as limit, taking strength development and bond strength between layers into account. In case deformations are chosen, the user may define the maximum allowed displacement as an optimization target.

The image shows a software interface for an optimization module, divided into two main sections: 'Target' and 'Simulated Annealing'.

Target Section:

- Opt. Parameter:** A dropdown menu set to 'Floor load'.
- Goal:** A dropdown menu set to 'Tensile strength ortho'.
- Displacement Y:** A slider set to 10.000.
- Displacement Z:** A slider set to 15.000.

Simulated Annealing Section:

- Iterations:** A slider set to 20.
- R1:** A slider set to 1.
- R2:** A slider set to 1000.
- Temperature:** A slider set to 30.
- Change domain:** A dropdown menu set to 1/4.
- Tolerance %:** A slider set to 1.0.

On the right side of each section, there is a yellow box containing a list of parameters and their values, formatted as code:

Target Parameters:

```
OP = 6 #Optimization parameter
GOAL = 1 #Goal
OR = 1.0 #Orthotropic Ratio
T_targ1 = 3.0 #N/mm2 Tension X
T_targ2 = 3.0 #N/mm2 Tension Y
C_targ2 = 30.0 #N/mm2 Compression Y
C_targ1 = 30.0 #N/mm2 Compression X
D_targY = 10.0 #mm Displacement Y
D_targZ = 15.0 #mm Displacement Z
```

Simulated Annealing Parameters:

```
#Simulated Annealing
T = 30.0 #Initial Temperature
T_int = T
k = 1 #Initial step
I = 20.0 #Iterations
R1 = 1.0 #Range
R2 = 1000.0 #Range
SPLIT = 4 #Split of domain
BEST = 100000
STOP = 1.0 #Stop at % tolerance
```

Fig C.4 – Optimization module

The second block of this module contains the simulated annealing parameters, being the number of iterations and the initial domain. The user may also define the algorithm temperature, the point at which the domain starts to narrow, and the tolerance percentage that has to be achieved to abort the optimization loop.

C.3. Toggle Abaqus

The fourth component in the model triggers FEM software Abaqus to execute the script, as generated in Grasshopper. Some user friendly functions have been added: a button is created to toggle Abaqus, to clear the saved text data (optimization results) and the auto-save files of previously carried out analyses. The first python component contains the predefined scripts contains all Abaqus related commands, which are combined with the input properties and optimization algorithm in the second python component. The third component executes Abaqus (see below) and the fourth component clears all data.

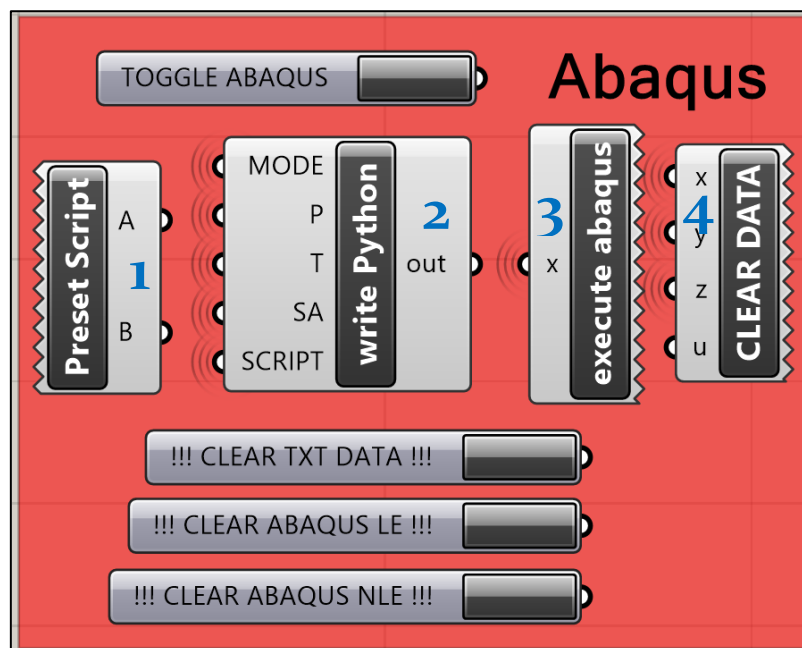


Fig C.5 – Abaqus commands

Abaqus is executed as ‘multi-threading’, but uses only a single worker. In theory, this could be expanded with multiple workers, carrying out multiple analyses at the same time. Using the principle of multi-threading prevents Grasshopper from freezing during the calculations, which does happen otherwise.

```

if x == True:
    import threading
    import os

    os.chdir("C:\SIMULIA\Abaqus\Commands")           #Abaqus directory

    def worker():
        os.system("abaqus cae script=Loop_SA.py")      #Script name
        return

    threads = []
    for i in range(1):
        t = threading.Thread(target=worker)
        threads.append(t)
        t.start()

```

C.4. Python scripts

These four python scripts components are used to combine all input parameters in lists of grouped values, which simplifies the process of writing scripts when parameters are added or removed. Additionally, these components write the text for the yellow text panels that can be seen in the previous modules, supporting the user in a graphical way. The 'plot concrete' component generates the points to plot the concrete stress-strain diagram, which is mainly a user-friendly function in the model.

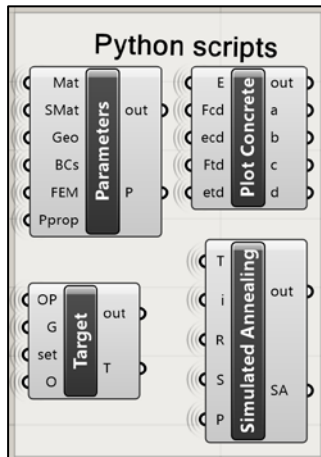


Fig C.6 –Python script blocks

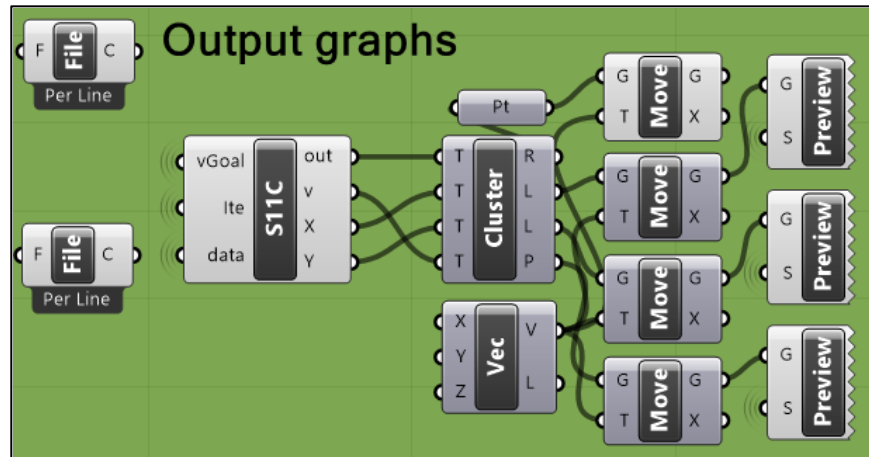


Fig C.7 – (Graphical) Output post processing using GH components

C.5. (Graphical) Output post processing

The Output graph components generate a live view of the optimization results, compared to the predefined goal. While the algorithm runs, analyses results are plotted in a graph such that the user can keep track of the progress. This way an intervention is also easily possible in case it is clear that the algorithm won't converge to the required solution. The component uses the predefined target of the optimization algorithm, and reads the analyses results from a text file that is extended each time a calculation is completed. An example is seen below:

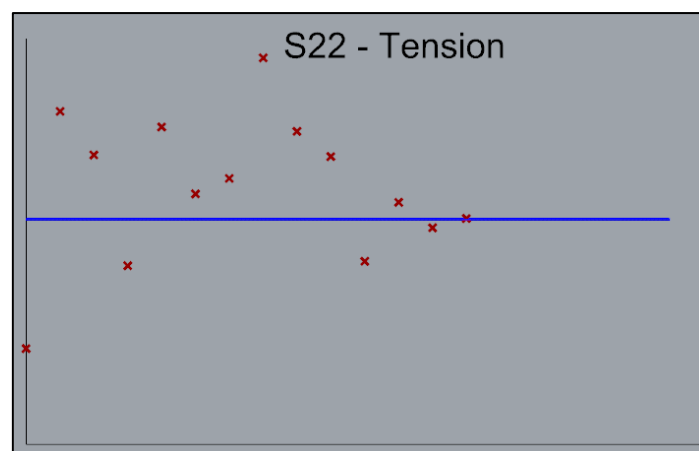


Fig C.6 – Graphical output of analysis results

C.6. (Textual) Optimization results

A final user-friendly extension of the model is the textual result component. This consists of 6 text blocks, which are live updated while the optimization algorithm and analyses run, as can be seen below:

Targets		Results LE	
0	1.038 F tension X	Worse values accepted: [669, 828, 175]	
1	1.038 F tension Y	#####	
2	7.95 F compression Y	1.0% Tolerance achieved	
3	7.95 F compression X	Iterations: 11	
4	10.0 Displacement Y	Target: 1.068	
5	15.0 Displacement Z	Result: 1.07378697395	
		Error: 0.54185149375%	
		Parameter: 0.564	
		Best A value: 282	
		All random values: [440, 918, 610, 894, 336, 229, 241, 311, 300, 264, 282]	
		All best A values: [440, 336, 229, 241, 311, 300, 264, 282]	
		Worse values accepted: []	
		#####	
Printer Properties		Failure Results LE	
0	Speed = 13 - OR = 1 - NW = 50.0 - NH = 40.0 - M = 1.0 - Fc_Mh = 0.322 - Ft_Mh = 0.404	29 NO FAILURE, BETTER VALUE ACCEPTED	
1	Speed = 14 - OR = 1 - NW = 50.0 - NH = 40.0 - M = 1.0 - Fc_Mh = 0.309 - Ft_Mh = 0.391	30 NO FAILURE, BETTER VALUE ACCEPTED	
2	Speed = 15 - OR = 1 - NW = 50.0 - NH = 40.0 - M = 1.0 - Fc_Mh = 0.297 - Ft_Mh = 0.379	31 NO FAILURE, BETTER VALUE ACCEPTED	
3	Speed = 16 - OR = 1 - NW = 50.0 - NH = 40.0 - M = 1.0 - Fc_Mh = 0.286 - Ft_Mh = 0.367	32 NO FAILURE, WORSE VALUE REJECTED	
4	Speed = 17 - OR = 1 - NW = 50.0 - NH = 40.0 - M = 1.0 - Fc_Mh = 0.275 - Ft_Mh = 0.356	33 NO FAILURE, WORSE VALUE REJECTED	
5	Speed = 18 - OR = 1 - NW = 50.0 - NH = 40.0 - M = 1.0 - Fc_Mh = 0.265 - Ft_Mh = 0.346	34 NO FAILURE, WORSE VALUE REJECTED	
		35 NO FAILURE, BETTER VALUE ACCEPTED	
		36 NO FAILURE, BETTER VALUE ACCEPTED	
		37 NO FAILURE, BETTER VALUE ACCEPTED	
		38 NO FAILURE, BETTER VALUE ACCEPTED	
		39 NO FAILURE, BETTER VALUE ACCEPTED	
		40 NO FAILURE, BETTER VALUE ACCEPTED	
		41 NO FAILURE, BETTER VALUE ACCEPTED	
		42 NO FAILURE, BETTER VALUE ACCEPTED	
		43 NO FAILURE, WORSE VALUE ACCEPTED	
		44 NO FAILURE, WORSE VALUE REJECTED	
		45 NO FAILURE, BETTER VALUE ACCEPTED	
		Range LE	
		30 283 - 335	
		31 1 - 1000	
		32 1 - 1000	
		33 1 - 1000	
		34 1 - 1000	
		35 1 - 1000	
		36 225 - 336	
		37 229 - 336	
		38 241 - 314	
		39 241 - 311	
		40 252 - 300	
		41 264 - 300	
		42 1 - 1000	
		43 1 - 1000	
		44 1 - 1000	
		45 1 - 1000	
		46 1 - 1000	
		RandV	
		30 303	
		31 440	
		32 918	
		33 610	
		34 894	
		35 336	
		36 229	
		37 241	
		38 311	
		39 300	
		40 264	
		41 282	
		42 99	
		43 85	
		44 776	
		45 90	
		46 818	

Fig C.7 – Textual optimization results

The first text panel shows the targets of the current optimization loop, which are only updated as the printer properties are varied. These printer properties are plotted in text panel 2, for each variational loop.

The 3rd panel shows the results of a single optimization loop. This contains the number of iterations required to reach the target, the best result and its error percentage, the corresponding parameter value and all random chosen values, amongst others. This panel is updated when a loop is completed.

Panel 4, 5 and 6 are updated for each calculation. They show the domain (5) in which the random value is chosen (6). Panel 4 is written when the calculation is complete, and tells the user if the analysis was better or worse than the previous ones, and if the value is accepted or rejected by the optimization algorithm.

Annex D. Python script

The final annex of this graduation thesis contains the entire Python script that is used to execute both Abaqus and run the optimization algorithm. The text in green starting with an '#' are comments. All blue text contains functions, while all red data is a variable parameter. Words written in grey between parentheses is textual data.

The script is schematically set up as follows:

- 1) Write parameters
- 2) Import Abaqus functions
- 3) Define functions for printer properties: bond strength and maturity
- 4) Define functions for simulated annealing: temperature, evaluation and domain
- 5) Initiate printer evaluation loop
 - 5.1) Write optimization parameters and goal
 - 5.2) Initiate optimization loop
 - 5.2.1) Update optimization variables
 - 5.2.2) Abaqus analysis based on chosen concrete model
 - 5.2.3) Evaluate results
 - 5.2.4) Repeat optimization loop or stop when tolerance achieved
 - 5.3) Repeat evaluation loop or stop when number of steps is reached
- 6) Write output

```
### LINEAR ELASTIC - HOMOGENEOUS SECTION ###
```

```
MODE = 1 # 1 = LE H, 2 = LE SW, 3 = NLE H, 4 = NLE SW
```

```
#3D Printer properties
```

```
pEVAL = 0 #Evaluated printer property
max_speed = 20.0 #M/Min Printer speed
speed_function = 1 #Printer speed function
max_width = 50.0 #mm Nozzle width
max_height = 50.0 #mm Layer height
t_incr = 15 #Number of evaluation steps
incr = 1 #Initial evaluation step
vary_MSA = False #Vary max. size aggregate
vary_FRC = False #Vary fibre content
Temperature = 20.0 #Concrete temperature
```

```
#Concrete properties
```

```
E = 30000.0 #Youngs Modulus
v = 0.2 #Poissons Ratio
Fcd1 = 30.0 #Fc;rep1
Fcd2 = 30.1 #Fc;rep2
ecd1 = 1.75 #ec;rep1 o/oo
ecd2 = 3.5 #ec;rep2 o/oo
Ftd1 = 3.0 #Ft;rep1
Ftd2 = 0.01 #Ft;rep2
etd2 = 1.0 #et;rep2 o/oo
FR = 1.16, 0.1, 1.28, 0.33
D = D_i = 2500.0 #Density
```

```
#Sandwich core properties
```

```
Ec = 100.0 #Youngs Modulus Core
Gc = 20000.0 #Shear Modulus Core
vc = 0.2 #Poissons Ratio Core
```

```
#Geometry
```

```
L = L_i = 5000.0 #Length
H = H_i = 3000.0 #Height
B = 8000.0 #Width
Ls = Ls_i = 600.0 #Support length
tf1 = t_i = 50.0 #Thickness Facel
tc = tc_i = 250.0 #Thickness Core
tf2 = t2_i = 50.0 #Thickness Face2
```

```
#BCs and Loading
```

```
Q = Q_i = 2.0 #Floor load N/mm2
P = P_i = 0.0003 #Wind Pressure N/mm2
Gr = -9.81 #Gravity
SUP = 1 #Z-Support Location
FW = 1 #Wall-Floor connection
```

```
#FEM properties
```

```
M = M_i = 3.0 #Mesh size
nSTEPS = Step_i = 300 #Number of Steps
```

```
from abaqus import *
from abaqusConstants import *
from part import *
from material import *
```

```
from section import *
from assembly import *
from step import *
from interaction import *
from load import *
from optimization import *
from job import *
from sketch import *
from visualization import *
from odbAccess import *

import math
import random
import decimal
import os

def print_eval(incr):
    print_props = []
    speed_incr = max_speed/t_incr
    width_incr = max_width/t_incr
    height_incr = max_height/t_incr

    if pEVAL == 0:                                     #speed
        speed = (incr*speed_incr)
        width = max_width
        height = max_height

    elif pEVAL == 1:                                   #nozzle_width
        speed = max_speed
        width = (incr*width_incr)
        height = max_height

    elif pEVAL == 2:                                   #layer_height
        speed = max_speed
        width = max_width
        height = (incr*height_incr)

    #SPEED - RATIO OF ORTHOTROPY
    time_gap = (L/1000) / speed                        #time gap between layers
    n_layers = int(H / height)                         #number of layers
    pT = time_gap*n_layers                             #total print time [min]

    if speed_function == 1:                             #ratio of orthotropy based on time gap
        OR = round((1)/((time_gap)**0.30), 2)
    elif speed_function == 2:
        OR = round(1.02-0.01*math.e**(0.0395*5*time_gap), 2)
    elif speed_function == 3:
        OR = round(1.005-0.05*time_gap, 2)
    if OR > 1:
        OR = 1

    #SPEED - MATURITY
    Ta = Temperature
    Tref = 20.

    if Ta < 20:
        Em = (33.5+1.47*(20-Ta))*1000
```

```

elif Ta >= 20:
    Em = 33500.
    Rm = 8.3144

Maturity = math.exp((Em/Rm)*(1/(273+Tref)-1/(273+Ta))) #Maturity
hours = pT/60
Mh = Maturity * hours

Fc_Mh = round((44.030 + 17.020*math.log(Mh))/(44.030 + 17.020*math.log(Maturity*24)), 3)
Ft_Mh = round((Fc_Mh**0.8), 3)

NW = width
NH = height

print_props.append(speed), print_props.append(OR), print_props.append(NW), print_props.
append(NH), print_props.append(round(Maturity, 3)), print_props.append(Fc_Mh),
print_props.append(Ft_Mh)
return print_props

def update_temperature(T,k):
    return T_int-(k*(T_int/I))

def eval(A):
    evalList = []
    if GOAL == 0:
        evalList.append(T_targ1 - max_S11_t)
        evalList.append(max_S22_t <= T_targ2)
        evalList.append(abs(max_S11_c) <= C_targ1)
        evalList.append(abs(max_S22_c) <= C_targ2)
        evalList.append(abs(max_U2) <= D_targY)
        evalList.append(abs(max_U3) <= D_targZ)
    if GOAL == 1:
        evalList.append(T_targ2 - max_S22_t)
        evalList.append(max_S11_t <= T_targ1)
        evalList.append(abs(max_S11_c) <= C_targ1)
        evalList.append(abs(max_S22_c) <= C_targ2)
        evalList.append(abs(max_U2) <= D_targY)
        evalList.append(abs(max_U3) <= D_targZ)
    if GOAL == 2:
        evalList.append(C_targ2 - abs(max_S22_c))
        evalList.append(max_S11_t <= T_targ1)
        evalList.append(max_S22_t <= T_targ2)
        evalList.append(abs(max_S11_c) <= C_targ1)
        evalList.append(abs(max_U2) <= D_targY)
        evalList.append(abs(max_U3) <= D_targZ)
    if GOAL == 3:
        evalList.append(C_targ1 - abs(max_S11_c))
        evalList.append(max_S11_t <= T_targ1)
        evalList.append(max_S22_t <= T_targ2)
        evalList.append(abs(max_S22_c) <= C_targ2)
        evalList.append(abs(max_U2) <= D_targY)
        evalList.append(abs(max_U3) <= D_targZ)
    if GOAL == 4:
        evalList.append(D_targY - abs(max_U2))
        evalList.append(max_S11_t <= T_targ1)
        evalList.append(max_S22_t <= T_targ2)
        evalList.append(abs(max_S11_c) <= C_targ1)

```

```

    evalList.append(abs(max_S22_c) <= C_targ2)
    evalList.append(abs(max_U3) <= D_targZ)
if GOAL == 5:
    evalList.append(D_targZ - abs(max_U3))
    evalList.append(max_S11_t <= T_targ1)
    evalList.append(max_S22_t <= T_targ2)
    evalList.append(abs(max_S11_c) <= C_targ1)
    evalList.append(abs(max_S22_c) <= C_targ2)
    evalList.append(abs(max_U2) <= D_targY)
return evalList

def failure_check(A):
if GOAL == 0:
    failureList = ['S22t', 'S11c', 'S22c', 'U2', 'U3']
if GOAL == 1:
    failureList = ['S11t', 'S11c', 'S22c', 'U2', 'U3']
if GOAL == 2:
    failureList = ['S11t', 'S22t', 'S11c', 'U2', 'U3']
if GOAL == 3:
    failureList = ['S11t', 'S22t', 'S22c', 'U2', 'U3']
if GOAL == 4:
    failureList = ['S11t', 'S22t', 'S11c', 'S22c', 'U3']
if GOAL == 5:
    failureList = ['S11t', 'S22t', 'S11c', 'S22c', 'U2']
for i, j in enumerate(eval(A)[1:len(eval(A))]):
    if j == False:
        fList.append('FAILURE')
        fList.append(failureList[i])
return fList

def get_range(k):
global R1
global R2
global R3
global R4

if k <= I/(SPLIT):
    if MODE == 1 or MODE == 2:
        rangeFile = open('LE_range.txt', 'a+')
    elif MODE == 3 or MODE == 4:
        rangeFile = open('SC_range.txt', 'a+')
    rangeFile.write(str(R1)+' - '+str(R2)+'\n')
    rangeFile.close()
    return range(R1, R2)

elif k > I/(SPLIT):
    if len(A_best_list)>= 2:
        bestEval1 = min(eval_list, key=lambda x:abs(abs(x)-0))
        bestEvalIndex1 = eval_list.index(bestEval1)
        eval_list.remove(bestEval1)
        bestA1 = A_best_list[bestEvalIndex1]
        bestA1Index = A_best_list.index(bestA1)
        A_best_list.remove(bestA1)

        bestEval2 = min(eval_list, key=lambda x:abs(abs(x)-0))
        bestEvalIndex2 = eval_list.index(bestEval2)
        bestA2 = A_best_list[bestEvalIndex2]

```

```

eval_list.insert(bestEvalIndex1, bestEval1)
A_best_list.insert(bestA1Index, bestA1)

if (bestEval1 >= 0 and bestEval2 < 0) or (bestEval1 <= 0 and bestEval2 > 0):
    R3 = min(bestA1, bestA2)
    R4 = max(bestA1, bestA2)

elif bestEval1 >= 0 and bestEval2 > 0 and bestA1 >= bestA2:
    R3 = bestA1
    if bestA1 - bestA2 == 0:
        R4 = bestA2
    else:
        alfa = math.atan((abs(bestEval2)-abs(bestEval1))/(bestA1-bestA2))
        delta = bestEval1/math.tan(alfa)
        R4 = int(bestA1+2*delta)

elif bestEval1 >= 0 and bestEval2 > 0 and bestA1 < bestA2:
    R4 = bestA1
    if bestA1 - bestA2 == 0:
        R3 = bestA2
    else:
        alfa = math.atan((abs(bestEval2)-abs(bestEval1))/(bestA2-bestA1))
        delta = bestEval1/math.tan(alfa)
        R3 = int(bestA1-2*delta)

elif bestEval1 <= 0 and bestEval2 < 0 and bestA1 <= bestA2:
    R4 = bestA1
    if abs(bestA1 - bestA2) == 0:
        R3 = bestA2
    else:
        alfa = math.atan((bestEval2-bestEval1)/(bestA2-bestA1))
        delta = bestEval1/math.tan(alfa)
        R3 = int(bestA1-2*delta)

elif bestEval1 <= 0 and bestEval2 <0 and bestA1 > bestA2:
    R3 = bestA1
    if abs(bestA1 - bestA2) == 0:
        R4 = bestA2
    else:
        alfa = math.atan((bestEval2-bestEval1)/(bestA1-bestA2))
        delta = bestEval1/math.tan(alfa)
        R4 = int(bestA1+2*delta)

if R3 <= 0:
    R3 = R1

else:
    R3 = R1
    R4 = R2
if MODE == 1 or MODE == 2:
    rangeFile = open('LE_range.txt', 'a+')
elif MODE == 3 or MODE == 4:
    rangeFile = open('SC_range.txt', 'a+')
rangeFile.write(str(R3)+' - '+str(R4)+'\n')
rangeFile.close()
return range(R3,R4)

```

#Viewport options

session.graphicsOptions.setValues(backgroundStyle=SOLID, backgroundColor='#FFFFFF')

while incr <= t_incr:

 speed = print_eval(incr)[0]

 OR = print_eval(incr)[1]

 tf1 = print_eval(incr)[2]

 tf2 = print_eval(incr)[2]

 NH = print_eval(incr)[3]

 Maturity = print_eval(incr)[4]

 Fc_Mh = print_eval(incr)[5]

 Ft_Mh = print_eval(incr)[6]

printFile = open('printProps.txt', 'a')

printFile.write('Speed = '+str(speed)+' - OR = '+str(OR)+' - NW = '+str(tf1)+' - NH = '+str(NH)+' - M = '+str(Maturity)+' - Fc_Mh = '+str(Fc_Mh)+' - Ft_Mh = '+str(Ft_Mh)+'\n')

printFile.close()

A_all_list = []

A_best_list = []

eval_list = []

A_worse_list = []

result_list = []

T_list = []

similar_list = []

#Simulated Annealing

T = 30.0 #Initial Temperature

T_int = T

k = 1 #Initial step

I = 20 #Iterations

R1 = 300 #Range

R2 = 1000 #Range

SPLIT = 2 #Split of domain

ST = 0.5 #Stop at %Tolerance

BEST = 100000

#Goal and Target

OP = 6 #Optimization parameter

GOAL = 1 #Goal

T_targ1 = Ftd1*Ft_Mh #N/mm2 Tension X-direction

T_targ2 = T_targ1 * OR #N/mm2 Tension Y-direction

C_targ2 = Fcd1*Fc_Mh #N/mm2 Compression Y-direction

C_targ1 = C_targ2 * OR #N/mm2 Compression X-direction

D_targY = 10.0 #mm Displacement Y

D_targZ = 15.0 #mm Displacement Z

targetFile = open('targets.txt', 'w')

targetFile.write(str(T_targ1)+'\n'+str(T_targ2)+'\n'+str(C_targ2)+'\n'+str(C_targ1)+'\n'+str(D_targY)+'\n'+str(D_targZ))

targetFile.close()

while T > 0:

 set = get_range(k)

 if len(set)<1:

 print 'ABORTED'

```
if MODE == 1 or MODE == 2:
    outputFile = open('LE_SA_PostProcessing.txt', 'a+')
    outputFile2 = open('LE_SA_RandomList.txt', 'a+')
elif MODE == 3 or MODE == 4:
    outputFile = open('SC_SA_PostProcessing.txt', 'a+')
    outputFile2 = open('SC_SA_RandomList.txt', 'a+')
outputFile.write("ABORTED"+'\n')
outputFile2.write('X'+'\n')
outputFile.close()
outputFile2.close()
break
A = random.choice(set)
A_all_list.append(A)
if MODE == 1 or MODE == 2:
    outputFile = open('LE_SA_RandomList.txt', 'a+')
elif MODE == 3 or MODE == 4:
    outputFile = open('SC_SA_RandomList.txt', 'a+')
outputFile.write(str(A)+'\n')
outputFile.close()

fList = []

if OP == 0:
    parameter = L_i
    L = (A * L_i)/1000

if OP == 1:
    parameter = H_i
    H = (A * H_i)/1000

if OP == 2:
    parameter = Ls_i
    Ls = (A * Ls_i)/1000

if OP == 3:
    parameter = t_i
    tf1 = (A * t_i)/1000

if OP == 4:
    parameter = tc_i
    tc = (A * tc_i)/1000

if OP == 5:
    parameter = t2_i
    tf2 = (A * t2_i)/1000

if OP == 6 and (MODE == 1 or MODE == 2):
    parameter = Q_i
    Q = (A * Q_i)/1000

elif OP == 6 and (MODE == 3 or MODE == 4):
    parameter = Step_i
    nSTEPS = (A * Step_i)/1000

if OP == 7 and (MODE == 1 or MODE == 2):
    parameter = P_i
    P = (A * P_i)/1000
```

```

elif OP == 7 and (MODE == 3 or MODE == 4):
    parameter = Step_i
    nSTEPS = (A * Step_i)/1000

if OP == 8:
    parameter = D_i
    D = (A * D_i)/1000

#Model
if MODE == 1 or MODE == 2:
    myModel = mdb.Model(name='wall_linearElastic'+str(incr)+'_'+str(k))
elif MODE == 3 or MODE == 4:
    myModel = mdb.Model(name='wall_smearedCrack'+str(incr)+'_'+str(k))
if 'Model-1' in mdb.models:
    del mdb.models['Model-1']

#Create main material - Linear Elastic
if MODE == 1 or MODE == 2:
    myMaterialLE1 = myModel.Material(name='concrete_linearElastic')
    elasticProperties = (float(E), float(v))
    myMaterialLE1.Elastic(table=(elasticProperties, ))
    myMaterialLE1.Density(table=((float(D)*10**-9), ))

#Create main material - Smeared Crack
elif MODE == 3 or MODE == 4:
    myMaterialSC = myModel.Material(name='concrete_smearedCrack')
    elasticProperties = (float(E), float(v))
    myMaterialSC.Elastic(table=(elasticProperties, ))
    compressionConcretel = (float(Fcd1), 0)
    compressionConcrete2 = (float(Fcd2), float(ecd2)*10**-3)
    myMaterialSC.Concrete(table=((compressionConcretel), (compressionConcrete2)))
    failureRatios = (FR)
    myMaterialSC.concrete.FailureRatios(table=(FR, ))
    Ftd12 = 1 - (float(Ftd1)-float(Ftd2))/float(Ftd1)
    myMaterialSC.concrete.TensionStiffening(table=((1.0, 0.0), (float(Ftd12), float(
    etd2)*10**-3)))
    myMaterialSC.Density(table=((float(D)*10**-9), ))

#Create sandwich core material - Linear Elastic
if MODE == 2 or MODE == 4:
    myMaterialLE2 = myModel.Material(name='core_linearElastic')
    elasticProperties = (float(Ec), float(v))
    myMaterialLE2.Elastic(table=(elasticProperties, ))

#Create support material - Linear Elastic
myMaterialLE3 = myModel.Material(name='support_linearElastic')
elasticProperties = (1, 0)
myMaterialLE3.Elastic(table=(elasticProperties, ))

#Create main section
if MODE == 1:
    myModel.HomogeneousShellSection(name='shellSection_Homogeneous', preIntegrate=OFF
    , material='concrete_linearElastic', thicknessType=UNIFORM, thickness=float(tf1),
    thicknessField='', idealization=NO_IDEALIZATION, poissonDefinition=DEFAULT,
    thicknessModulus=None, temperature=GRADIENT, useDensity=OFF, integrationRule=
    SIMPSON, numIntPts=9)

```

```

t_support = tf1

elif MODE == 2:
    sectionLayer1 = SectionLayer(material='concrete_linearElastic', thickness=float(
    tf1), orientAngle=0.0, numIntPts=9, plyName='Face1')

    sectionLayer2 = SectionLayer(material='core_linearElastic', thickness=float(tc),
    orientAngle=0.0, numIntPts=5, plyName='Core')

    sectionLayer3 = SectionLayer(material='concrete_linearElastic', thickness=float(
    tf2), orientAngle=0.0, numIntPts=9, plyName='Face2')

    myModel.CompositeShellSection(name='shellSection_Sandwich', preIntegrate=OFF,
    idealization=NO_IDEALIZATION, symmetric=False, thicknessType=UNIFORM,
    poissonDefinition=DEFAULT, thicknessModulus=None, temperature=GRADIENT,
    useDensity=OFF, integrationRule=SIMPSON, layup=(sectionLayer1, sectionLayer2,
    sectionLayer3, ))

    t_support = tf1 + tc + tf2

elif MODE == 3:
    myModel.HomogeneousShellSection(name='shellSection_Homogeneous', preIntegrate=OFF
    , material='concrete_smearedCrack', thicknessType=UNIFORM, thickness=float(tf1),
    thicknessField='', idealization=NO_IDEALIZATION, poissonDefinition=DEFAULT,
    thicknessModulus=None, temperature=GRADIENT, useDensity=OFF, integrationRule=
    SIMPSON, numIntPts=9)

    t_support = tf1

elif MODE == 4:
    sectionLayer1 = SectionLayer(material='concrete_smearedCrack', thickness=float(
    tf1), orientAngle=0.0, numIntPts=9, plyName='Face1')

    sectionLayer2 = SectionLayer(material='core_linearElastic', thickness=float(tc),
    orientAngle=0.0, numIntPts=5, plyName='Core')

    sectionLayer3 = SectionLayer(material='concrete_smearedCrack', thickness=float(
    tf2), orientAngle=0.0, numIntPts=9, plyName='Face2')

    myModel.CompositeShellSection(name='shellSection_Sandwich', preIntegrate=OFF,
    idealization=NO_IDEALIZATION, symmetric=False, thicknessType=UNIFORM,
    poissonDefinition=DEFAULT, thicknessModulus=None, temperature=GRADIENT,
    useDensity=OFF, integrationRule=SIMPSON, layup=(sectionLayer1, sectionLayer2,
    sectionLayer3, ))

    t_support = tf1 + tc + tf2

#Create support section
myModel.HomogeneousShellSection(name='supportSection_Homogeneous', preIntegrate=OFF,
material='support_linearElastic', thicknessType=UNIFORM, thickness=float(t_support),
thicknessField='', idealization=NO_IDEALIZATION, poissonDefinition=DEFAULT,
thicknessModulus=None, temperature=GRADIENT, useDensity=OFF, integrationRule=SIMPSON,
numIntPts=9)

#Create main part
mySketch = myModel.ConstrainedSketch(name='__profile__', sheetSize=5000.0)

```

```

mySketch.rectangle(point1=(0.0, 0.0), point2=(float(L), float(H)))
if MODE == 1 or MODE == 2:
    myPart = myModel.Part(name='wall_linearElastic'+str(incr)+'_'+str(k),
        dimensionality=THREE_D, type=DEFORMABLE_BODY)
elif MODE == 3 or MODE == 4:
    myPart = myModel.Part(name='wall_smearedCrack'+str(incr)+'_'+str(k),
        dimensionality=THREE_D, type=DEFORMABLE_BODY)
myPart.BaseShell(sketch=mySketch)

#Partition main part
myPart.DatumPlaneByPrincipalPlane(principalPlane=YZPLANE, offset=(float(Ls)))
myPart.DatumPlaneByPrincipalPlane(principalPlane=YZPLANE, offset=(float(L)-(float(Ls)
))))

myPart.PartitionEdgeByDatumPlane(edges=myPart.edges.findAt((((int(L)/2), (int(H)),
0.0), )), datumPlane=myPart.datums[2])
myPart.PartitionEdgeByDatumPlane(edges=myPart.edges.findAt((((int(L)/2), 0.0, 0.0),
)), datumPlane=myPart.datums[2])
myPart.PartitionEdgeByDatumPlane(edges=myPart.edges.findAt((((int(L)/2), (int(H)),
0.0), )), datumPlane=myPart.datums[3])
myPart.PartitionEdgeByDatumPlane(edges=myPart.edges.findAt((((int(L)/2), 0.0, 0.0),
)), datumPlane=myPart.datums[3])

#Create support part
mySupportSketch = myModel.ConstrainedSketch(name='__profile__', sheetSize=5000.0)
mySupportSketch.rectangle(point1=(0.0, 0.0), point2=((float(Ls)), 100))
mySupportPart = myModel.Part(name='support_linearElastic'+str(incr)+'_'+str(k),
    dimensionality=THREE_D, type=DEFORMABLE_BODY)
mySupportPart.BaseShell(sketch=mySupportSketch)

#Create faces sets
myPart.Set(faces=myPart.faces.findAt((((int(L)/2), (int(H)/2), 0.0), )), name=
'shellSet_all')
mySupportPart.Set(faces=mySupportPart.faces.findAt((((int(Ls)), 50.0, 0.0), )), name=
'supportSet_all')

#Assign section
if MODE == 1 or MODE == 3:
    myPart.SectionAssignment(region=myPart.sets['shellSet_all'], sectionName=
'shellSection_Homogeneous', offset=0.0, offsetType=MIDDLE_SURFACE, offsetField='',
, thicknessAssignment=FROM_SECTION)
elif MODE == 2 or MODE == 4:
    myPart.SectionAssignment(region=myPart.sets['shellSet_all'], sectionName=
'shellSection_Sandwich', offset=0.0, offsetType=MIDDLE_SURFACE, offsetField='',
thicknessAssignment=FROM_SECTION)

mySupportPart.SectionAssignment(region=mySupportPart.sets['supportSet_all'],
sectionName='supportSection_Homogeneous', offset=0.0, offsetType=MIDDLE_SURFACE,
offsetField='', thicknessAssignment=FROM_SECTION)

#Mesh parts
import mesh
elemType1 = mesh.ElemType(elemCode=S8R, elemLibrary=STANDARD)          #S8R Elements
#elemType1 = mesh.ElemType(elemCode=S4R, elemLibrary=STANDARD)          #S4R Elements

myPart.seedPart(size=(int(Ls)/int(M)), deviationFactor=0.1, minSizeFactor=0.1)
myPart.generateMesh()

```

```

myPart.setElementType(regions=myPart.sets['shellSet_all'], elemTypes=(elemType1, ))

mySupportPart.seedPart(size=(int(Ls)/int(M)), deviationFactor=0.1, minSizeFactor=0.1)
mySupportPart.generateMesh()
mySupportPart.setElementType(regions=mySupportPart.sets['supportSet_all'], elemTypes
=(elemType1, ))

#Assembly
myAssembly = myModel.rootAssembly
myAssembly.DatumCsysByDefault(CARTESIAN)
myInstance = myAssembly.Instance(name='wall', part=myPart, dependent=ON)
myInstance2 = myAssembly.Instance(name='support1', part=mySupportPart, dependent=ON)
myInstance3 = myAssembly.Instance(name='support2', part=mySupportPart, dependent=ON)

myAssembly.translate(instanceList=('support1', ), vector=(0.0, -100.0, 0.0))
myAssembly.translate(instanceList=('support2', ), vector=((float(L)-(float(Ls)), -
100.0, 0.0)))

#Sets for BC's and loading
myAssembly.Surface(sidelEdges=myInstance.edges.findAt((((int(L)/2), (int(H)), 0.0),
)), name='topEdge')
myAssembly.Surface(sidelFaces=myInstance.faces.findAt((((int(L)/2), (int(H)/2), 0.0),
)), name='shellFace')
myAssembly.Set(edges=myInstance.edges.findAt(((0.0, (float(H)/2), 0.0), )), name=
'leftSupportZ')
myAssembly.Set(edges=myInstance.edges.findAt((((float(L)), (float(H)/2), 0.0), )),
name='rightSupportZ')
myAssembly.Set(edges=myInstance2.edges.findAt((((float(Ls)/2), -100, 0.0), )), name=
'leftSupportX')
myAssembly.Set(edges=myInstance2.edges.findAt(((0.0, -50, 0.0), )), name=
'leftSupportY')
myAssembly.Set(edges=myInstance3.edges.findAt((((float(L)-float(Ls)/2), -100, 0.0),
)), name='rightSupportX')
myAssembly.Set(edges=(myInstance.edges.findAt((((float(Ls/2)), (float(H)), 0.0),
)))+(myInstance.edges.findAt((((float(L/2)), (float(H)), 0.0), ))+myInstance.edges.
findAt((((float(L-Ls/2)), (float(H)), 0.0), )), name='topSupportZ')
myAssembly.Set(edges=(myInstance.edges.findAt((((float(Ls/2)), 0.0, 0.0), ))+((
myInstance.edges.findAt((((float(L/2)), 0.0, 0.0), ))+myInstance.edges.findAt((((
float(L-Ls/2)), 0.0, 0.0), ))), name='bottomSupportZ')

#Nodesets for output
n1 = myInstance.nodes
nodelist = []
for nodes in n1:
    nodelist.append(nodes)

Polygon = []
Polygon.append((-1,-1+(0.1*H)))
Polygon.append(((1.1*L),-1+(0.1*H)))
Polygon.append(((1.1*L),(1.1*H)))
Polygon.append((-1,(1.1*H)))

#Polygon1

nodes_notinside=[]
nodes_inside=[]

for nodes in nodelist:
    x = nodes.coordinates[0]

```

```

y = nodes.coordinates[1]
z = nodes.coordinates[2]
N = nodes.label

def point_inside_polygon(x,y,Polygon):
    n = len(Polygon)
    inside = False
    plx,ply = Polygon[0]

    for i in range(n+1):
        p2x,p2y = Polygon[i % n]
        Polygon[i % n]

        if y > min(ply,p2y):
            if y <= max(ply,p2y):
                if x <= max(plx,p2x):
                    if ply != p2y:
                        xinters = (y-ply)*(p2x-plx)/(p2y-ply)+plx
                        if plx == p2x or x <= xinters:
                            inside = not inside

        plx,ply = p2x,p2y

    return inside
if point_inside_polygon(x,y,Polygon)==True:
    nodes_inside.append(nodes)
else:
    nodes_notinside.append(nodes)

```

```

odelist1 = []

```

```

for nodes in nodes_inside:
    x = nodes.coordinates[0]
    y = nodes.coordinates[1]
    z = nodes.coordinates[2]
    N = nodes.label
    b = (n1[(N-1):(N)])
    oodelist1.append(b)

```

```

Polygon = [] #Polygon2
Polygon.append((1.2*Lv,-1))
Polygon.append(((L-1.2*Lv),-1))
Polygon.append(((L-1.2*Lv),-1+(0.1*H)))
Polygon.append((1.2*Lv,-1+(0.1*H)))

```

```

nodes_notinside=[]
nodes_inside=[]

```

```

for nodes in oodelist:
    x = nodes.coordinates[0]
    y = nodes.coordinates[1]
    z = nodes.coordinates[2]
    N = nodes.label

    def point_inside_polygon(x,y,Polygon):
        n = len(Polygon)
        inside = False
        plx,ply = Polygon[0]

```

```

        for i in range(n+1):
            p2x,p2y = Polygon[i % n]
            Polygon[i % n]

            if y > min(ply,p2y):
                if y <= max(ply,p2y):
                    if x <= max(plx,p2x):
                        if ply != p2y:
                            xinters = (y-ply)*(p2x-plx)/(p2y-ply)+plx
                            if plx == p2x or x <= xinters:
                                inside = not inside

            plx,ply = p2x,p2y

        return inside
    if point_inside_polygon(x,y,Polygon)==True:
        nodes_inside.append(nodes)
    else:
        nodes_notinside.append(nodes)

for nodes in nodes_inside:
    x = nodes.coordinates[0]
    y = nodes.coordinates[1]
    z = nodes.coordinates[2]
    N = nodes.label
    b = (n1[(N-1):(N)])
    nodelist1.append(b)

myAssembly.Set(nodes=nodelist1, name='NodeSet1')
myAssembly.Set(nodes=n1[0:(int(len(nodelist1)))] , name='NodeSet_all')

#Elementsets for output
n1 = myInstance.nodes
e1 = myInstance.elements

Polygon = [] #Polygon1
Polygon.append((-1,-1+(0.1*H)))
Polygon.append(((1.1*L),-1+(0.1*H)))
Polygon.append(((1.1*L),(1.1*H)))
Polygon.append((-1,(1.1*H)))

nodes_notinside=[]
nodes_inside=[]

elementsInsidePolygon = []

for elements in e1:
    i_node = 0
    counter = 0

    while i_node <= 3:
        NODE = elements.connectivity[i_node]
        x = n1[NODE].coordinates[0]
        y = n1[NODE].coordinates[1]
        z = n1[NODE].coordinates[2]
        N = n1[NODE].label

```

```

def point_inside_polygon(x,y,Polygon):
    n = len(Polygon)
    inside = False
    plx,ply = Polygon[0]

    for i in range(n+1):
        p2x,p2y = Polygon[i % n]
        Polygon[i % n]

        if y > min(ply,p2y):
            if y <= max(ply,p2y):
                if x <= max(plx,p2x):
                    if ply != p2y:
                        xinters = (y-ply)*(p2x-plx)/(p2y-ply)+plx
                        if plx == p2x or x <= xinters:
                            inside = not inside
                    plx,ply = p2x,p2y

    return inside

if point_inside_polygon(x,y,Polygon) == True:
    counter = counter + 1
else:
    counter = counter
i_node += 1

if counter > 1:
    Ele = elements.label
    b = (el[(Ele-1):(Ele)])
    elementsInsidePolygon.append(b)

Polygon = [] #Polygon2
Polygon.append((1.2*Lv,-1))
Polygon.append(((L-1.2*Lv),-1))
Polygon.append(((L-1.2*Lv),-1+(0.1*H)))
Polygon.append((1.2*Lv,-1+(0.1*H)))

nodes_notinside=[]
nodes_inside=[]

for elements in el:
    i_node = 0
    counter = 0

    while i_node <= 3:
        NODE = elements.connectivity[i_node]
        x = n1[NODE].coordinates[0]
        y = n1[NODE].coordinates[1]
        z = n1[NODE].coordinates[2]
        N = n1[NODE].label

        def point_inside_polygon(x,y,Polygon):
            n = len(Polygon)
            inside = False
            plx,ply = Polygon[0]

            for i in range(n+1):

```

```

        p2x,p2y = Polygon[i % n]
        Polygon[i % n]

        if y > min(ply,p2y):
            if y <= max(ply,p2y):
                if x <= max(plx,p2x):
                    if ply != p2y:
                        xinters = (y-ply)*(p2x-plx)/(p2y-ply)+plx
                        if plx == p2x or x <= xinters:
                            inside = not inside
                    plx,ply = p2x,p2y

        return inside

    if point_inside_polygon(x,y,Polygon) == True:
        counter = counter + 1
    else:
        counter = counter
    i_node += 1

    if counter > 3:
        Ele = elements.label
        b = (el[(Ele-1):(Ele)])
        elementsInsidePolygon.append(b)

myAssembly.Set(elements=elementsInsidePolygon, name='elementSet1')

#Support nodeset
n2 = myInstance2.nodes
supportNodeList = []
for nodes in n2:
    supportNodeList.append(nodes)

myAssembly.Set(nodes = n2[0:(int(len(supportNodeList)))] , name='NodeSupportSet_all')

#Constraints
region1 = myAssembly.Surface(sidelEdges=myInstance.edges.findAt((((float(Ls)/2), 0,
0.0), )), name='m_Surf-1')
region2 = myAssembly.Surface(sidelEdges=myInstance.edges.findAt((((float(L)-float(Ls)
)/2), 0.0, 0.0), )), name='m_Surf-2')
region3 = myAssembly.Surface(sidelEdges=myInstance2.edges.findAt((((float(Ls)/2), 0,
0.0), )), name='s_Surf-1')
region4 = myAssembly.Surface(sidelEdges=myInstance3.edges.findAt((((float(L)-float(Ls)
)/2), 0.0, 0.0), )), name='s_Surf-2')

myModel.Tie(name='Constraint-Left', master=region1,
    slave=region3, positionToleranceMethod=COMPUTED, adjust=ON,
    tieRotations=ON, thickness=ON)
myModel.Tie(name='Constraint-Right', master=region2,
    slave=region4, positionToleranceMethod=COMPUTED, adjust=ON,
    tieRotations=ON, thickness=ON)

#Create step
if MODE == 1 or MODE == 2:
    myModel.StaticStep(name='applyLoading', previous='Initial')
elif MODE == 3 or MODE == 4:
    myModel.StaticStep(name='applyBasicLoading', previous='Initial')

```

```

myModel.StaticRiksStep(name='applyLoading', previous='applyBasicLoading',
maxNumInc=int(nSTEPS), timeIncrementationMethod=FIXED, initialArcInc=0.01, noStop
=OFF)
#myModel.StaticRiksStep(name='applyLoading', previous='applyBasicLoading',
maxNumInc=int(nSTEPS), initialArcInc=0.05, minArcInc=1e-5, maxArcInc=1e+36)

#Apply loading and BC's
if MODE == 1 or MODE == 2:
    if Q != 0:
        QL = Q*(B/(2*1000))
        myModel.ShellEdgeLoad(name='LineLoad', createStepName='applyLoading', region=
myAssembly-surfaces['topEdge'], magnitude=(QL), distributionType=UNIFORM,
field='', localCsys=None)
        if FW == 1:
            QM = (Q*(B/1000)*(3/2*(B/1000)/(H/1000)))/(12*(3/2*(B/1000)/(H/1000)+1))*
1000
            myModel.ShellEdgeLoad(name='topMoment', createStepName='applyLoading',
region=myAssembly-surfaces['topEdge'], magnitude=(-QM), distributionType=
UNIFORM, field='', localCsys=None, traction=MOMENT)
    if P != 0:
        myModel.Pressure(name='Pressure', createStepName='applyLoading', region=
myAssembly-surfaces['shellFace'], distributionType=UNIFORM, field='',
magnitude=(P), amplitude=UNSET)
    if Gr != 0:
        myModel.Gravity(name='GravityLoad', createStepName='applyLoading', comp2=
float(Gr), distributionType=UNIFORM, field='', region=myInstance.sets[
'shellSet_all'])

if (MODE == 3 or MODE == 4) and OP == 6:
    if P != 0:
        myModel.Pressure(name='Pressure', createStepName='applyBasicLoading', region=
myAssembly-surfaces['shellFace'], distributionType=UNIFORM, field='',
magnitude=(P), amplitude=UNSET)
    if Gr != 0:
        myModel.Gravity(name='GravityLoad', createStepName='applyBasicLoading', comp2
=float(Gr), distributionType=UNIFORM, field='', region=myInstance.sets[
'shellSet_all'])
    if Q != 0:
        if FW == 1:
            QM = (Q*(B/1000)*(3/2*(B/1000)/(H/1000)))/(12*(3/2*(B/1000)/(H/1000)+1))*
1000
            myModel.ShellEdgeLoad(name='topMoment', createStepName=
'applyBasicLoading', region=myAssembly-surfaces['topEdge'], magnitude=(-
QM), distributionType=UNIFORM, field='', localCsys=None, traction=MOMENT)
            QL = Q*(B/(2*1000))
            myModel.ShellEdgeLoad(name='LineLoad', createStepName='applyLoading', region=
myAssembly-surfaces['topEdge'], magnitude=(QL), distributionType=UNIFORM,
field='', localCsys=None)

elif (MODE == 3 or MODE == 4) and OP == 7:
    if Q != 0:
        QL = Q*(B/(2*1000))
        myModel.ShellEdgeLoad(name='LineLoad', createStepName='applyBasicLoading',
region=myAssembly-surfaces['topEdge'], magnitude=(QL), distributionType=
UNIFORM, field='', localCsys=None)
        if FW == 1:
            QM = (Q*(B/1000)*(3/2*(B/1000)/(H/1000)))/(12*(3/2*(B/1000)/(H/1000)+1))*

```

```

1000
myModel.ShellEdgeLoad(name='topMoment', createStepName=
'applyBasicLoading', region=myAssembly-surfaces['topEdge'], magnitude=(-
QM), distributionType=UNIFORM, field='', localCsys=None, traction=MOMENT)
if Gr != 0:
myModel.Gravity(name='GravityLoad', createStepName='applyBasicLoading', comp2
=float(Gr), distributionType=UNIFORM, field='', region=myInstance.sets[
'shellSet_all'])
if P != 0:
myModel.Pressure(name='Pressure', createStepName='applyLoading', region=
myAssembly-surfaces['shellFace'], distributionType=UNIFORM, field='',
magnitude=(P), amplitude=UNSET)

elif (MODE == 3 or MODE == 4) and (OP != 6 or OP != 7):
if P != 0:
myModel.Pressure(name='Pressure', createStepName='applyBasicLoading', region=
myAssembly-surfaces['shellFace'], distributionType=UNIFORM, field='',
magnitude=(P), amplitude=UNSET)
if Gr != 0:
myModel.Gravity(name='GravityLoad', createStepName='applyBasicLoading', comp2
=float(Gr), distributionType=UNIFORM, field='', region=myInstance.sets[
'shellSet_all'])
if Q != 0:
if FW == 1:
QM = (Q*(B/1000)*(3/2*(B/1000)/(H/1000)))/(12*(3/2*(B/1000)/(H/1000)+1))*
1000
myModel.ShellEdgeLoad(name='topMoment', createStepName=
'applyBasicLoading', region=myAssembly-surfaces['topEdge'], magnitude=(-
QM), distributionType=UNIFORM, field='', localCsys=None, traction=MOMENT)
QL = Q*(B/(2*1000))
myModel.ShellEdgeLoad(name='LineLoad', createStepName='applyLoading', region=
myAssembly-surfaces['topEdge'], magnitude=(QL), distributionType=UNIFORM,
field='', localCsys=None)

if SUP == 0:
myModel.DisplacementBC(name='Roller_Z_left', createStepName='Initial', region=
myAssembly.sets['leftSupportZ'], u1=UNSET, u2=UNSET, u3=SET, url=UNSET, ur2=UNSET
, ur3=UNSET, amplitude=UNSET, distributionType=UNIFORM, fieldName='',
localCsys=None)
myModel.DisplacementBC(name='Roller_Z_right', createStepName='Initial', region=
myAssembly.sets['rightSupportZ'], u1=UNSET, u2=UNSET, u3=SET, url=UNSET, ur2=
UNSET, ur3=UNSET, amplitude=UNSET, distributionType=UNIFORM, fieldName='',
localCsys=None)

elif SUP == 1:
myModel.DisplacementBC(name='Roller_Z_top', createStepName='Initial', region=
myAssembly.sets['topSupportZ'], u1=UNSET, u2=UNSET, u3=SET, url=UNSET, ur2=UNSET,
ur3=UNSET, amplitude=UNSET, distributionType=UNIFORM, fieldName='', localCsys=
None)
myModel.DisplacementBC(name='Roller_Z_bottom', createStepName='Initial', region=
myAssembly.sets['bottomSupportZ'], u1=UNSET, u2=UNSET, u3=SET, url=UNSET, ur2=
UNSET, ur3=UNSET, amplitude=UNSET, distributionType=UNIFORM, fieldName='',
localCsys=None)

myModel.DisplacementBC(name='Roller_X_left', createStepName='Initial', region=
myAssembly.sets['leftSupportX'], u1=UNSET, u2=SET, ur3=UNSET, amplitude=UNSET,
distributionType=UNIFORM, fieldName='', localCsys=None)

```

```

myModel.DisplacementBC(name='Roller_X_right', createStepName='Initial', region=
myAssembly.sets['rightSupportX'], u1=UNSET, u2=SET, ur3=UNSET, amplitude=UNSET,
distributionType=UNIFORM, fieldName='', localCsys=None)
myModel.DisplacementBC(name='Roller_Y_left', createStepName='Initial', region=
myAssembly.sets['leftSupportY'], u1=SET, u2=UNSET, ur3=UNSET, amplitude=UNSET,
distributionType=UNIFORM, fieldName='', localCsys=None)

#Request field output
myModel.FieldOutputRequest(name='F-Output-1', createStepName='applyLoading',
variables=('S', 'PE', 'PEEQ', 'PEMAG', 'LE', 'U', 'RF', 'CF'))
if MODE == 3 or MODE == 4:
    myModel.FieldOutputRequest(name='F-Output-1', createStepName='applyBasicLoading',
        variables=('S', 'PE', 'PEEQ', 'PEMAG', 'LE', 'U', 'RF', 'CF'))

#Create and submit Job
if MODE == 1 or MODE == 2:
    myJob=mdb.Job(name='wall_linearElastic'+str(incr)+'_'+str(k), model=
        'wall_linearElastic'+str(incr)+'_'+str(k), description='', type=ANALYSIS, atTime=
        None, waitMinutes=0, waitHours=0, queue=None, memory=90, memoryUnits=PERCENTAGE,
        getMemoryFromAnalysis=True, explicitPrecision=SINGLE, nodalOutputPrecision=SINGLE
        , echoPrint=OFF, modelPrint=OFF, contactPrint=OFF, historyPrint=OFF,
        userSubroutine='', scratch='', multiprocessingMode=DEFAULT, numCpus=1, numGPUs=0)

elif MODE == 3 or MODE == 4:
    myJob=mdb.Job(name='wall_smearedCrack'+str(incr)+'_'+str(k), model=
        'wall_smearedCrack'+str(incr)+'_'+str(k), description='', type=ANALYSIS, atTime=
        None, waitMinutes=0, waitHours=0, queue=None, memory=90, memoryUnits=PERCENTAGE,
        getMemoryFromAnalysis=True, explicitPrecision=SINGLE, nodalOutputPrecision=SINGLE
        , echoPrint=OFF, modelPrint=OFF, contactPrint=OFF, historyPrint=OFF,
        userSubroutine='', scratch='', multiprocessingMode=DEFAULT, numCpus=1, numGPUs=0)

myJob.submit(consistencyChecking=OFF)
myJob.waitForCompletion()

# Open the odb
if MODE == 1 or MODE == 2:
    myODBname='C:\SIMULIA\Abaqus\Commands\wall_linearElastic'+str(incr)+'_'+str(k)
    #myODBname='C:\Temp\shell_linearElastic'+str(incr)+'_'+str(k)
elif MODE == 3 or MODE == 4:
    myODBname='C:\SIMULIA\Abaqus\Commands\wall_smearedCrack'+str(incr)+'_'+str(k)
    #myODBname='C:\Temp\wall_smearedCrack'+str(incr)+'_'+str(k)
myOdb = session.openOdb(name=myODBname+'.odb')

if len(myOdb.steps['applyLoading'].frames) != 0:
    #Max U2 deformation of elastic support
    dispNodes = myOdb.rootAssembly.nodeSets['NODESUPPORTSET_ALL']
    dispFrame = myOdb.steps['applyLoading'].frames[-1]
    dispField = dispFrame.fieldOutputs['U']
    disp_at_Nodes = dispField.getSubset(region=dispNodes)

    max_U2_sup = 0.
    for dispVal in disp_at_Nodes.values:
        #dispVal[0] is U1, [1] is U2, [2] is U3
        if dispVal.data[1] < max_U2_sup:
            max_U2_sup = dispVal.data[1]

#Process displacement

```

```

dispNodes = myOdb.rootAssembly.nodeSets['NODESET1']
dispFrame = myOdb.steps['applyLoading'].frames[-1]
dispField = dispFrame.fieldOutputs['U']
disp_at_Nodes = dispField.getSubset(region=dispNodes)

max_U2 = 0. #Initiate a variable
max_U3 = 0. #Initiate a variable

#Write U2 displacement
for dispVal in disp_at_Nodes.values:
    realdispVal = abs(dispVal.data[1]-max_U2_sup)
    #dispVal[0] is U1, [1] is U2, [2] is U3
    if realdispVal > max_U2:
        max_U2 = realdispVal
        label = dispVal.nodeLabel

if MODE == 1 or MODE == 2:
    outputFile = open('LE_SA_displacementY.txt','a+')
elif MODE == 3 or MODE == 4:
    outputFile = open('SC_SA_displacementY.txt','a+')
outputFile.write(str(k)+'\n')
outputFile.write('Node '+str(label)+' ')
outputFile.write(str(max_U2)+'\n')
outputFile.close()

outputFile = open('displacementY.txt','a+')
outputFile.write(str(max_U2)+'\n')
outputFile.close()

#Write U3 displacement
for dispVal in disp_at_Nodes.values:
    if abs(dispVal.data[2]) > max_U3:
        max_U3 = abs(dispVal.data[2])
        label = dispVal.nodeLabel

if MODE == 1 or MODE == 2:
    outputFile = open('LE_SA_displacementZ.txt','a+')
elif MODE == 3 or MODE == 4:
    outputFile = open('SC_SA_displacementZ.txt','a+')
outputFile.write(str(k)+'\n')
outputFile.write('Node '+str(label)+' ')
outputFile.write(str(max_U3)+'\n')
outputFile.close()

outputFile = open('displacementZ.txt','a+')
outputFile.write(str(max_U3)+'\n')
outputFile.close()

#Process stresses
stressElements = myOdb.rootAssembly.elementSets['ELEMENTSET1']
stressFrame = myOdb.steps['applyLoading'].frames[-1]
stressField = stressFrame.fieldOutputs['S']
stress_at_Elements = stressField.getSubset(region=stressElements)

#Write S11 Compression
max_S11_c = 0.
for stressVal in stress_at_Elements.values:

```

```

        stressDataList = []
        elementLabelList = []
        stressDataList.append(stressVal.data[0])
        elementLabelList.append(stressVal.elementLabel)
        if min(stressDataList) < max_S11_c:
            max_S11_c = min(stressDataList)
            eleLabel = elementLabelList[-1]
    if MODE == 1 or MODE == 2:
        outputFile = open('LE_SA_S11c.txt','a+')
    elif MODE == 3 or MODE == 4:
        outputFile = open('SC_SA_S11c.txt','a+')
    outputFile.write(str(k)+'\n')
    outputFile.write('Element '+str(eleLabel)+' ')
    outputFile.write(str(max_S11_c)+'\n')
    outputFile.close()

outputFile = open('S11c.txt','a+')
outputFile.write(str(max_S11_c)+'\n')
outputFile.close()

#Write S22 Compression
max_S22_c = 0.
for stressVal in stress_at_Elements.values:
    stressDataList = []
    elementLabelList = []
    stressDataList.append(stressVal.data[1])
    elementLabelList.append(stressVal.elementLabel)
    if min(stressDataList) < max_S22_c:
        max_S22_c = min(stressDataList)
        eleLabel = elementLabelList[-1]
    if MODE == 1 or MODE == 2:
        outputFile = open('LE_SA_S22c.txt','a+')
    elif MODE == 3 or MODE == 4:
        outputFile = open('SC_SA_S22c.txt','a+')
    outputFile.write(str(k)+'\n')
    outputFile.write('Element '+str(eleLabel)+' ')
    outputFile.write(str(max_S22_c)+'\n')
    outputFile.close()

outputFile = open('S22c.txt','a+')
outputFile.write(str(max_S22_c)+'\n')
outputFile.close()

#Write S11 Tension
max_S11_t = 0.
for stressVal in stress_at_Elements.values:
    stressDataList = []
    elementLabelList = []
    stressDataList.append(stressVal.data[0])
    elementLabelList.append(stressVal.elementLabel)
    if max(stressDataList) > max_S11_t:
        max_S11_t = max(stressDataList)
        eleLabel = elementLabelList[-1]
    if MODE == 1 or MODE == 2:
        outputFile = open('LE_SA_S11t.txt','a+')
    elif MODE == 3 or MODE == 4:
        outputFile = open('SC_SA_S11t.txt','a+')

```

```

outputFile.write(str(k)+'\n')
outputFile.write('Element '+str(eleLabel)+' ')
outputFile.write(str(max_S11_t)+'\n')
outputFile.close()

outputFile = open('S11t.txt','a+')
outputFile.write(str(max_S11_t)+'\n')
outputFile.close()

#Write S22 Tension
max_S22_t = 0.
for stressVal in stress_at_Elements.values:
    stressDataList = []
    elementLabelList = []
    stressDataList.append(stressVal.data[1])
    elementLabelList.append(stressVal.elementLabel)
    if max(stressDataList) > max_S22_t:
        max_S22_t = max(stressDataList)
        eleLabel = elementLabelList[-1]
if MODE == 1 or MODE == 2:
    outputFile = open('LE_SA_S22t.txt','a+')
elif MODE == 3 or MODE == 4:
    outputFile = open('SC_SA_S22t.txt','a+')
outputFile.write(str(k)+'\n')
outputFile.write('Element '+str(eleLabel)+' ')
outputFile.write(str(max_S22_t)+'\n')
outputFile.close()

outputFile = open('S22t.txt','a+')
outputFile.write(str(max_S22_t)+'\n')
outputFile.close()

#Process reaction forces X
reactionNodesX1 = myOdb.rootAssembly.nodeSets['LEFTSUPPORTY']
reactionFrameX1 = myOdb.steps['applyLoading'].frames[-1]
reactionFieldX1 = reactionFrameX1.fieldOutputs['RF']
reaction_forcesX1 = reactionFieldX1.getSubset(region=reactionNodesX1)
rForcesX = []

for reactionVal in reaction_forcesX1.values:
    rForcesX.append(reactionVal.data[0])

if abs(sum(rForcesX)/1000)>= 0.1:
    xLoad = open('xLoad.txt', 'a+')
    xLoad.write(str(sum(rForcesX)/1000)+'\n')
    xLoad.close()
else:
    xLoad = open('xLoad.txt', 'a+')
    xLoad.write('0'\n')
    xLoad.close()

#Process reaction forces Y
reactionNodesY1 = myOdb.rootAssembly.nodeSets['LEFTSUPPORTX']
reactionFrameY1 = myOdb.steps['applyLoading'].frames[-1]
reactionFieldY1 = reactionFrameY1.fieldOutputs['RF']
reaction_forcesY1 = reactionFieldY1.getSubset(region=reactionNodesY1)
rForcesY = []

```

```

for reactionVal in reaction_forcesY1.values:
    rForcesY.append(reactionVal.data[1])

reactionNodesY2 = myOdb.rootAssembly.nodeSets['RIGHTSUPPORTX']
reactionFrameY2 = myOdb.steps['applyLoading'].frames[-1]
reactionFieldY2 = reactionFrameY2.fieldOutputs['RF']
reaction_forcesY2 = reactionFieldY2.getSubset(region=reactionNodesY2)

for reactionVal in reaction_forcesY2.values:
    rForcesY.append(reactionVal.data[1])

if abs(sum(rForcesY)/1000)>= 0.1:
    if Gr == 0:
        L_LOAD = L/1000 - 2*(Ls/1000)
        LOAD = round((sum(rForcesY)/1000)/L_LOAD, 3)

    elif Gr != 0:
        if MODE == 1 or MODE == 3:
            T = tf1
        elif MODE == 2 or MODE == 4:
            T = tf1+tf2
        G_LOAD = H/1000 * L/1000 * T/1000 * D * abs(Gr)/1000
        L_LOAD = L/1000 - 2*(Ls/1000)
        LOAD = round(((sum(rForcesY)/1000) - G_LOAD)/L_LOAD , 3)

    yLoad = open('yLoad.txt', 'a+')
    yLoad.write(str(LOAD)+'\n')
    yLoad.close()
else:
    yLoad = open('yLoad.txt', 'a+')
    yLoad.write('0'+'\n')
    yLoad.close()

#Process reaction forces Z
if SUP == 0:
    reactionNodesZ1 = myOdb.rootAssembly.nodeSets['LEFTSUPPORTZ']
elif SUP == 1:
    reactionNodesZ1 = myOdb.rootAssembly.nodeSets['TOPSUPPORTZ']

reactionFrameZ1 = myOdb.steps['applyLoading'].frames[-1]
reactionFieldZ1 = reactionFrameZ1.fieldOutputs['RF']
reaction_forcesZ1 = reactionFieldZ1.getSubset(region=reactionNodesZ1)
rForcesZ = []

for reactionVal in reaction_forcesZ1.values:
    rForcesZ.append(reactionVal.data[2])
if SUP == 0:
    reactionNodesZ2 = myOdb.rootAssembly.nodeSets['RIGHTSUPPORTZ']
elif SUP == 1:
    reactionNodesZ2 = myOdb.rootAssembly.nodeSets['BOTTOMSUPPORTZ']

reactionFrameZ2 = myOdb.steps['applyLoading'].frames[-1]
reactionFieldZ2 = reactionFrameZ2.fieldOutputs['RF']
reaction_forcesZ2 = reactionFieldZ2.getSubset(region=reactionNodesZ2)

for reactionVal in reaction_forcesZ2.values:

```

```

rForcesZ.append(reactionVal.data[2])

if abs(sum(rForcesZ)/1000)>=0.1:
    A_LOAD = L/1000*H/1000
    LOAD = round((sum(rForcesZ)/1000)/A_LOAD, 3)
    zLoad = open('zLoad.txt', 'a+')
    zLoad.write(str(LOAD)+'\n')
    zLoad.close()
else:
    zLoad = open('zLoad.txt', 'a+')
    zLoad.write('0'+'\n')
    zLoad.close()

#Evaluate job
print eval(A)

if GOAL == 0:
    result = max_S11_t
    targ = T_targ1
elif GOAL == 1:
    result = max_S22_t
    targ = T_targ2
elif GOAL == 2:
    result = max_S22_c
    targ = C_targ2
elif GOAL == 3:
    result = max_S11_c
    targ = C_targ1
elif GOAL == 4:
    result = max_U2
    targ = D_targY
elif GOAL == 5:
    result = max_U3
    targ = D_targZ

result_list.append(float(result))

if MODE == 1 or MODE == 2:
    failuretxt = open('LE_SA_failureCheck.txt', 'a+')
    outputFile = open('LE_SA_PostProcessing.txt', 'a+')
elif MODE == 3 or MODE == 4:
    failuretxt = open('SC_SA_failureCheck.txt', 'a+')
    outputFile = open('SC_SA_PostProcessing.txt', 'a+')

if abs(eval(A)[0]) < BEST and False not in eval(A)[1:len(eval(A))]:
    BEST = abs(eval(A)[0])
    eval_list.append(eval(A)[0])
    A_best_list.append(A)
    failuretxt.write('NO FAILURE, BETTER VALUE ACCEPTED'+'\n')
    failuretxt.close()
    if abs(abs(eval(A)[0])/(targ))*100 <= ST:
        print str(ST)+'% Tolerance achieved'
        outputFile.write(str(ST)+'% Tolerance achieved'+'\n')
        outputFile.close()
        k += 1
        break

```

```

elif abs(eval(A)[0]) < BEST and False in eval(A)[1:len(eval(A))]:
    BEST = abs(eval(A)[0])
    eval_list.append(eval(A)[0])
    A_best_list.append(A)
    failuretxt.write(str(failure_check(A))+ ' BETTER VALUE ACCEPTED'+'\n')
    failuretxt.close()
    if abs(abs(eval(A)[0])/(targ))*100 <= ST:
        print str(ST)+'% Tolerance achieved'
        outputFile.write(str(ST)+'% Tolerance achieved'+'\n')
        outputFile.close()
        k += 1
        break

elif abs(eval(A)[0]) > BEST and False not in eval(A)[1:len(eval(A))]:
    delta = abs(abs(eval(A)[0]/BEST)-1)
    p = math.exp((-delta * 20 / T))
    if random.random() < p:
        BEST = abs(eval(A)[0])
        eval_list.append(eval(A)[0])
        A_best_list.append(A)
        A_worse_list.append(A)
        failuretxt.write('NO FAILURE, WORSE VALUE ACCEPTED'+'\n')
        failuretxt.close()
    else:
        failuretxt.write('NO FAILURE, WORSE VALUE REJECTED'+'\n')
        failuretxt.close()
    if abs(abs(eval(A)[0])/(targ))*100 <= ST:
        print str(ST)+'% Tolerance achieved'
        outputFile.write(str(ST)+'% Tolerance achieved'+'\n')
        outputFile.close()
        k += 1
        break

elif abs(eval(A)[0]) > BEST and False in eval(A)[1:len(eval(A))]:
    delta = abs(abs(eval(A)[0]/BEST)-1)
    p = math.exp((-delta * 20 / T))
    if random.random() < p:
        BEST = abs(eval(A)[0])
        eval_list.append(eval(A)[0])
        A_best_list.append(A)
        A_worse_list.append(A)
    failuretxt.write(str(failure_check(A))+'\n')
    failuretxt.close()
    if abs(abs(eval(A)[0])/(targ))*100 <= ST:
        print str(ST)+'% Tolerance achieved'
        outputFile.write(str(ST)+'% Tolerance achieved'+'\n')
        outputFile.close()
        k += 1
        break

elif abs(eval(A)[0]) == BEST:
    print eval(A)
    failuretxt.close()
    if k > I/(SPLIT):
        similar_list.append(abs(eval(A)[0]))
    if len(similar_list) == 3:
        outputFile.write('ABORTED - 3 Times same eval found'+'\n')

```

```

        outputFile.close()
        k += 1
        break

    else:
        eval_list.append(999)
        if MODE == 1 or MODE == 2:
            failuretxt = open('LE_SA_failureCheck.txt', 'a+')
        elif MODE == 3 or MODE == 4:
            failuretxt = open('SC_SA_failureCheck.txt', 'a+')
        failuretxt.write('NO OPT. POSSIBLE - ITERATION SKIPPED'+'\n')
        failuretxt.close()

    T = update_temperature(T,k)
    T_list.append(T)

    myOdb.close()

    k += 1

best_eval = float(min(eval_list, key=lambda x:abs(abs(x)-0)))
best_result = float(min(result_list, key=lambda x:abs(abs(x)-targ)))
bestEvalIndex1 = eval_list.index(best_eval)
bestA1 = A_best_list[bestEvalIndex1]

if MODE == 1 or MODE == 2:
    outputFile = open('LE_SA_PostProcessing.txt', 'a+')
elif MODE == 3 or MODE == 4:
    outputFile = open('SC_SA_PostProcessing.txt', 'a+')
outputFile.write("Iterations: "+str(k-1)+'\n')
outputFile.write("Target: "+str(targ)+'\n')
outputFile.write("Result: "+str(best_result)+'\n')
outputFile.write("Error: "+str(abs(best_eval)/targ*100)+"%\n")
outputFile.write("Parameter: "+str(bestA1 * parameter/1000)+'\n')
outputFile.write("Best A value: "+str(bestA1)+'\n')
outputFile.write("All random values: "+str(A_all_list)+'\n')
outputFile.write("All best A values: "+str(A_best_list)+'\n')
outputFile.write("Worse values accepted: "+str(A_worse_list)+'\n')
outputFile.write("#####"+'\n')
outputFile.close()

graphdata = open('graphdata.txt', 'a+')
graphdata.write(str(bestA1 * parameter/1000)+'\n')
graphdata.close()

print "Iterations:", k-1
print "Target:", targ
print "Result:", best_result
print "Error:", (abs(best_eval)/targ*100), "%"
print "Parameter:", (bestA1/1000)
print "Best A value:", bestA1
print "All random values:", A_all_list
print "All best A values:", A_best_list
print "All evaluated values:", eval_list
print "Worse values accepted:", A_worse_list
incr += 1
os._exit(0)

```
