An analysis of crack progression using micro-structures

Isa Ritfeld

Civil Engineering

Structural Engineering

Technical University of Delft

Submitted in partial satisfaction of the requirements for the
Degree of Bachelor Student
in Civil Engineering

Supervisor H. Alkisaei MSc
Second Supervisor Dr. Ir. P.C.J. Hoogenboom

June, 2021

Abstract

The failure of structures is a concern in engineering. Mathematical formulas are required to determine the failure mechanisms of the structures, however, the pattern of the cracks cannot be found by this method. Therefore, numerical models are of great interest. In this report, using the computer program CALFEM, the programming language python and basic mechanics, models are created consisting of rods. The purpose of this research is to determine the capabilities and limitations of lattice models for describing crack development in unreinforced concrete. The experiment that the models are based on, is a single-edge notched beam under four point loading. In the models, the height and length consist of small repeating elements. It determines the amount of rods in the system. The rods in the models contain the characteristics of plain concrete. The effects of the three-dimensional beam are included in the properties of the rods. The script written shows the crack progression of the models. With the models made, limits can be found that affect the correspondence of the models with the experiment. The limits are the structures given, the variables used and the properties of the rods. The models can be improved by using rods of different orientations, evenly distributing the area of the beam, and adjusting the variables such that the cracks created not appear angular. With this research, a foundation has been laid for using modeling to create structures in the future.

List of Tables

2.1 Characteristics of plain concrete [4]. 5

List of Figures

1.1 Lattice-structure built on Renishaw AM250 metal AM system [1] 2

2.1 The set-up of the experiment 3
2.2 Side view of the crack pattern 4
2.3 3D view of the crack growth in FEM 4
2.4 Model with variables 7
2.5 A visual of the border of the models made by the script of Python 7
2.6 Visualisation of the combination structures 8
2.7 A rod model with nh = 3 and nl = 12 8
2.8 A rod model with nh = 6 and nl = 24 8
2.9 The distribution of area in 2D per model per repeating element 9
2.10 The area that a internal rod represents in different models 9

3.1 The crack development of the model with horizontal rods 12
3.2 The crack development of the model with horizontal and vertical rods . . . 13
3.3 The crack development of the model with diagonals 14
3.4 The crack development of the combination model with diagonals to the left 15
3.5 The crack development of the combination model with diagonals to the right 16
3.6 The crack development of the combination model with diagonals in both . 16

A.1 The normal stress of a prismatic bar [3] 21

C.1 Model with horizontal rods 37
C.2 Model with horizontal and vertical rods 37
C.3 Model with diagonals 38
C.4 Combination model with diagonals to the left 38
C.5 Combination model with diagonals to the right 38
C.6 Combination model with diagonals in both directions 38

D.1 The variable analysis of the model with horizontal rods 40
D.2 The variable analysis of the model with horizontal and vertical rods 42
D.3 The variable analysis of the model with diagonals 44
D.4 The variable analysis of a combination model 46
D.5 The variable analysis of a combination model 48
D.6 The variable analysis of a combination model 50

Table of Contents

1 Introduction **1**

1.1 Problem . . . 1

1.2 Objective . . . 1

1.3 Approach . . . 2

2 The model **3**

2.1 The experiment . . . 3

2.1.1 Model requirements . . . 5

2.2 The models . . . 5

2.2.1 Variables of the models . . . 6

2.2.2 2D visualizations of the models in Python . . . 7

3 Results **11**

3.1 Approach . . . 11

3.2 Horizontal rods . . . 12

3.3 Horizontal and vertical rods . . . 12

3.4 Diagonals . . . 13

3.5 Combination models . . . 14

3.5.1 Combination with diagonals to the left . . . 14

3.5.2 Combination with diagonals to the right . . . 15

3.5.3 Combination with diagonals in both directions . . . 15

4 Discussion **17**

5 Conclusion **19**

A Basic Mechanics **21**

B Python code **23**

B.1 The rod model . . . 24

B.2 Stress models . . . 31

C Structure **37**

D Variables analysis **39**

D.1 Model with horizontal rods . . . 39

D.2 Model with horizontal and vertical rods 41
D.3 Model with diagonals 43
D.3.1 Combination with diagonals to the left 45
D.3.2 Combination with diagonals to the right 47
D.3.3 Combination with diagonals in both directions 49

Chapter 1

Introduction

In this chapter, an introduction is given on the subject, describing the problem, purpose and approach of this research. The tools used to make the models are explained.

1.1 Problem

Preventing the failure of structures is a concern in engineering. To understand how a failure mechanism is obtained, it is necessary to look at the properties of the material. It involves progressive micro-cracking, debounding or other processes of internal damages [2]. With mathematical formulas, the cracks sizes can be determined.

The problem, however, is that the pattern of the cracks cannot be found by this method. The formulas used to determine the size are simplified: they are mathematical approximations of the reality. Laboratory work is required to determine the progression of the cracks. Therefore, numerical models are of great interest. In this report, a numerical analysis of plain concrete using micro-structures is presented and discussed.

1.2 Objective

The purpose of this research is to determine the capabilities and limitations of lattice models for describing crack development in unreinforced concrete. The kind of model used consists of small rods, which contain the properties of the material under research. A example of a lattice structure is shown in figure 1.1. For each rod, the acting forces are calculated, which is used to determine the stresses. Due to the grid structure, certain variables must be taken into account, such as the angles of the joints, the construction itself and the size of the structure.

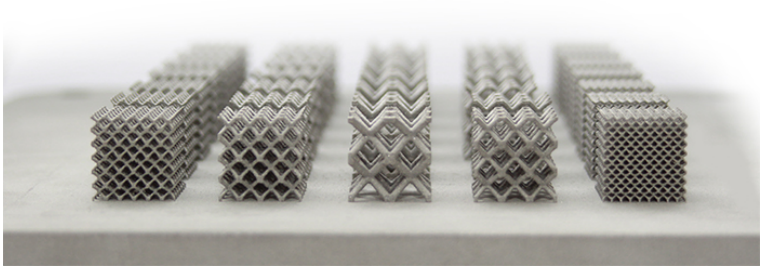

Figure 1.1: Lattice-structure built on Renishaw AM250 metal AM system [1]

By using different structures and modifying the properties of the rods, the effects of the variables and the limits of the models can be determined. The following questions can then be answered:

- Is it possible to describe crack development in materials by means of modelling?
- How comparable are these models to reality?
- What influence does the structure have on the model?
- What influence do the properties of the material have on the model?
- What are the limitations of the models made?

Before explaining the experiment and the models, it is necessary to explain what tools are used to create the models.

1.3 Approach

To create models and obtain results, the computer program CALFEM is used. CALFEM is an abbreviation for 'Computer Aided Learning of the Finite Element Method'. With the finite element method, it is possible to make a grid structure containing rods. In addition, the package contains functions that are used to calculate the forces per rod. CALFEM is a package that is imported into Python, a programming language that allows calculations and plotting.

Basic mechanics are applied to determine the stresses in the rods. The basic mechanics used is explained in Appendix A. With the purpose know and the tools clarified, the models can be made. To obtain the models, the real experiment needs to be explained. It gives the requirements and preconditions of the models. This is done in the next chapter.

Chapter 2

The model

In the models, the beam is characterized by rods of various orientations. The rods may be imagined to represent the material in micro-structure. In this chapter, the real experiment is shown. The models created and used to obtain the results are presented and discussed.

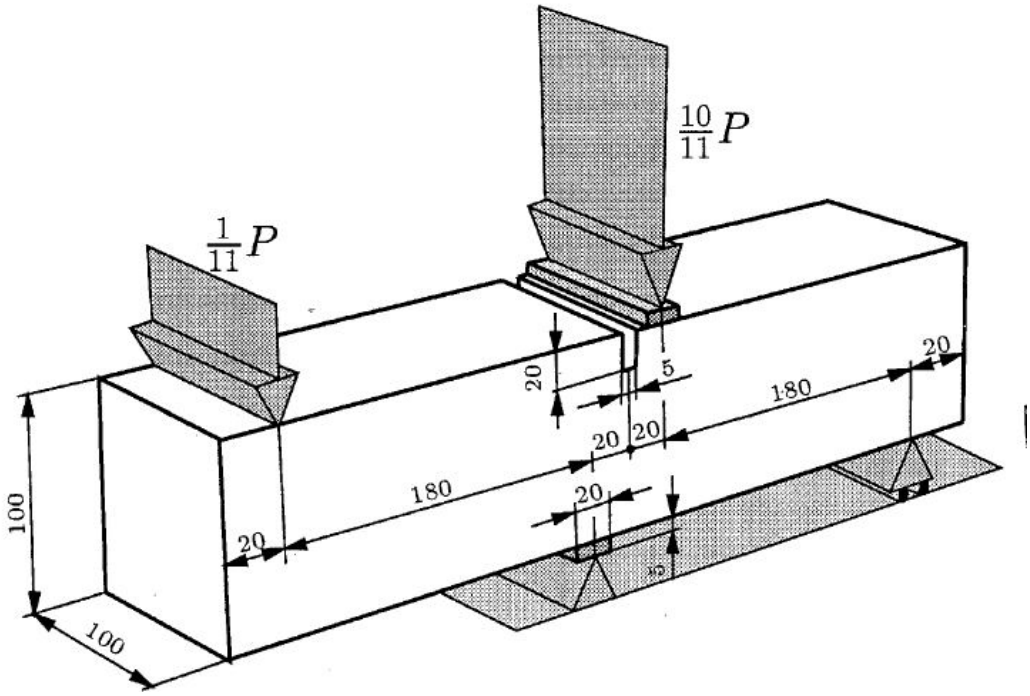

Figure 2.1: The set-up of the experiment

2.1 The experiment

The experiment that the models are based on, is a single-edge notched beam under four point loading [2]. The dimensions are shown in millimeter in figure 2.1. In the beam, there is a notch: a created weak spot in the specimen. As forces are gradually placed on the system, a crack begins to appear in the beam. Due to the notch, the crack starts at the top. The course of the crack is visible in figure 2.2.

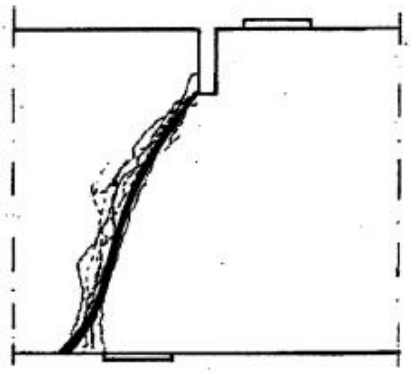

Figure 2.2: Side view of the crack pattern

Next to the real experiment, a Finite Element Analysis (FEM) of the experiment was done [2]. The model uses small planes to indicate the material under research. In figure 2.3 the FEM gives the progression of the crack in 3D.

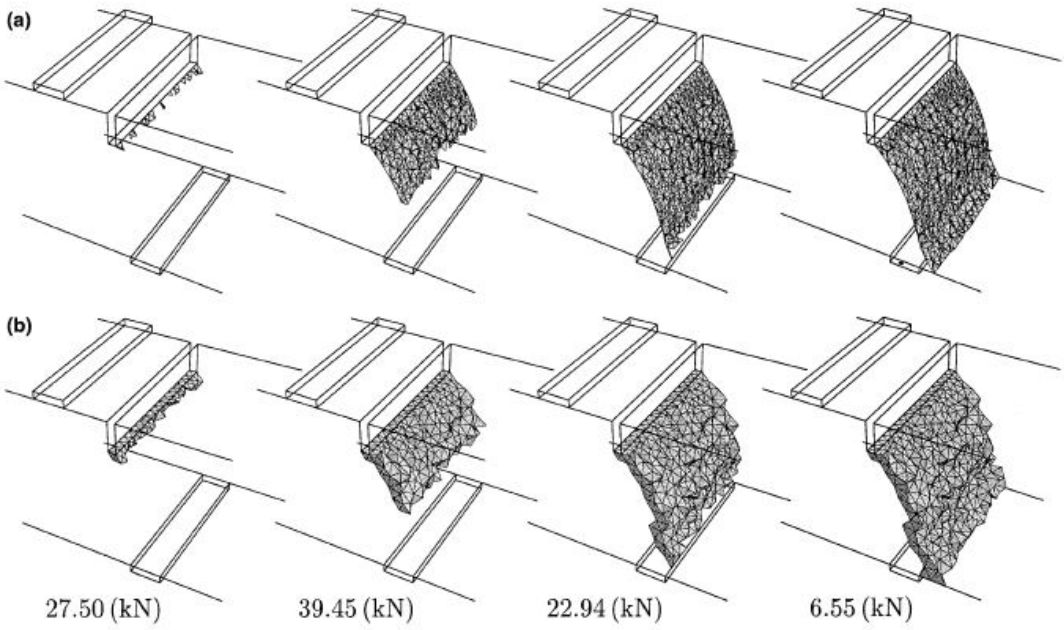

Fig. 15. (a) Crack-growth and (b) evolution of the partition of unity finite elements for different levels of loads.

Figure 2.3: 3D view of the crack growth in FEM

2.1.1 Model requirements

With the information from the experiment the requirements and conditions can be determined. The structure of the models needs to have the same dimensions as the real beam. The loads and the boundary conditions need to be called on the same places as the real experiment. If that is not feasible, it should be as close as possible to the original place.

In addition, the rods characteristics must be modified in order to represent the material. In the original test, plain concrete was used. The characteristics of interest are listed in table 2.1 below. The tensile strength is the resistance that must be overcome to break the cohesion of the material. When the strength is reached, cracks start to occur. Were the strength is reach the stiffness, the resistance to deformation, has become zero. So whenever the rods reach the tensile strength, the stiffness have to be reduce to zero.

Table 2.1: Characteristics of plain concrete [4].

Characteristics	Unit	Value
Moment of inertia	kg*m2	1/12*b*h
Elasticity modulus	N/mm2	24800
Tensile strength	N/mm2	2.80

The effects of the three-dimensional beam need to be included in the properties of the rods. Because the rods differ from orientation and position, these properties can variate within the system. This depends on how the cross sections of the rods are taken and what the rods represent in the system. Between the models the properties differ, since the type of rods called differ in each model. With the requirements, the models can be made.

2.2 The models

For comparison purposes, six models were made with different structures. The structures consist of rods in different directions. There are variations, but also similarities between the models.

2.2.1 Variables of the models

One of these similarities is the variables used. The variables are based on the dimensions and data from the real experiment. Some can be adjusted, other variables represent dimensions and should not be changed. The variables are listed and elaborated below.

To avoid errors, all distances are given in meters. The forces acting on the systems are given in kilograms of Newtons. Since the material properties are given in other units, it should be remembered that the magnitude of the values must be adjusted for proper use.

The variables in the model are:

- P: the external forces working on the beam
- l: the length of the beam
- h: the height of the beam
- b: the width of the beam
- t: the thickness of the rods
- dl: small repeatable lengths
- dh: small repeatable heights
- nl: number of the repeatable lengths
- nh: number of the repeatable heights

In figure 2.4 the third dimension, the width of the beam, can be seen. The rods in 2D operate across the entire width of the beam, which affects the area of the cross-section of the rods. The coordinate system is located at the bottom left of the beam. The height and length of the beam both consist of small repeating elements. Since the structure differs from model to model, figure 2.4 represents one of the models made. With the variables, the models can be created in Python.

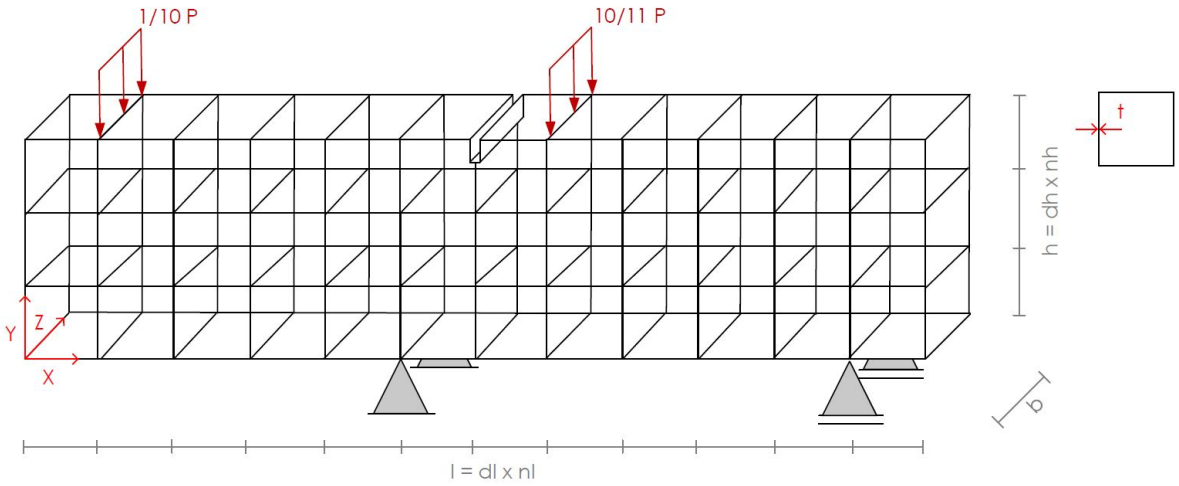

Figure 2.4: Model with variables

2.2.2 2D visualizations of the models in Python

The script allows the models to be rendered in Python. It is not necessary to manually perform the calculations. It is advised to use the script when looking at the results of the models. The script can be found in Appendix B. It gives a step by step explanation how the models are made.

The rod models

The models created are two-dimensional. Each model consist of two parts: the rod model and the tension model. The rod models make the structures of the models. Each rod model contains a border, indicated in figure 2.5. The notch is a fixed place in the system that need to be supported by the structure chosen.

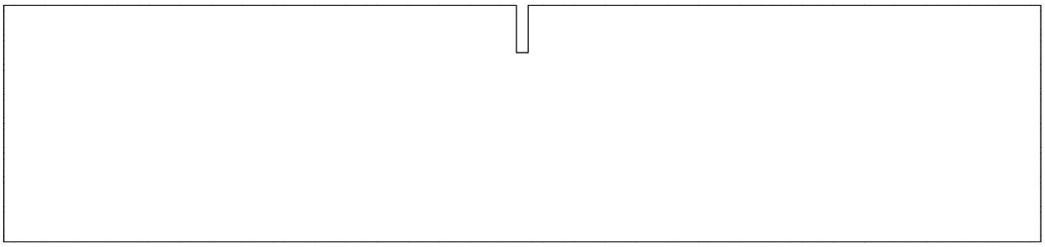

Figure 2.5: A visual of the border of the models made by the script of Python

Inside the border, different kind of rods are called, which are horizontal rods, vertical rods, and diagonals. Different structures are made by combining different rods. In the combinations in the figure 2.6 the horizontal rods, vertical rods and diagonals are combined in one system. The rod models are given in Appendix C.

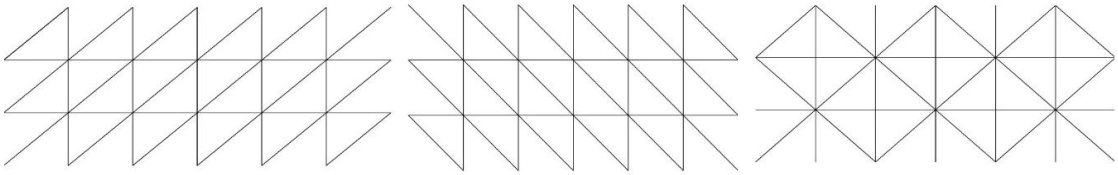

Figure 2.6: Visualisation of the combination structures

The amount of rods depends on the number of repeating elements. The amount of repeating elements can be changed in height and in length. With changing the structure, the characteristics of the rods change. The script paints a picture of the situation once the number of repeating elements is specified. In the figures 2.7 and 2.8 the difference is shown in the model with vertical and horizontal rods.

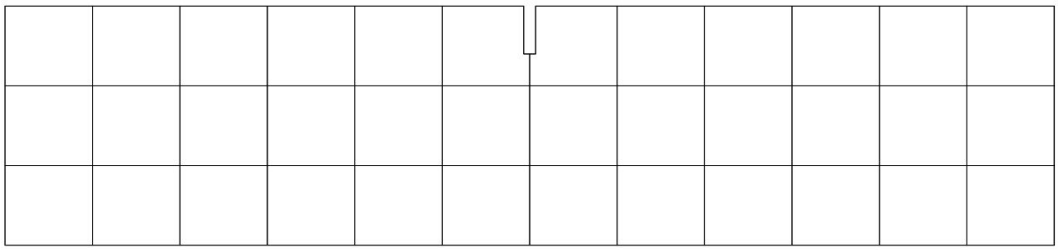

Figure 2.7: A rod model with nh = 3 and nl = 12

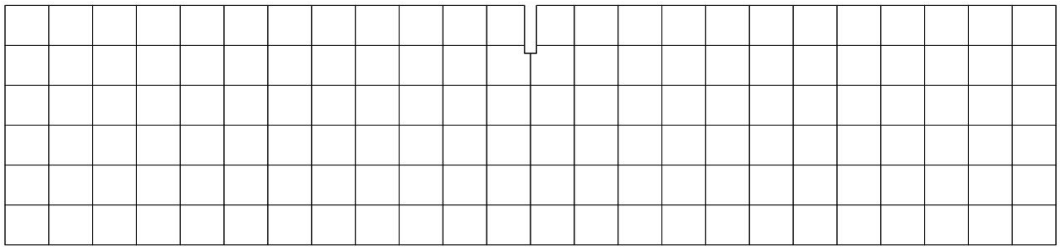

Figure 2.8: A rod model with nh = 6 and nl = 24

The stress models

With the rod models made, a foundation is laid for the stress models. In the experiment the beam is gradually loaded by a force P resulting in a crack in the middle of the beam. To replicate this in the models, the force is loaded step by step, with a stress check after each step. The steps taken are per kilo of Newton. With the functions from CALFEM the forces acting on the rods can be calculated in each step. With the forces and the formula A.1 the stresses can be calculated per rod.

To determine the stresses, the properties of the rods must be determined. The rods contain the same Young Modulus E, but the cross-section area A and the moment of inertia I differ from rod to rod. It depends on the orientations and the positions of the rods.

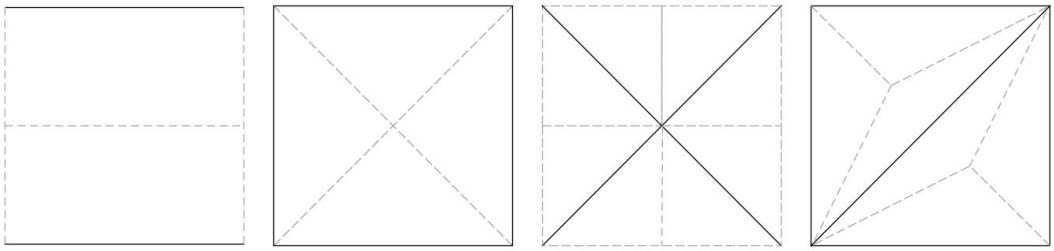

Figure 2.9: The distribution of area in 2D per model per repeating element

For each model, the total area of the beam is divided between the rods. Figure 2.9 shows the distribution of the rods per repeating element in the model with respectively horizontal rods, horizontal and vertical rods, diagonals and different types of rods. This is done in 2D. Internal rods are present in several repeating elements. In figure 2.10 it can be seen that internal rods represent double the amount of area.

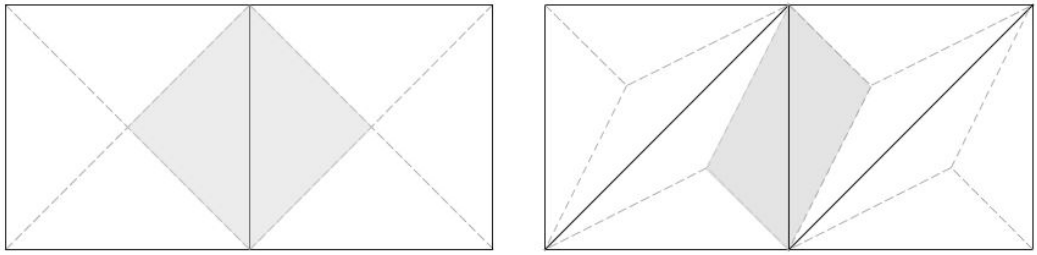

Figure 2.10: The area that a internal rod represents in different models

The model with horizontal rods contains vertical rods on the sides. To keep the distribution easy, these bars were chosen to represent only their own thickness. This is also the case for the model with diagonals. The rods on the outside of the system represents their own thickness. Since the models are in 2D, the third dimension must also be represented. This is done by multiplying the area the rods represent by the width of the beam:

$$A_{represents} * b$$

To get the area of the section per rod, the volume is divided by the length of the respective bar. The position of the rod determines the length:

- Horizontal rods have a length of dl
- Vertical rods have a length of dh
- Diagonals have a length of $\sqrt{dl^2 + dh^2}$

The moment of inertia can be calculated using the calculated cross-section area A:

$$\frac{1}{12} * b * \frac{A_{rod}}{b}$$

Once in a rod the tensile strength of plain concrete is exceeded, the local stiffness matrix of the rod is made almost equal to zero, since otherwise the models cannot solve it. Plots show in each step the rods in red when the tensile strength has been exceeded. To ensure that no duplicate plots or too many plots are made, several conditions are given in the script in Appendix B for plotting. Once the plotting is successful, the progression of the cracks for each model can be viewed. What came out of each model is shown in the next chapter.

Chapter 3

Results

After making the models, results were obtained of the stress distribution of the beam. With the stresses, an estimation can be made of where the cracks in the system may occur. Each model has its own limits and its own results, depending on the magnitude of the variables. In this chapter, the results of the models are shown and explained.

3.1 Approach

In the experiment the beam is loaded by a force P, whose value may vary. It is chosen to look at the load case of 39.45 kilogram of Newtons. The crack progression is shown in figure 2.3. Before the results can be used, the influence of the variables of the models have to be determined for each model. The following aspects of the models are considered:

- The amount of repeating elements in height
- The amount of repeating elements in length
- The amount of repeating elements in length and in height
- The thickness of the rods

The system is adjusted and viewed on a per-variable basis. The limits of the models are reached when either the tensile strength is not reached or the models cannot solve the requested structure.

To ensure that the models can be compared, it was decided to test the models with the same number of quantities of repeating elements. Sometimes this means that the results of one model are worse than with another quantity. The variables analyses with the obtained results are placed in Appendix D. With the influences of the variables known, the variables are chosen to plot the crack progressions of the models. The course of the crack progression is viewed and explained below.

3.2 Horizontal rods

The first model to be analyzed is the model with horizontal rods. A view of the model is given in figure C.1. In figure D.1 the influence of the variables is shown. Since the variables influencing the structure give little changes, it was chosen to look at the crack progression of the model with an nh of 30, an nl of 30 and a rod thickness of 0.1 mm. When step by step the force is increased, in figure 3.1 a crack develops in the system, indicated by red. It can be seen that the crack starts at the bottom of the system. The crack runs towards the middle. At some point two new cracks develop: in the upper left and around the notch. The crack at the top left develops into a full-length crack.

With this model, the limits are in the crack formation: The model consists of horizontal rods and can therefore only form a crack in horizontal direction.

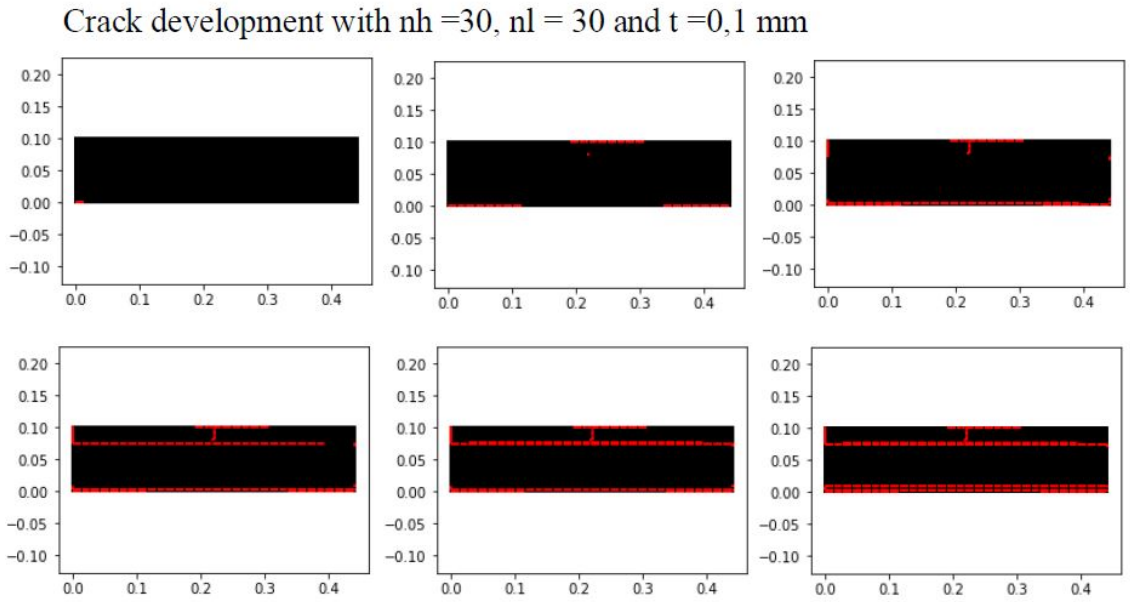

Figure 3.1: The crack development of the model with horizontal rods

3.3 Horizontal and vertical rods

To see how the cracks run when there are vertical rods in addition to the horizontal ones, a model with horizontal and vertical rods was made. The model is shown in figure C.2. The variable analysis for this model is given in figure D.2.

To look at the cracking behavior of this model, it was decided (base on the results in figure D.2) to look at the model with a nh of 30, a nl of 30 and a t of 0.1 mm. By keeping the variables of the structures low, the red area remains around the notch. A crack starts at the top at the notch. The crack at the top in figure 3.2 develops to the left, until a new crack appears below. Both continue to develop: the cracks become wider, but do not develop towards each other.

In this model, fewer rods in total are colored red: fewer cracks develop compared to the horizontal model. The cracks are more concentrated in the center of the beam. The variables do not represent limits, however, as with the horizontal model, the results can be affected by the sizes of the values. With a low amount of repeating parts, the cracks formed may appear angular.

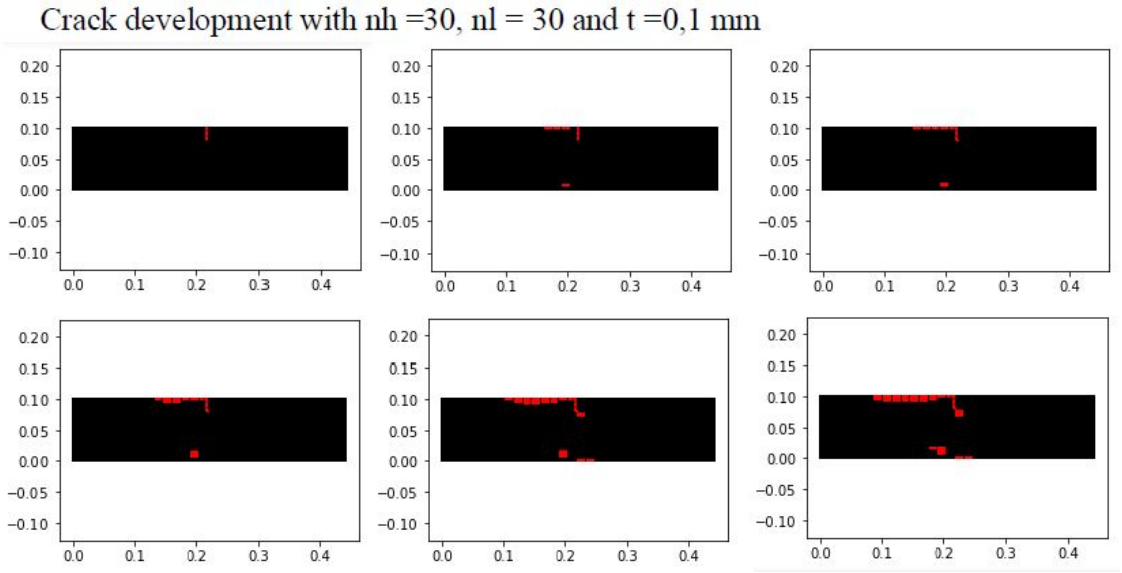

Figure 3.2: The crack development of the model with horizontal and vertical rods

3.4 Diagonals

To see the influence of oblique rods, a model was made with diagonals. The diagonals are in both directions. In the figure C.3 a picture of the structure has been made. With the data of figure D.3, it was decided to look at the crack progression of the model with a nh of 40, a nl of 70 and a t of 0.1 mm.

Figure 3.3 shows that a crack occurs at the notch and at the left boundary condition. The cracks grow toward each other. Once connected, the crack begins to grow.

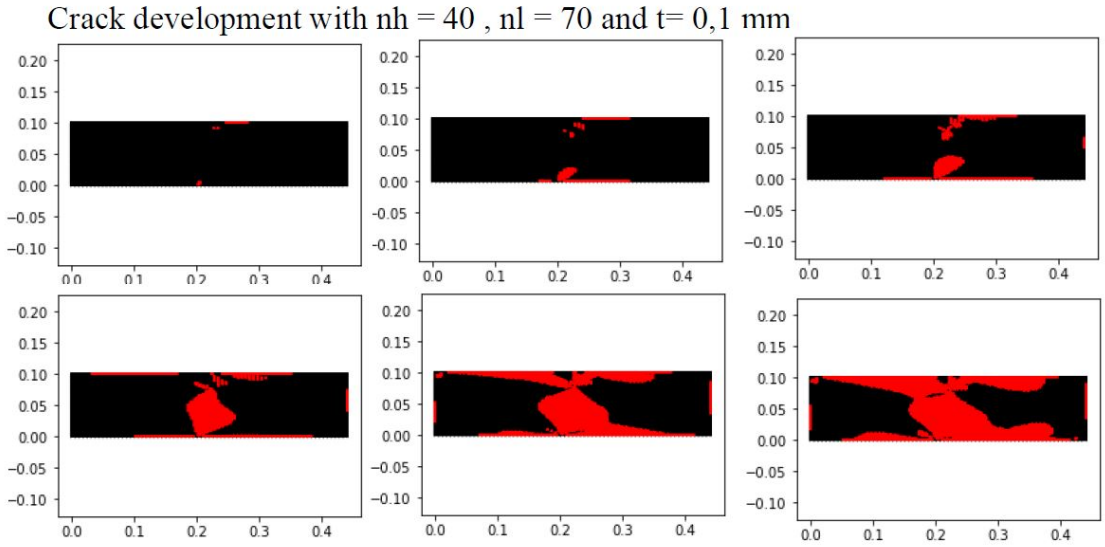

Figure 3.3: The crack development of the model with diagonals

Cracks also appear in the outer edges. At some point, the model is no longer able to handle the force: the variables must be set high before a proper result can be obtained. Compared to the other models, the distribution between the outer edge and the inner edge is uneven. The uneven distribution may have caused the weak respond of the system.

3.5 Combination models

With the combination models, horizontal rods, vertical rods and diagonals are combined. Three models were made with different type of diagonals. First the models are considered with diagonals in one direction, then the model is considered with diagonals in both directions.

3.5.1 Combination with diagonals to the left

The first combination model contains diagonals in the left direction, shown in figure C.4. Using figure D.4, it was decided to look at the crack progression of the model with a nh of 50, a nl of 60 and a t of 0.1mm.

With the addition of an orientation, it can be seen that the crack progression improves. In figure 3.4 the crack begins below the notch. The red area becomes larger.

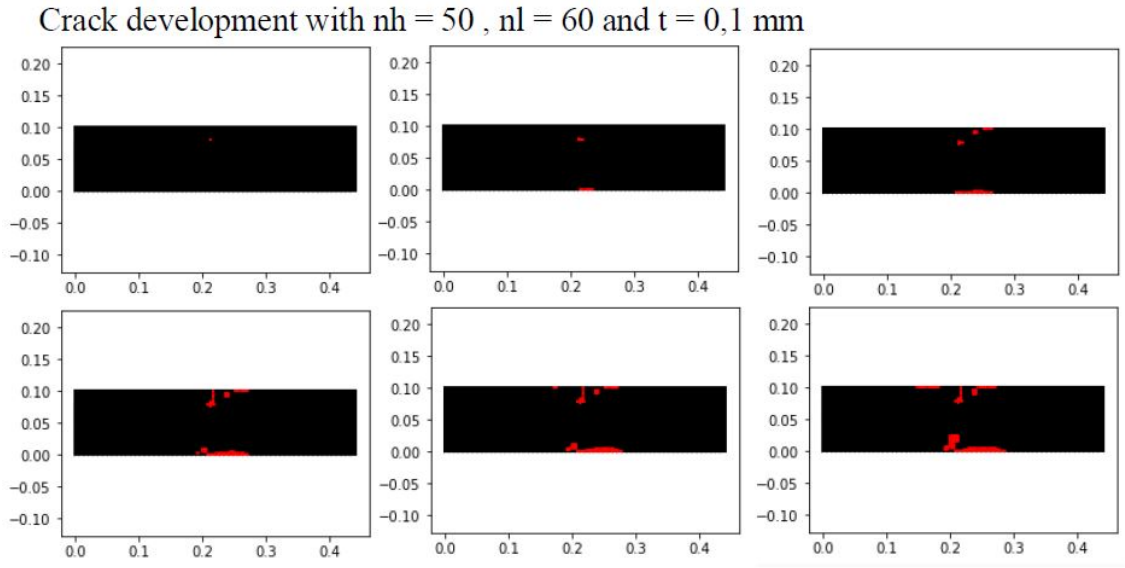

Figure 3.4: The crack development of the combination model with diagonals to the left

It is accompanied by a formation of a tear at the bottom of the system. Both cracks grow as the force increases. The area of the beam is evenly distributed, which increase the strength of the model. The direction of the crack is representative, however, the bottom of the beam should not contain any cracks.

3.5.2 Combination with diagonals to the right

The second combination has diagonals to the right. A view of the model is given in figure C.5. It was decided to look at the crack progression of the model with a nh of 50, a nl of 60 and a t of 0.1 mm. The result of the variable analysis is shown in D.5.

There are no diagonals in the direction of the crack. Compared to the combination model with the diagonals to the left, in the plots of figure 3.5, more rods achieved the tensile strength. At the notch the crack starts. Another crack arises below. The cracks grow toward each other and the red area increases. It is noteworthy that the progression of the crack differs with the direction of the diagonals changed.

3.5.3 Combination with diagonals in both directions

The last model has four different rods, since the diagonals are in different directions. The model is shown in figure C.6. With the data on the variables, it was decided to look at the crack progression of the model with a nh of 50, a nl of 60 and a t of 0.1 mm.

The plots in figure 3.6 show that the crack starts at the notch. It progresses downward toward the left boundary condition. At the bottom no crack occurs.

With changing the structure and variabels, one model can produce diffferent reults in crack progression. Some results are expected, however, some results are surprising. It is possible that the results obtained are not completely accurate, due to the limits or simplifications applied. Some aspects of the models can be suspected to deviate the results. In the next chapter, these aspects are named and discussed.

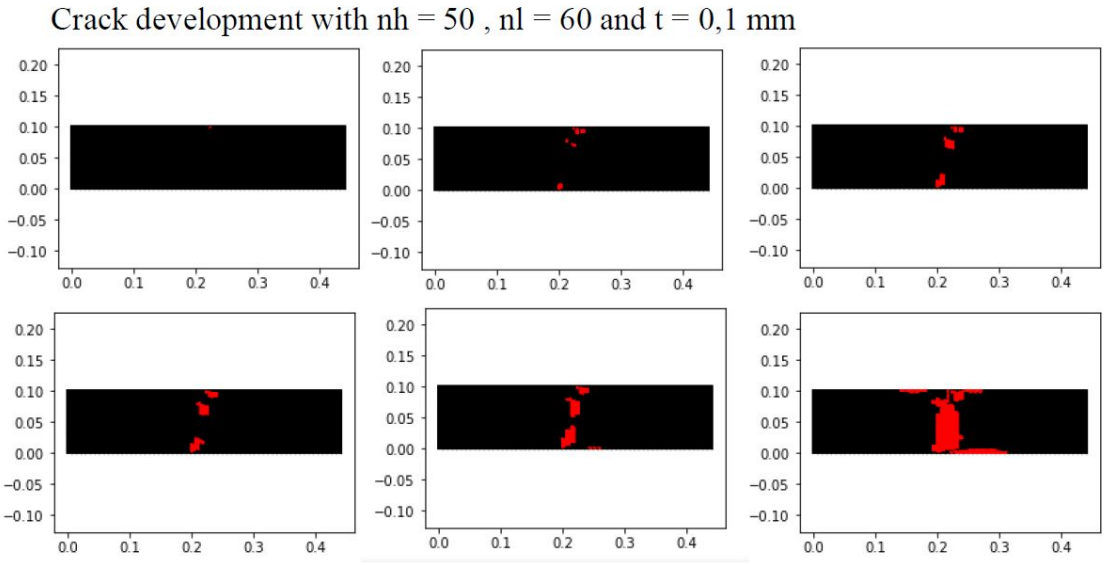

Figure 3.5: The crack development of the combination model with diagonals to the right

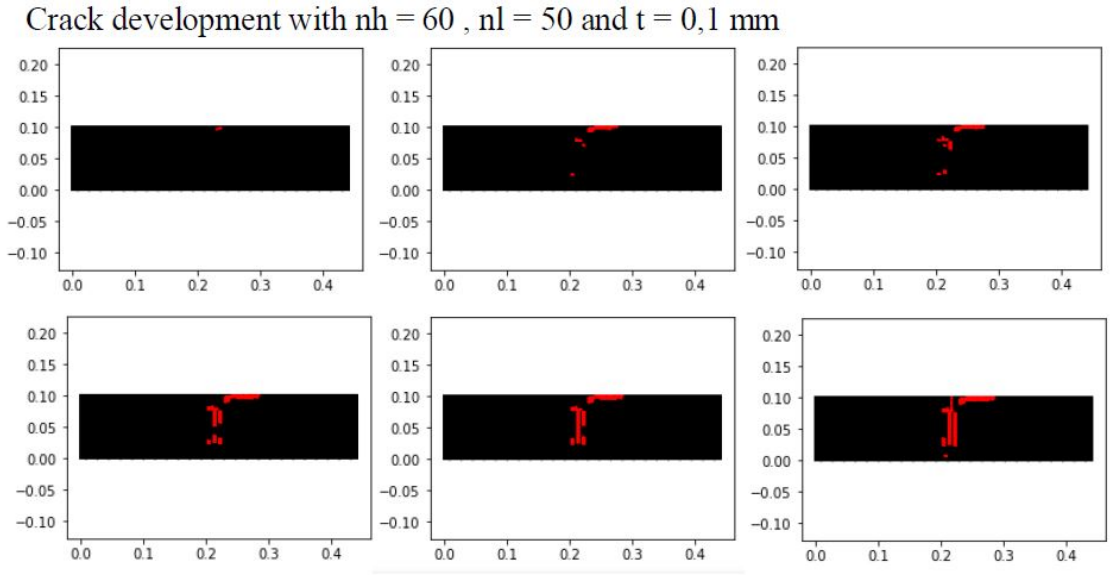

Figure 3.6: The crack development of the combination model with diagonals in both

Chapter 4

Discussion

With modeling, there are aspects that can affect the accuracy of the results obtained. The influences of these aspects can vary, depending on the differences between the models and the experiment.

The structure chosen for example can affect the results gotten from the models. The directions of the rods affects the course of the crack. When the tensile strength is exceeded, the rods are colored red. As a result, the directions of the cracks are limited by the directions of the rods. The differences varies between models since the orientations of the rods differ. When the amount of repeating elements is low the crack course is also limited. With the experiment there are no limits where the crack may or may not occur, since a cohesive material is used. The material can be seen as a model with an infinite number of rods in all directions. The rods then represent as a surface their own thickness.

When this is seen, it can be discussed that the influences of the variables are not according to expectations. In fact, some variables need to be decreased in order for the model to be more in line with the experiment. The models are not incorrect: with the given structure, the models indicate the corresponding crack progression. Compared to reality, however, it does not match. The influences of the limits cause deviations in the results.

When looking at the properties of the rods, it can also be speculated that the results are different due to an incorrect distribution in area. The beam is distributed among the rods, however, when the distribution is adjusted, it produces variation in the results obtained. In the model with diagonals it can be seen that the outer edge, representing its own thickness, fails sooner than the inner edge. The result is to be expected looking at formula A.1: as the area of the bar becomes smaller, the normal stress becomes higher. A fairer distribution ensures that the crack starts at the notch, the spot that has been made weak for the experiment.

Some elements of the model may give small variations in the results obtained. One such element is the placement of the forces and boundary conditions. The forces and boundary conditions take place at a defined coordinate. Due to a change in the number of coordinates and repeating elements, a method is used in python which indicates the coordinate which is closest to the original place.

Another element is the external force gradually increases with time. In the model, forces are placed step by step, however, the step sizes taken are large: for every thousand Newtons, the forces acting on the system are increased. When looking at the experiment, it can be discussed that the red area seen in the stress models should have been larger. Rods may have exceeded the tensile strength at a lower force. As a result, some rods last longer in the system. The magnitude of the influence is uncertain, as it does not mean that the red area needs to become larger. Rods may receive more stress, however, the rods may still not reach the tensile strength.

To understand the results and make the right conclusions, it is important to understand the points mentioned above. The models made are based on the experiment taken, however, there are limits that can give deviations to the results. When improving the model, it can be advise to change one of these aspects to see if it could differ the results.

Chapter 5

Conclusion

The goal of this research is to determine the capabilities and limitations of lattice models for describing crack development in unreinforced concrete. The models made have given results that have been compared with the results of the experiment. With the results, limits were found that have an effect on the correspondence of the models to the real. To make the models correspond with the experiment, it is important to know where the differences lie. Based on the models, conclusions can be made about whether it is possible to determine crack progression in unreinforced concrete using modeling.

The models paint a picture of the system with the given structures. In the making of the rod models, there were no complications that caused no results. The stress models are able to show the crack progression in the models with different structures. It can be concluded that the crack progression can be determined using modeling.

Nevertheless, this does not mean that the cracks obtained were all representative. Certain aspects of the models cause the correspondence to be low. So there are limits that affect the results. One of the limits is the structure given. Cracks are determined by the orientation of the rods. Once one type of rod is used in the system, it is not possible for the crack to progress in another direction. Thus, it is recommended when making a structure to use rods with different orientations and directions.

The variables of the system also affect the cracks obtained. When the rods are larger, the cracks become angular. When the amount of repeating elements is increased, it can be seen that the cracks get more round, so it is important to start the structure variables at a certain value. The influence of the variables varies from model to model. To understand the variables, it is recommended to do a variable analysis.

In addition to the structure of the system, the properties of the rods also influence the results.

The moment of inertia and cross-sectional area of the rods were determined based on the position and the orientation of the rod. Since the rods represent a part of the system, the distribution of the area affects the results obtained. A fair distribution of area ensures that no weak spots are created in the system. However, it is important that everything is represented.

By adjusting the structure and determining the properties of the rods, images of the crack progression of plain concrete can be obtained using modeling. The cracks obtained may not be as expected, but by researching modeling, a foundation has been laid for using modeling to create structures in the future. All models made in this project have performed their function in this regard.

Appendix A

Basic Mechanics

Basic mechanics are applied to determine the stresses in the rods. Stresses occur when the internal structure or surface resists a force that causes or attempts to cause a deformation. The forces are in different directions. A force in the direction of the material is called the normal force. The normal force F in figure A.1 can cause the material to extend or shrink. It creates tension in the beam, called normal stress. If a homogeneous cross section is assumed (a cross section made of the same material throughout), the normal stress σ can be calculated by dividing the normal force F by the cross section A. The normal stress is equally distributed over the entire cross section [3].

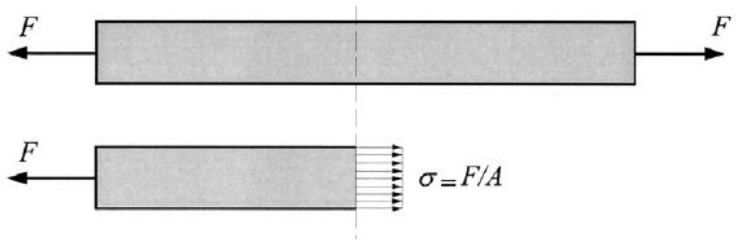

Figure A.1: The normal stress of a prismatic bar [3]

When the material is subjected to a load at a particular point, bending can occur. A moment is a force at a certain distance, which can cause rotation. When a material is subjected by a moment M, bending stresses occur. The magnitude of these stresses depend on the material resistance to change in angular velocity with a certain mass (moment of inertia) I and the height of the cross section. Using formula A.1, the total stress σ can be calculated [3]. In the formula the normal stress is given with the symbol N. The symbol z indicates half of the height of the beam.

$$\sigma = \frac{N}{A} + \frac{M * z}{I} \tag{A.1}$$

The strength of a material is the resistance that must be overcome to break the cohesion of the material [3]. The strength of compression or tension can differ.

Appendix B

Python code

This section of the report briefly explains the script made in Python to obtain the models. First, the script for the rod model is explained. This is followed by the script for the stress model.

Import These are the packages needed in Python to use CALFEM and to create the models.

```
import calfem.core as cfc
import calfem.geometry as cfg
import calfem.mesh as cfm
import calfem.vis as cfv
import calfem.utils as cfu
import calfem.vis_mpl as cfvm
import visvis as vv
import numpy as np
import matplotlib.pyplot as plt
import math
%matplotlib inline
```

B.1 The rod model

Input variables These are the variables specifically for the single-edge notched beam under four point loading. The dimensions of the rods are based on the dimensions of the beam.

```
#dimensions
b = 100/1000
h = 100/1000
l1= 200/1000
l2= l1 + 220/1000
l3= l2 + 20/1000
lh= 220/1000
l= 440/1000

#the amount of squares
nh= 3
nl= 12

#the values of the cubes
dh= h/nh
dl= l/nl

#the thickness of the rods
t = 1*10**-4
```

Characteristics rods The characteristics of the rods include the Young's modulus E, the cross-section area A and the moment of inertia I. Since the rods differ in orientation and position, the cross-section and moment of inertia differ per rod. The four different distribution of area in figure 2.9 needed to be taken to account. For the model with the combinations, the same characteristics are assigned.

Horizontal rods

```
#Young modulus is th same at every point
E = 24.8*10**6

#the vertical rods:
Abv = t*b
Iv = 1/12*b*t**3

#the horizontal rods:
Abh = (1/4*dh*dl*b)/dl
Afh = (2/4*dh*dl*b)/dl
Ibh = 1/12*b*((Abh/b)**3)
Ifh = 1/12*b*((Afh/b)**3)

#the ep needed
epbv = np.array(([E,Abv,Iv]))
epbh = np.array(([E,Abh,Ibh]))
epfh = np.array(([E,Afh,Ifh]))
```

Horizontal and vertical rods

```
#Young modulus is th same at every point
E = 24.8*10**6

#the vertical rods:
Abv = (1/4*dh*dl*b)/dh
Afv = (2/4*dh*dl*b)/dh
Ibv = 1/12*b*((Abv/b)**3)
Ifv = 1/12*b*((Afv/b)**3)

#the horizontal rods:
Abh = (1/4*dh*dl*b)/dl
Afh = (2/4*dh*dl*b)/dl
Ibh = 1/12*b*((Abh/b)**3)
Ifh = 1/12*b*((Afh/b)**3)

#the ep needed
epbv = np.array(([E,Abv,Ibv]))
epfv = np.array(([E,Afv,Ifv]))
epbh = np.array(([E,Abh,Ibh]))
epfh = np.array(([E,Afh,Ifh]))
```

Diagonals

```
#Young modulus is th same at every point
E = 24.8*10**6

#the vertical rods:
Abv = t*b
Ibv = 1/12*b*(t**3)

#the horizontal rods:
Abh = t*b
Ibh = 1/12*b*(t**3)

#The diagonals:
Afd = ((dh)*b*(dl)*(1/4))/(np.sqrt((1/2*dl)**2 + (1/2*dh)**2))
Ifd = 1/12*b*((Afd/b)**3)
Ifv = Ibv

#the ep needed
epbv = np.array(([E,Abv,Ibv]))
epbh = np.array(([E,Abh,Ibh]))
epfd = np.array(([E,Afd,Ifd]))
epfv = epbv
```

Combination models

```
#Young modulus is th same at every point
E = 24.8*10**6

#the vertical rods:
Abv = (1/6*dh*dl*b)/dh
Afv = (1/3*dh*dl*b)/dh
Ibv = 1/12*b*((Abv/b)**3)
Ifv = 1/12*b*((Afv/b)**3)

#the horizontal rods:
Abh = (1/6*dh*dl*b)/dl
Afh = (1/3*dh*dl*b)/dl
Ibh = 1/12*b*((Abh/b)**3)
Ifh = 1/12*b*((Afh/b)**3)

#The diagonals:
Afd = (1/3*dh*dl*b)/(np.sqrt((dl)**2 + (dh)**2))
Ifd = 1/12*b*(Afd/b)**3

#the ep needed
epbv = np.array(([E,Abv,Ibv]))
epfv = np.array(([E,Afv,Ifv]))
epbh = np.array(([E,Abh,Ibh]))
epfh = np.array(([E,Afh,Ifh]))
epfd = np.array(([E,Afd,Ifd]))
```

Coordinates A matrix is made, Coord, with the coordinates of each point in the X-Y plane as seen in figure 2.4. Coordinates are created for the straight rods and the oblique rods. The naming of the coordinates takes into account the boundaries of the system and the location of the notch. The coordinates of the notch are fixed numbers and do not vary due to the change of repeating elements in the system. Stacking the different coordinates using the function *vstack* gives the total coordinate system.

Before the notch

```
x= np.arange(nl+1)*dl
y= np.arange(0,0.08,dh)
ad= y[-1]

x,y = np.meshgrid(x,y)
xy = np.array((x,y))
n0 = (nl+1)*len(y)
points0= np.reshape(xy,(2,n0))
```

```
x2= np.arange(nl)*dl +0.5*dl
y= np.arange(0,0.08,dh)
y2= np.arange(0,0.08, dh) + 0.5*dh
y2= y2[(y2 <= 0.08)]
y2 = y2[(y2 < y[-1])]

n00 = len(x2)*len(y2)
x2,y2 = np.meshgrid(x2,y2)
xy2 = np.array((x2,y2))
points00= np.reshape(xy2,(2,n00))
```

Left of the notch

```
xarr1 = np.arange(0,0.2175, dl)
a = xarr1[-1]
value11 = (0.2175 + a)/2
obj = np.shape(xarr1)
value = 0.2175
xarr1 = np.insert(xarr1, obj, value, axis=None)
leng1 = len(xarr1)
yarr1= np.array(np.arange(0.1,0.08,-dh)[::-1])

n1 = len(xarr1)*len(yarr1)
xarr1,yarr1 = np.meshgrid(xarr1,yarr1)
xy1 = np.array((xarr1,yarr1))
points1= np.reshape(xy1,(2,n1))
```

```
xarr11 = np.arange(0,0.2175, dl) + 0.5*dl
xarr11= xarr11[(xarr11 <= 0.2175)]
xarr11 = np.array(xarr11)
if nl % 2 != 0 and nl != 0 :
    print('nl is', nl, 'so oneven')
    obj11 = len(xarr11)
    xarr11 = np.insert(xarr11, obj11, value11, axis=None)
lx11= len(xarr11)
yarr11= np.array(np.arange(0.1,0.08,-dh)[::-1]) - 0.5*dh
ly11 = len(yarr11)

n11 = len(xarr11)*len(yarr11)
xarr11,yarr11 = np.meshgrid(xarr11,yarr11)
xy11= np.array((xarr11,yarr11))
points11= np.reshape(xy11,(2,n11))
```

Right of the notch

```
xarr2 = np.arange(0.440,0.2225, -dl)[::-1]
obj = 0
value = 0.2225
xarr2 = np.insert(xarr2, obj, value, axis=None)
leng2 = len(xarr2)
yarr2= np.array(np.arange(0.1,0.08,-dh)[::-1])

n2 = len(xarr2)*len(yarr2)
xarr2,yarr2 = np.meshgrid(xarr2,yarr2)
xy2 = np.array((xarr2,yarr2))
points2= np.reshape(xy2,(2,n2))
```

```
xarr2 = np.arange(0.440,0.2225, -dl)[::-1]
xarr22 = xarr2 - 0.5*dl
xarr22= xarr22[(xarr22 >= 0.2225)]
if nl % 2 != 0 and nl != 0 :
    print('nl is', nl, 'so oneven')
    value22 = (0.2225 + xarr2[0])/2
    print(value22)
    obj22 = 0
    xarr22 = np.insert(xarr22, obj22, value22, axis=None)
lx22= len(xarr22)
yarr22= np.array(np.arange(0.1,0.08,-dh)[::-1]) - 0.5*dh
ly22 = len(yarr22)

n22 = len(xarr22)* len(yarr22)
xarr22,yarr22 = np.meshgrid(xarr22,yarr22)
xy22= np.array((xarr22,yarr22))
points22= np.reshape(xy22,(2,n22))
```

At the notch

```
xgek= np.arange(0.2175, 0.22275, 0.005)
ygek= [0.8]*2

xgek,ygek= np.meshgrid(xgek,ygek)
xygek= np.array(([0.2175,0.22, 0.2225], [0.08, 0.08, 0.08]))
n3= 3
points3 = np.reshape(xygek, (2,n3))
```

All the coordinates

```
points = np.hstack([points0, points1, points2, points3, points00, points11, points22])
coord = np.swapaxes(points,0,1)
coord = np.array(coord)
add= np.array(([0.22, ad]))
coord=np.vstack((coord, add))
```

Dof With the coordinate system, a matrix, Dof, must also be created using the function *createdofs* for the three degrees of freedom: horizontal plane, vertical plane and rotation. Each rod has three degrees of freedom. The first point is indicated by [1,2,3], the second point by [4,5,6], and so on.

```
dof= cfc.createdofs(len(coord),3)
```

Edof The Edof matrix is an array where every line in the array corresponds to an rod. The array [1; 2; 3; 4; 5; 6] will be an rod that connects grid point 1 and 2. And because of the corresponding coordinates, the elements are specified which results in the entire lattice-structure. For the rod models, the outer edge of the system is made first. Then the structures are defined and applied to how it is desired. Four different structures are used for this research. The models with diagonals in both directions can be obtained by combining the structure with the diagonals to the left with the structure with the diagonals to the right.

Border

```
#6 omdat 2 keer 3 graden van vrijheid is
edofh= np.zeros((stavenh,6))

#below notch
#horizontal beams
for i in range(nl):
    edofh[i] = np.hstack((np.array(dof[i]),
                                np.array(dof[i+1])))
#with notch
#horizontalleft
for i in range(leng1-1):
        edofh[i+ st1h] = np.hstack((np.array(dof[i + A + G]),
                                 np.array(dof[i+1 + A + G])))
#horizontal right
for i in range(leng2-1):
    edofh[i + st1h +st2h1] = np.hstack((np.array(dof[i +  B +I]),
                                      np.array(dof[i+1+  B+ I])))
edofh[stavenh-2]= np.hstack((np.array(dof[n0 +n1+n2]),
                             np.array(dof[n0 +n1+n2+1])))
edofh[stavenh-1]= np.hstack((np.array(dof[n0 +n1+n2+1]),
                             np.array(dof[n0 +n1+n2+2])))
```

```
edofv = np.zeros((stavenv,6))
#below notch
#vertical beams left
for j in range(len(y)-1):
    for i in range(1):
        edofv[i +j*(1)] = np.hstack((np.array(dof[i+j*(nl+1)]),
                                     np.array(dof[i+(j+1)*(nl+1)])))
#vertical beams right
for j in range(len(y)-1):
    edofv[st1v +j] = np.hstack((np.array(dof[nl+j*(nl+1)]),
                                np.array(dof[nl+(j+1)*(nl+1)])))
#with notch
#vertical left
for j in range(len(yarr1)-1):
    for i in range(1):
            edofv[2*st1v +j] = np.hstack((np.array(dof[i+j*(leng1)+ A]),
                                          np.array(dof[i+(j+1)*(leng1)+ A])))
#vertical left first row
for i in range(1):
    edofv[2*st1v +st2v1 + i] = np.hstack((np.array(dof[i+ C]),
                                          np.array(dof[i+ D])))
#vertical right
for j in range(len(yarr2)-1):
    edofv[2*st1v + st2v1 +1 + j ] = np.hstack((np.array(dof[(j+1)*leng2+B-1]),
                                           np.array(dof[(j+2)*leng2 + B - 1])))
#vertical right first row
edofv[stavenv-3 ] = np.hstack((np.array(dof[ F ]),
                               np.array(dof[ J])))
edofv[stavenv-2]= np.hstack((np.array(dof[n0 +n1 + n2+2]),
                             np.array(dof[n1 +n2 +n0 -leng2])))
edofv[stavenv-1]= np.hstack((np.array(dof[n0 +n1 + n2]),
                             np.array(dof[n1 +n0 -1])))
```

```
edofb= np.vstack((edofh,edofv))
```

Horizontal rods filling

```
edoffh= np.zeros((filling,6))

#below notch
#horizontal beams
for j in range(len(y)-1):
    for i in range(nl):
        edoffh[i+j*nl] = np.hstack((np.array(dof[i+(j+1)*(nl+1)]),
                                    np.array(dof[i+1+(j+1)*(nl+1)])))
#with notch
#horizontal1
for j in range(len(yarr1)-1):
    for i in range(leng1-1):
        edoffh[i+j*(leng1-1) + f1h] = np.hstack((np.array(dof[i + j*leng1 + A]),
                                                 np.array(dof[i+1 +j*leng1+ A])))
#horizontal2
for j in range(len(yarr2)-1):
    for i in range(leng2-1):
        edoffh[i+j*(leng2-1) + f1h + f2h1] = np.hstack((np.array(dof[i + j*leng2 + B]),
                                                 np.array(dof[i+1 +j*leng2 + B])))
```

Vertical rods filling

```
edoffv= np.zeros((filling ,6))

#below notch
#the vertical beams
for j in range(len(y)-1):
    for i in range(nl-1):
        edoffv[i+j*(nl-1)] = np.hstack((np.array(dof[i+1+j*(nl+1)]),
                                        np.array(dof[i+1+(j+1)*(nl+1)])))
#with notch
#vertical1 first row
for i in range(leng1-2):
    edoffv[i + f1v] = np.hstack((np.array(dof[i+ 1+ C]),
                                 np.array(dof[i+ 1+ D])))
#vertical1 other rows
for j in range(len(yarr1)-1):
    for i in range(leng1-2):
            edoffv[i+j*(leng1-2)+ f1v + f2v1f] = np.hstack((np.array(dof[i+1+j*(leng1)+ A]),
                                        np.array(dof[i+1+(j+1)*(leng1)+ A])))
#vertical2 first row
for i in range(leng2-2):
    edoffv[i+ f1v + f2v1 ] = np.hstack((np.array(dof[i+ F]),
                                        np.array(dof[i+ D +leng1*len(yarr1)+1])))
#vertical2 other rows
for j in range(len(yarr2)-1):
    for i in range(leng2-2):
        edoffv[i+j*(leng2-2) + f1v + f2v1 +f2v2f ] = np.hstack((np.array(dof[i+1+j*(leng2)+ B]),
                                             np.array(dof[i+1+(j+1)*(leng2)+ B])))
```

Diagonals to the left filling

```
diag0= nl*(len(y)-1)
edoff0 = np.zeros((diag0*2,6))

#below notch
for j in range(len(y)-1):
    for i in range(nl):
        edoff0[i+j*(nl)] = np.hstack((np.array(dof[j*(nl+1)+ i]), np.array(dof[H + j*nl+ i])))
        edoff0[i+j*(nl) + diag0] = np.hstack((np.array(dof[(j+1)*(nl+1)+ i+1]), np.array(dof[H + j*nl+ i])))
```

```
diag1 = lx11*ly11 # each kind of rod left of the notch
c = int(diag1*2)
edoff1 = np.zeros((c,6))

#with notch
#left
for j in range(ly11):
    for i in range(lx11):
        edoff1[i+j*(lx11) ] = np.hstack((np.array(dof[j*leng1+ i+1 +n0]), np.array(dof[H + j*lx11+ i+ n00])))

for j in range(ly11-1):
    for i in range(lx11):
        edoff1[i+j*(lx11) + diag1] = np.hstack((np.array(dof[j*leng1+ i+1 + n0]), np.array(dof[H + (j+1)*lx11+ i+ n00])))

for i in range(lx11):
    edoff1[i +diag1*2 -lx11] = np.hstack((np.array(dof[i + n0-(nl+1)]), np.array(dof[H + i+ n00])))
```

```
diag2 = lx22*ly22 # each kind of rod right of the notch
b = int(diag2*2)
edoff2 = np.zeros((b,6))

#with notch
#right
for j in range(ly22):
    for i in range(lx22):
        edoff2[i+j*(lx22)] = np.hstack((np.array(dof[j*leng2+ i +1+ n0 +n1 ]), np.array(dof[H + j*lx11+ i+ n00 +n11])))

for j in range(ly22-1):
    for i in range(lx22):
        edoff2[i+j*(lx22) + diag2] = np.hstack((np.array(dof[j*leng2+ i +1+ n0 + n1 ]), np.array(dof[H + (j+1)*lx11+ i+ n00+n1
1])))

for i in range(lx22):
    edoff2[i +diag2*2 -lx22] = np.hstack((np.array(dof[i + n0-lx22]), np.array(dof[H + i+ n00 +n11+1])))

edoff2[len(edoff2)-1]= np.hstack((np.array(dof[len(coord)-1]), np.array(dof[H + n00 +n11])))
```

Diagonals to the right

```
diag0= nl*(len(y)-1)
edoff0 = np.zeros((diag0*2,6))

#below notch
for j in range(len(y)-1):
    for i in range(nl):
        edoff0[i+j*(nl)] = np.hstack((np.array(dof[j*(nl+1)+ i+1]), np.array(dof[H + j*nl+ i])))
        edoff0[i+j*(nl) + diag0] = np.hstack((np.array(dof[(j+1)*(nl+1)+ i]), np.array(dof[H + j*nl+ i])))
```

```
diag1 = lx11*ly11 # each kind of rod left of the notch
c = int(diag1*2)
edoff1 = np.zeros((c,6))

#with notch
#Left
for j in range(ly11):
    for i in range(lx11):
        edoff1[i+j*(lx11) ] = np.hstack((np.array(dof[j*leng1+ i +n0]), np.array(dof[H + j*lx11+ i+ n00])))

for j in range(ly11-1):
    for i in range(lx11):
        edoff1[i+j*(lx11) + diag1] = np.hstack((np.array(dof[j*leng1+ i + 1+ n0]), np.array(dof[H + (j+1)*lx11+ i+ n00])))

for i in range(lx11-1):
    edoff1[i +diag1*2 -lx11] = np.hstack((np.array(dof[i + n0-(nl)]), np.array(dof[H + i+ n00])))

edoff1[len(edoff1)-1]= np.hstack((np.array(dof[len(coord)-1]), np.array(dof[H + n00 + lx11 -1])))
```

```
diag2 = lx22*ly22 # each kind of rod right of the notch
b = int(diag2*2)
edoff2 = np.zeros((b,6))

#with notch
#right
for j in range(ly22):
    for i in range(lx22):
        edoff2[i+j*(lx22) ] = np.hstack((np.array(dof[j*leng2+ i +n0 +n1]), np.array(dof[H + j*lx11+ i+ n00 +n11])))

for j in range(ly22-1):
    for i in range(lx22):
        edoff2[i+j*(lx22) + diag2] = np.hstack((np.array(dof[j*leng2+ i + n0 + n1+1 ]), np.array(dof[H + (j+1)*lx11+ i+ n00+n1
1])))

for i in range(lx22):
    edoff2[i +diag2*2 -lx22] = np.hstack((np.array(dof[i + n0-lx22]), np.array(dof[H +  i+ n00 +n11])))
```

B.2 Stress models

Boundary conditions and external forces After defining the system, the boundary conditions and the external forces needed to be defined. Because the amount of coordinates change in the system, the forces cannot always be placed in the correct position in the system. To overcome this, the coordinate that is closest is chosen. If the direction of the force is opposite to the coordinate system, the value of the force must be entered as negative.

Boundary conditions

```
xb= np.arange(nl+1)*dl
#first boundary condition: restriction vertical and horizontal
bc=[]
lenghtI= l1
print()
myarray= np.array(xb)
pos=(np.abs(xb-lenghtI)).argmin()
bc=np.append(bc, dof[pos,0:2])

#second boundary condition: restriction vertical
lenghtII= l2
myarray= np.array(xb)
pos= (np.abs(xb-lenghtII)).argmin()
bc=np.append(bc, dof[pos,1])
bcVal=np.zeros((1,3))
```

External forces

```
xp1 = np.arange(0,0.2175, dl)
xp2 = np.arange(0.440,0.2225, -dl)[::-1]
```

```
lp1= 20/1000
#Location first force
pp=[]
lenghtp1= lp1
myarray= np.array(xp1)

pos1= (np.abs(xp1-lenghtp1)).argmin()
pp = np.append(pp, xp1[pos1])
p1 = (nl+1)*len(y) + leng1*(len(yarr1)-1) + pos1
```

```
lp2= 240/1000
#Location second force
pp2=[]
lenghtp2= lp2
myarray= np.array(xp2)

pos2= (np.abs(xp2-lenghtp2)).argmin()
pp = np.append(pp2, xp2[pos2])
p2 = (nl+1)*len(y) + leng1*len(yarr1) + leng2*(len(yarr2)-1) + pos2 + 1
```

Solving the system Now that the entire structure is defined, the stresses can be found. The models begin by defining the matrices for the force and the global stiffness. Each rod has a local stiffness matrix, calculated by the function *beam2e*. Using *if* statements, the right characteristics are used per rod. Through the function *assem*, all local stiffness matrices are combined into the global stiffness matrix. After the stiffness matrix is determined, the forces are applied to the structure in steps. At each step, the system is solved by using the function *solveq*. For each rod, the forces acting on the end of the rod are calculated using the function *beam2s*. Again *if* statements were used for the characteristics of the rods. By using the formula 1.1 the stresses are determined. When the stress of a rod is greater than the tensile strength, the local stiffness matrix is made almost equal to zero. This is by adding the negative version of the local stiffness matrix with the function *assem* in the global stiffness matrix. The system is then solved with the new stiffness matrix and an increased force. This is repeated until the end of the range (the maximum force) is reached.

In the meantime, plots are made of the results obtained. When a rod reaches the tensile strength of plain concrete, it is added with the function *append* in an empty list called edoft. Using *eldraw*, the rod model is plotted in black. The rods named in edoft are plotted in red by the same function. The red then represents the crack. To ensure that there are no duplicate plots, terms such as 'before'and 'previous' are used. The term 'k' causes a reduction in the amount of plots made.

For the six models, the structure of the code is the same, however, the difference is in the if statements. First, the initial global stiffness matrix is given by model. These are followed by the rest of the code.

Model with horizontal rods and horizontal and vertical rods

```
K= np.zeros((3*len(coord), 3*len(coord)))
f = np.zeros((3*len(coord), 1))

ex,ey= cfc.coordxtr(edof,coord,dof)
edof_int = np.int_(edof)

q = len(edofb)+ len(edoffh)
r = len(edofb)+ len(edoffh) + len(edoffv)

for i in range(len(edof)):
    if i <= len(edofh):
        Ke = cfc.beam2e(ex[i,:],ey[i,:],epbh)
    if i > len(edofh) and i <= len(edofb):
        Ke = cfc.beam2e(ex[i,:],ey[i,:],epbv)
    if i > len(edofb) and i <= q :
        Ke = cfc.beam2e(ex[i,:],ey[i,:],epfh)
    else:
        Ke = cfc.beam2e(ex[i,:],ey[i,:],epfv)
    K = cfc.assem(edof_int[i,:],K,Ke)
```

Model with diagonals

```
K= np.zeros((3*len(coord), 3*len(coord)))
f = np.zeros((3*len(coord), 1))

ex,ey= cfc.coordxtr(edof,coord,dof)
edof_int = np.int_(edof)

q = len(edofb) + len(edoff0) + len(edoff1) + len(edoff2)

for i in range(len(edof)):
    if i <= len(edofh):
        Ke = cfc.beam2e(ex[i,:],ey[i,:],epbh)
    if i > len(edofh) and i <= len(edofb):
        Ke = cfc.beam2e(ex[i,:],ey[i,:],epbv)
    if i > len(edofb) and i <= q :
        Ke = cfc.beam2e(ex[i,:],ey[i,:],epfd)
    else:
        Ke = cfc.beam2e(ex[i,:],ey[i,:],epfv)
    K = cfc.assem(edof_int[i,:],K,Ke)
```

Models with combination

```
q = len(edofb)+ len(edoffh) +len(edoffv)
r = len(edofb)+ len(edoffh) + len(edoffv) + len(edoff0) + len(edoff1) + len(edoff2)

K= np.zeros((3*len(coord), 3*len(coord)))
f = np.zeros((3*len(coord), 1))

ex,ey= cfc.coordxtr(edof,coord,dof)
edof_int = np.int_(edof)

q = len(edofb)+ + len(edoff0) + len(edoff1) + len(edoff2)

for i in range(len(edof)):
    if i <= len(edofh):
        Ke = cfc.beam2e(ex[i,:],ey[i,:],epbh)
    if i > len(edofh) and i <= len(edofb):
        Ke = cfc.beam2e(ex[i,:],ey[i,:],epbv)
    if i > len(edofb) and i <= q :
        Ke = cfc.beam2e(ex[i,:],ey[i,:],epfh)
    if i > q + len(edoffv) and i <= r :
        Ke = cfc.beam2e(ex[i,:],ey[i,:],epfd)
    else:
        Ke = cfc.beam2e(ex[i,:],ey[i,:],epfv)
    K = cfc.assem(edof_int[i,:],K,Ke)
```

Model with horizontal rods and horizontal and vertical rods

```
q = len(edofb)+ len(edoffh)

P= 39

done = np.zeros((len(edof),1))
for i in range(P):
    if i > 0:
        before = len(edoft)
    f[3*p1-2]= -1/11*i
    f[3*p2-2]= -10/11*i

    a,r = cfc.solveq(K,f,np.int_(bc),bcVal)
    ed= cfc.extract_eldisp(edof_int,a)
    for j in range(len(edof)):
        if j <= len(edofh):
            es,edi,eci = cfc.beam2s(ex[j,:], ey[j,:], epbh, ed[j,:])
            tension = es[:,0]/epbh[1] + (es[:,2]*1/2*t)/Ibh
        if j > len(edofh) and j <= len(edofb):
            es,edi,eci = cfc.beam2s(ex[j,:], ey[j,:], epbv, ed[j,:])
            tension = es[:,0]/epbv[1] + (es[:,2]*1/2*t)/Ibv
        if j > len(edofb) and j <= q :
            es,edi,eci = cfc.beam2s(ex[j,:], ey[j,:], epfh, ed[j,:])
            tension = es[:,0]/epfh[1] + (es[:,2]*1/2*t)/Ifh
        else:
            es,edi,eci = cfc.beam2s(ex[j,:], ey[j,:], epbv, ed[j,:])
            tension = es[:,0]/epbv[1] + (es[:,2]*1/2*t)/Ifv

        if tension[0] > 2.8*10**3:
            done[j]+=1
            c = 0.00000000000001
            if done[j] < 1:
                if j <= len(edofh):
                    Ke = -cfc.beam2e(ex[j,:],ey[j,:],epbh) + c
                if j > len(edofh) and j <= len(edofb):
                    Ke = -cfc.beam2e(ex[j,:],ey[j,:],epbv) + c
                if j > len(edofb) and j <= q :
                    Ke = -cfc.beam2e(ex[j,:],ey[j,:],epfh) + c
                else:
                    Ke = -cfc.beam2e(ex[j,:],ey[j,:],epbv) + c
                K = cfc.assem(edof_int[j,:],K,Ke)

                a,r = cfc.solveq(K,f,np.int_(bc),bcVal)
                ed= cfc.extract_eldisp(edof_int,a)
```

```
    edoft = []
    for k in range(len(edof)):
        if i > 0:
            previous = len(edoft)
        if k <= len(edofh):
            es,edi,eci = cfc.beam2s(ex[k,:], ey[k,:], epbh, ed[k,:])
            tension = es[:,0]/epbh[1] + (es[:,2]*1/2*t)/Ibh
        if k > len(edofh) and k <= len(edofb):
            es,edi,eci = cfc.beam2s(ex[k,:], ey[k,:], epbv, ed[k,:])
            tension = es[:,0]/epbv[1] + (es[:,2]*1/2*t)/Ibv
        if k > len(edofb) and k <= q :
            es,edi,eci = cfc.beam2s(ex[k,:], ey[k,:], epfh, ed[k,:])
            tension = es[:,0]/epfh[1] + (es[:,2]*1/2*t)/Ifh
        else:
            es,edi,eci = cfc.beam2s(ex[k,:], ey[k,:], epbv, ed[k,:])
            tension = es[:,0]/epbv[1] + (es[:,2]*1/2*t)/Ifv
        if tension[0] > 2.8*10**3:
            edoft.append(np.array(edof[k]))
        if i == 0:
            if len(edoft)>= 1:
                ex3,ey3= cfc.coordxtr(edoft,coord,dof)
                cfvm.figure()
                cfvm.eldraw2(ex,ey,[1,1,0])
                cfvm.eldraw2(ex3,ey3,[2,4,0])
                cfvm.showAndWait()
        if i > 0:
            if len(edoft)>= 1:
                if len(edoft)> before:
                    if len(edoft) > previous:
                        if (k % 1 == 0):
                            ex3,ey3= cfc.coordxtr(edoft,coord,dof)
                            cfvm.figure()
                            cfvm.eldraw2(ex,ey,[1,1,0])
                            cfvm.eldraw2(ex3,ey3,[2,4,0])
                            cfvm.showAndWait()
```

Model with diagonals

```
q = len(edofb) + len(edoff0) + len(edoff1) + len(edoff2)

P= 39

done = np.zeros((len(edof),1))
for i in range(P):
    if i > 0:
        before = len(edoft)
    f[3*p1-2]= -1/11*i
    f[3*p2-2]= -10/11*i

    a,r = cfc.solveq(K,f,np.int_(bc),bcVal)
    ed= cfc.extract_eldisp(edof_int,a)
    for j in range(len(edof)):
        if j <= len(edofh):
            es,edi,eci = cfc.beam2s(ex[j,:], ey[j,:], epbh, ed[j,:])
            tension = es[:,0]/epbh[1] + (es[:,2]*1/2*t)/Ibh
        if j > len(edofh) and j <= len(edofb):
            es,edi,eci = cfc.beam2s(ex[j,:], ey[j,:], epbv, ed[j,:])
            tension = es[:,0]/epbv[1] + (es[:,2]*1/2*t)/Ibv
        if j > len(edofb) and j <= q :
            es,edi,eci = cfc.beam2s(ex[j,:], ey[j,:], epfd, ed[j,:])
            tension = es[:,0]/epfd[1] + (es[:,2]*1/2*t)/Ifd
        else:
            es,edi,eci = cfc.beam2s(ex[j,:], ey[j,:], epfv, ed[j,:])
            tension = es[:,0]/epfv[1] + (es[:,2]*1/2*t)/Ifv

        if tension[0] > 2.8*10**3:
            done[j]+=1
            c = 0.00000000000001
            if done[j] < 1:
                if j <= len(edofh):
                    Ke = -cfc.beam2e(ex[j,:],ey[j,:],epbh) + c
                if j > len(edofh) and j <= len(edofb):
                    Ke = -cfc.beam2e(ex[j,:],ey[j,:],epbv) + c
                if j > len(edofb) and j <= q :
                    Ke = -cfc.beam2e(ex[j,:],ey[j,:],epfd) + c
                else:
                    Ke = -cfc.beam2e(ex[j,:],ey[j,:],epfv) + c
                K = cfc.assem(edof_int[j,:],K,Ke)

                a,r = cfc.solveq(K,f,np.int_(bc),bcVal)
                ed= cfc.extract_eldisp(edof_int,a)
```

```
    edoft = []
    for k in range(len(edof)):
        if i > 0:
            previous = len(edoft)
        if k <= len(edofh):
            es,edi,eci = cfc.beam2s(ex[k,:], ey[k,:], epbh, ed[k,:])
            tension = es[:,0]/epbh[1] + (es[:,2]*1/2*t)/Ibh
        if k > len(edofh) and k <= len(edofb):
            es,edi,eci = cfc.beam2s(ex[k,:], ey[k,:], epbv, ed[k,:])
            tension = es[:,0]/epbv[1] + (es[:,2]*1/2*t)/Ibv
        if k > len(edofb) and k <= q :
            es,edi,eci = cfc.beam2s(ex[k,:], ey[k,:], epfd, ed[k,:])
            tension = es[:,0]/epfd[1] + (es[:,2]*1/2*t)/Ifd
        else:
            es,edi,eci = cfc.beam2s(ex[k,:], ey[k,:], epfv, ed[k,:])
            tension = es[:,0]/epfv[1] + (es[:,2]*1/2*t)/Ifv

        if tension[0] > 2.8*10**3:
            edoft.append(np.array(edof[k]))
        if i == 0:
            if len(edoft)>= 1:
                ex3,ey3= cfc.coordxtr(edoft,coord,dof)
                cfvm.figure()
                cfvm.eldraw2(ex,ey,[1,1,0])
                cfvm.eldraw2(ex3,ey3,[2,4,0])
                cfvm.showAndWait()
        if i > 0:
            if len(edoft)>= 1:
                if len(edoft)> before:
                    if len(edoft) > previous:
                        if (k % 1 == 0):
                            ex3,ey3= cfc.coordxtr(edoft,coord,dof)
                            cfvm.figure()
                            cfvm.eldraw2(ex,ey,[1,1,0])
                            cfvm.eldraw2(ex3,ey3,[2,4,0])
                            cfvm.showAndWait()
```

Models with combination

```
P= 39

done = np.zeros((len(edof),1))
for i in range(P):
    if i > 0:
        before = len(edoft)
    f[3*p1-2]= -1/11*i
    f[3*p2-2]= -10/11*i

    a,r = cfc.solveq(K,f,np.int_(bc),bcVal)
    ed= cfc.extract_eldisp(edof_int,a)
    for j in range(len(edof)):
        if j <= len(edofh):
            es,edi,eci = cfc.beam2s(ex[j,:], ey[j,:], epbh, ed[j,:])
            tension = es[:,0]/epbh[1] + (es[:,2]*1/2*t)/Ibh
        if j > len(edofh) and j <= len(edofb):
            es,edi,eci = cfc.beam2s(ex[j,:], ey[j,:], epbv, ed[j,:])
            tension = es[:,0]/epbv[1] + (es[:,2]*1/2*t)/Ibv

        q = len(edofb)+ len(edoffh)
        r = len(edofb)+ len(edoffh) + len(edoff0) + len(edoff1) + len(edoff2)
        if j > len(edofb) and j <= q :
            es,edi,eci = cfc.beam2s(ex[j,:], ey[j,:], epfh, ed[j,:])
            tension = es[:,0]/epfh[1] + (es[:,2]*1/2*t)/Ifh
        if j > q + len(edoffv) and j <= r :
            es,edi,eci = cfc.beam2s(ex[j,:], ey[j,:], epfd, ed[j,:])
            tension = es[:,0]/epfd[1] + (es[:,2]*1/2*t)/Ifd
        else:
            es,edi,eci = cfc.beam2s(ex[j,:], ey[j,:], epfv, ed[j,:])
            tension = es[:,0]/epfv[1] + (es[:,2]*1/2*t)/Ifv

        if tension[0] > 2.8*10**3:
            done[j]+=1
            c = 0.00000000000001
            if done[j] < 1:
                if j <= len(edofh):
                    Ke = -cfc.beam2e(ex[j,:],ey[j,:],epbh) + c
                if j > len(edofh) and j <= len(edofb):
                    Ke = -cfc.beam2e(ex[j,:],ey[j,:],epbv) + c

                q = len(edofb)+ len(edoffh)
                r = len(edofb)+ len(edoffh) + len(edoffv) + len(edoff0) + len(edoff1) + len(edoff2)

                if j > len(edofb) and j <= q :
                    Ke = -cfc.beam2e(ex[j,:],ey[j,:],epfh) + c
                if j > q + len(edoffv) and j <= r :
                    Ke = -cfc.beam2e(ex[j,:],ey[j,:],epfd) + c
                else:
                    Ke = -cfc.beam2e(ex[j,:],ey[j,:],epfv) + c
                K = cfc.assem(edof_int[j,:],K,Ke)

                a,r = cfc.solveq(K,f,np.int_(bc),bcVal)
                ed= cfc.extract_eldisp(edof_int,a)
```

```
    edoft = []
    for k in range(len(edof)):
        if i > 0:
            previous = len(edoft)
        if k <= len(edofh):
            es,edi,eci = cfc.beam2s(ex[k,:], ey[k,:], epbh, ed[k,:])
            tension = es[:,0]/epbh[1] + (es[:,2]*1/2*t)/Ibh
        if k > len(edofh) and k <= len(edofb):
            es,edi,eci = cfc.beam2s(ex[k,:], ey[k,:], epbv, ed[k,:])
            tension = es[:,0]/epbv[1] + (es[:,2]*1/2*t)/Ibv

        q = len(edofb)+ len(edoffh)
        r = len(edofb)+ len(edoffh) + len(edoffv) + len(edoff0) + len(edoff1) + len(edoff2)

        if k > len(edofb) and k <= q :
            es,edi,eci = cfc.beam2s(ex[k,:], ey[k,:], epfh, ed[k,:])
            tension = es[:,0]/epfh[1] + (es[:,2]*1/2*t)/Ifh
        if k > q + len(edoffv) and k <= r :
            es,edi,eci = cfc.beam2s(ex[k,:], ey[k,:], epfd, ed[k,:])
            tension = es[:,0]/epfd[1] + (es[:,2]*1/2*t)/Ifd
        else:
            es,edi,eci = cfc.beam2s(ex[k,:], ey[k,:], epfv, ed[k,:])
            tension = es[:,0]/epfv[1] + (es[:,2]*1/2*t)/Ifv
        if tension[0] > 2.8*10**3:
            edoft.append(np.array(edof[k]))
        if i == 0:
            if len(edoft)>= 1:
                ex3,ey3= cfc.coordxtr(edoft,coord,dof)
                cfvm.figure()
                cfvm.eldraw2(ex,ey,[1,1,0])
                cfvm.eldraw2(ex3,ey3,[2,4,0])
                cfvm.showAndWait()
        if i > 0:
            if len(edoft)>= 1:
                if len(edoft)> before:
                    if len(edoft) > previous:
                        if (k % 1 == 0):
                            ex3,ey3= cfc.coordxtr(edoft,coord,dof)
                            cfvm.figure()
                            cfvm.eldraw2(ex,ey,[1,1,0])
                            cfvm.eldraw2(ex3,ey3,[2,4,0])
                            cfvm.showAndWait()
```

Appendix C

Structure

In this section, a picture of the rod models are given. It is to get an idea which models were made.

Horizontal rods

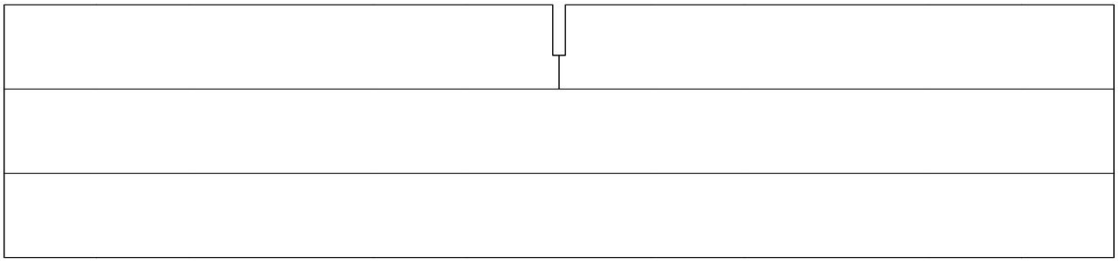

Figure C.1: Model with horizontal rods

Horizontal and vertical rods

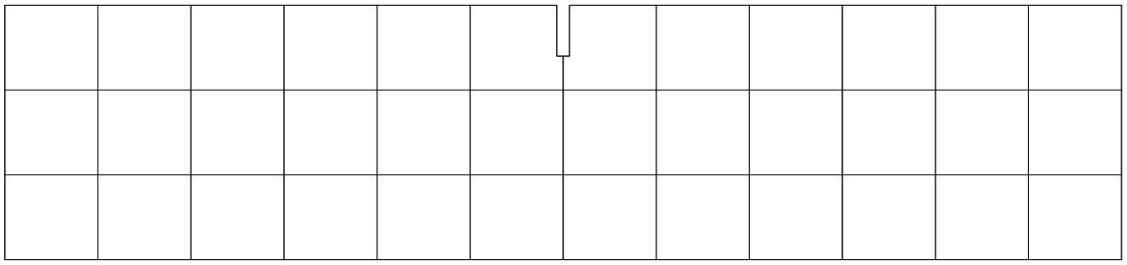

Figure C.2: Model with horizontal and vertical rods

Diagonals

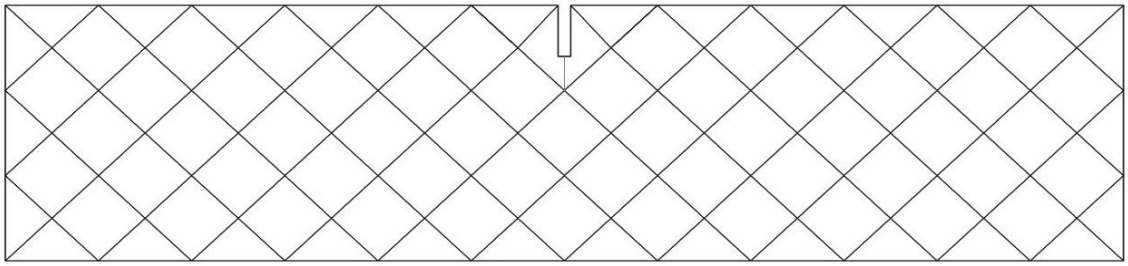

Figure C.3: Model with diagonals

Combination with diagonals to the left

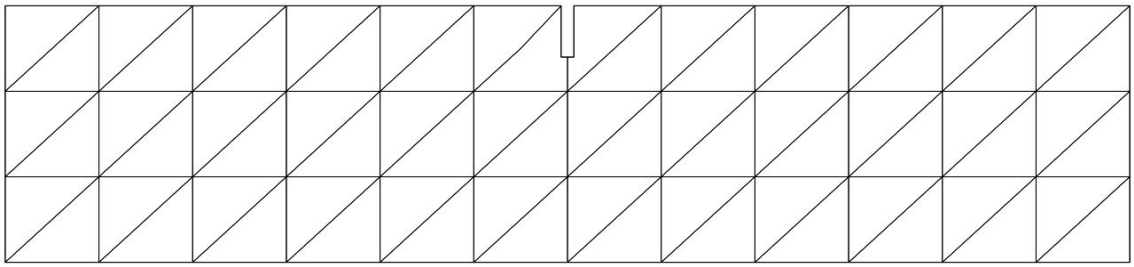

Figure C.4: Combination model with diagonals to the left

Combination with diagonals to the right

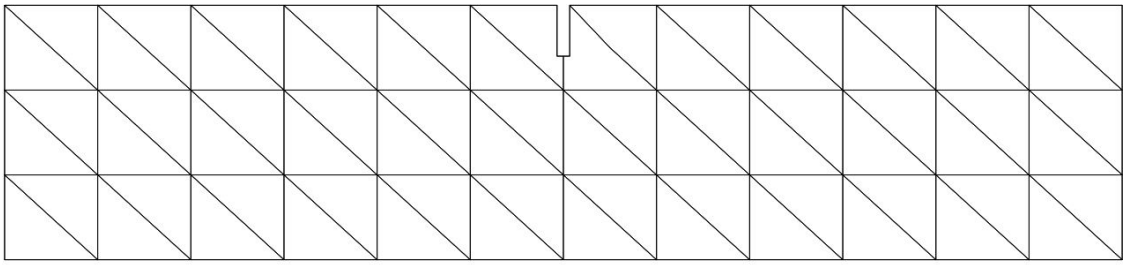

Figure C.5: Combination model with diagonals to the right

Combination with diagonals in both directions

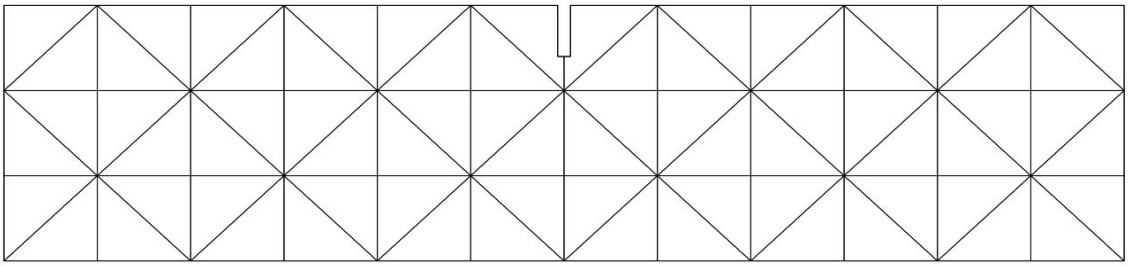

Figure C.6: Combination model with diagonals in both directions

Appendix D

Variables analysis

In this section, a variables analysis is given for each model. The results from the models are shown and explained. Recommendations are given based on the influences of the variables. This is the basis for Chapter 3.

D.1 Model with horizontal rods

The first model contains horizontal rods. When the variables are adjusted, the results are changed. This differs for each variable. When the amount of elements is adjusted in the height direction, it can be seen in figure D.1 that the cracks become larger. Thus, more rods will reach the tensile strength. The differences are especially visible at the top of the beam: as the amount of elements increases, the red-colored area grows around the notch. Because it is not clear whether an increase of elements in the height direction improves the model, there is no advice regarding the amount of elements in the height direction. For the amount of elements in the longitudinal direction, there is also no recommendation, since no change in the results was seen. The results continue to display approximately the same. When both are increased, little difference is seen in the plots. This is to be expected since adjusting both individually produces little differences in the results. Changing the thicknesses of the rods does affect the results. When the thicknesses of the rods are made smaller, it can be seen that an entire section of the beam turns red. It is recommended to keep the thickness at 0.1 mm, to reduce the red area in the model.

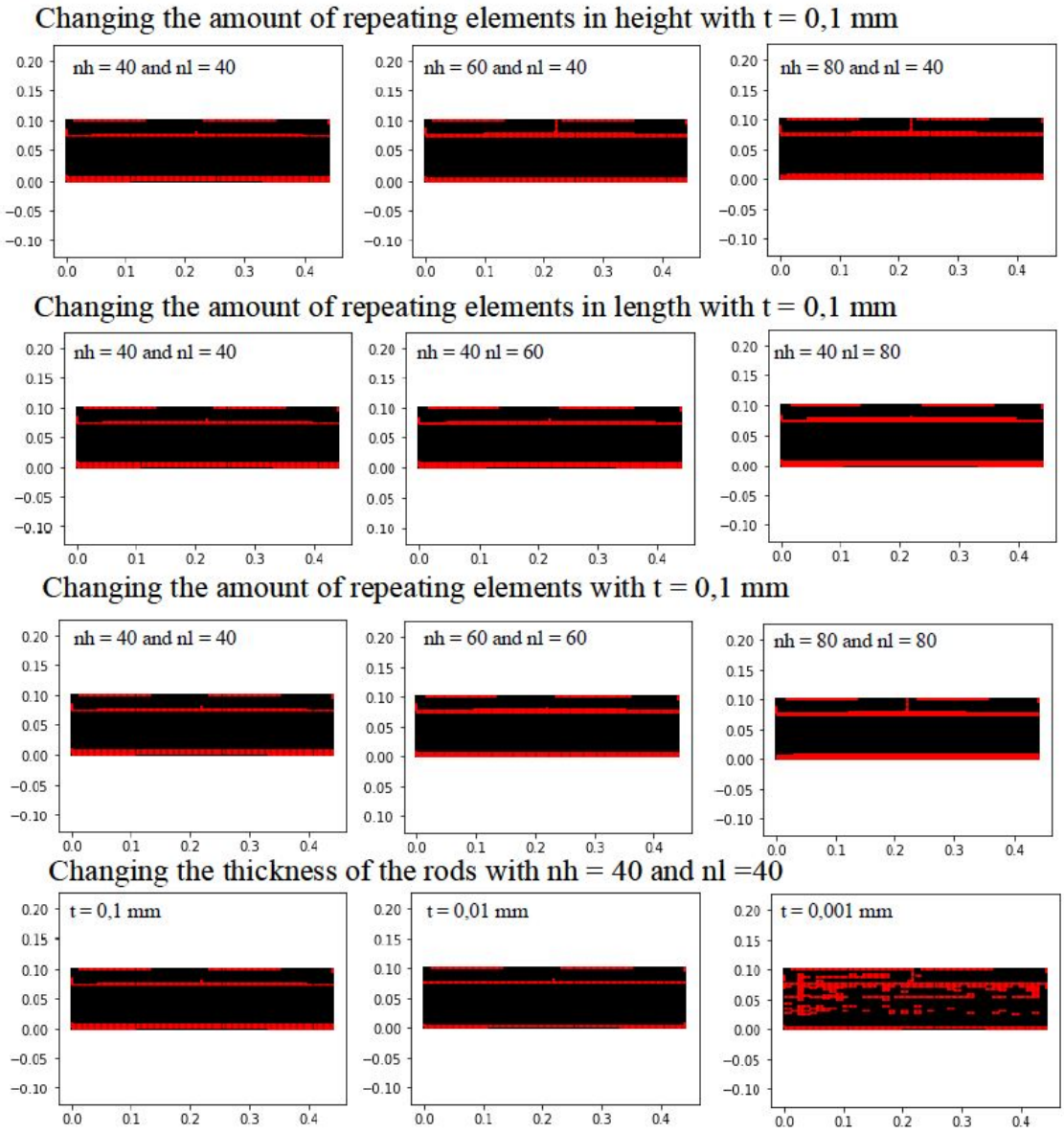

Figure D.1: The variable analysis of the model with horizontal rods

D.2 Model with horizontal and vertical rods

When adding vertical rods in to the model with horizontal rods, other results are obtained. The result are schown in figure D.2. When the amount of repeating elements is increased in height, it can be seen that the red area becomes larger and wider. The area at the top moves to the right, while at the bottom it moves more to the left. Since this makes the model less representative, it is recommended to keep the amount of elements in the height low. When the amount of elements is changed in length, this is also recommended. Increasing the elements in the longitudinal direction increases the red area right below and the area left of the notch. When the total amount of repeating elements is increased, the red area becomes a combination of the red areas plotted before. The thickness of the rods give little influence in this model.

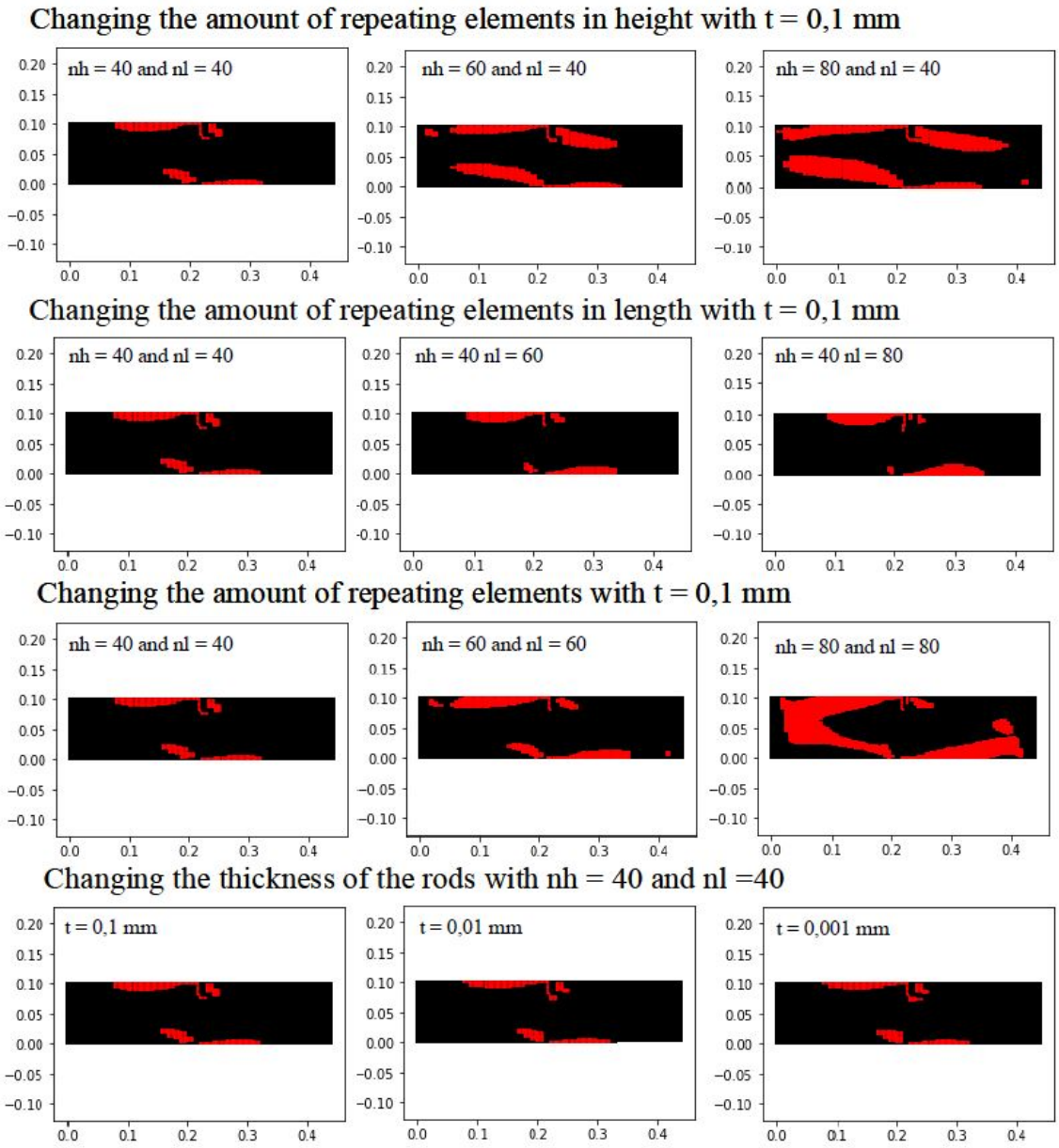

Figure D.2: The variable analysis of the model with horizontal and vertical rods

D.3 Model with diagonals

With a model consisting of diagonals, the results of figure D.3 were found. Compared to the other two models, a difference can be seen. The model is weaker: it has more red than black areas. The variables must be adjusted to get a better result. When changing the amount of repeating elements in the height direction, it is recommended to reduce the amount of elements: With more elements in height, the red area becomes larger as seen in the figure. When the amount of elements in the length direction is increased, an improvement can be seen. The red area becomes more concentrated in the middle of the beam. When nl contains the value 80, there is hardly any cracking. Since the structure variables have different influences on the results, it is interesting to look at the increase of both variables. When both structure variables are chosen to be set high, it can be seen that the red area plotted does not reduce: no improvement in the system can be seen. It is therefore advisable to keep the difference in elements in the height and length direction high. When the thickness of the bars is reduced, it can be seen that that the red area not changes. Thus, there is no advice regarding changing the properties of the rods.

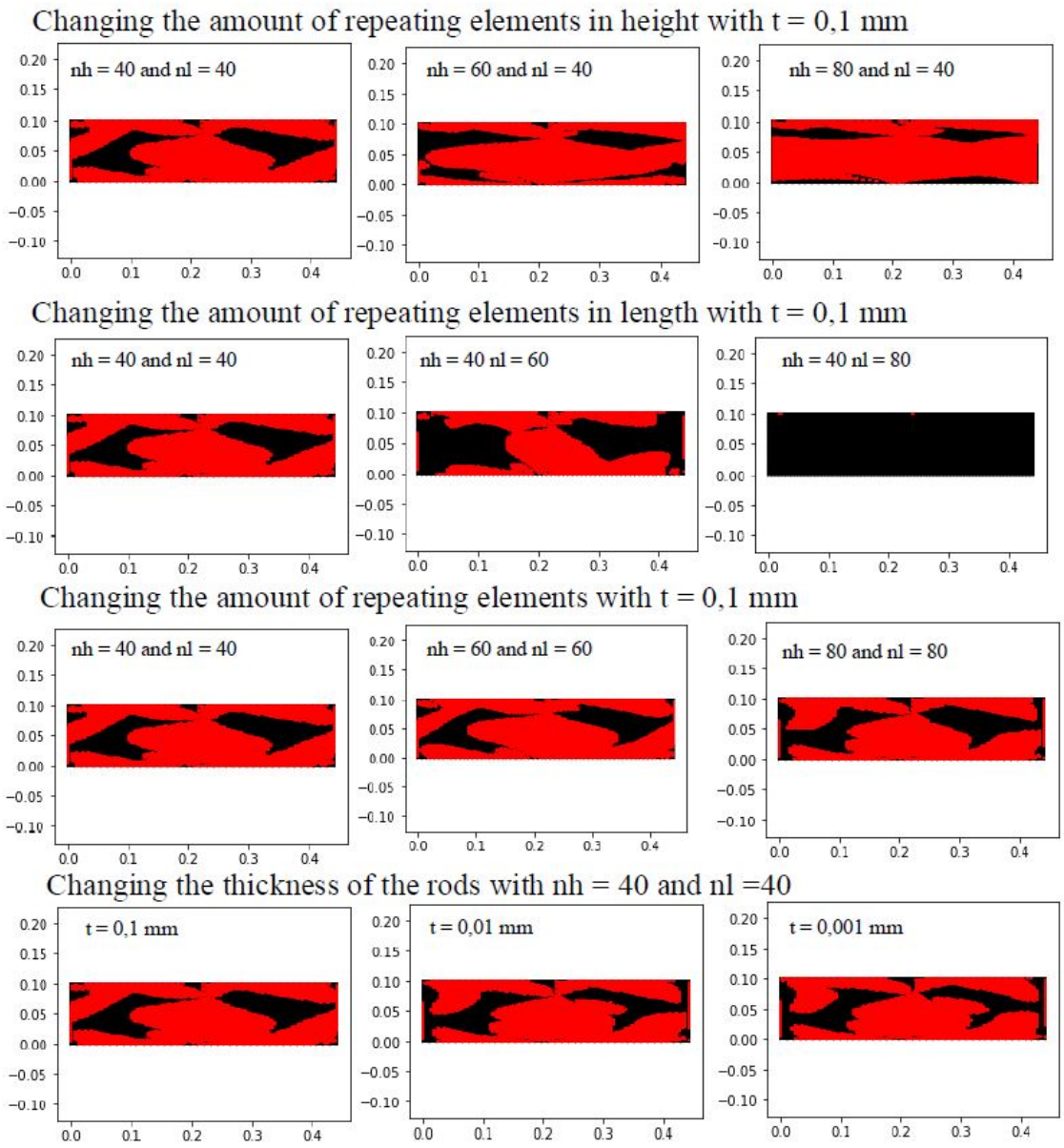

Figure D.3: The variable analysis of the model with diagonals

D.3.1 Combination with diagonals to the left

When a combination is made with diagonals in the left direction it can be seen in figure D.4 that the the red area takes place more around the notch. Still, the variables give influences on the sizes of the red area found. When the amount of rods is changed in the height direction, the red area becomes smaller. At the bottom, the crack disappears. However, when the amount of repeating elements in the length direction are changed, more of the mechanism collapses. The cracks are growing towards each other. To see if a better picture is obtained as the amount of elements in both the length and height directions is increased, plots of these were also made. The crack is a combination of the two plots: the crack is smaller, however, in the same direction as the previous plots made.It is recommended to keep the amount of elements in the height high and to keep the amount of elements in length direction between the values 50-70. When the thickness of the bars is reduced, it can be seen that that the red area not changes. Thus, there is no advice regarding changing the thickness of the rods.

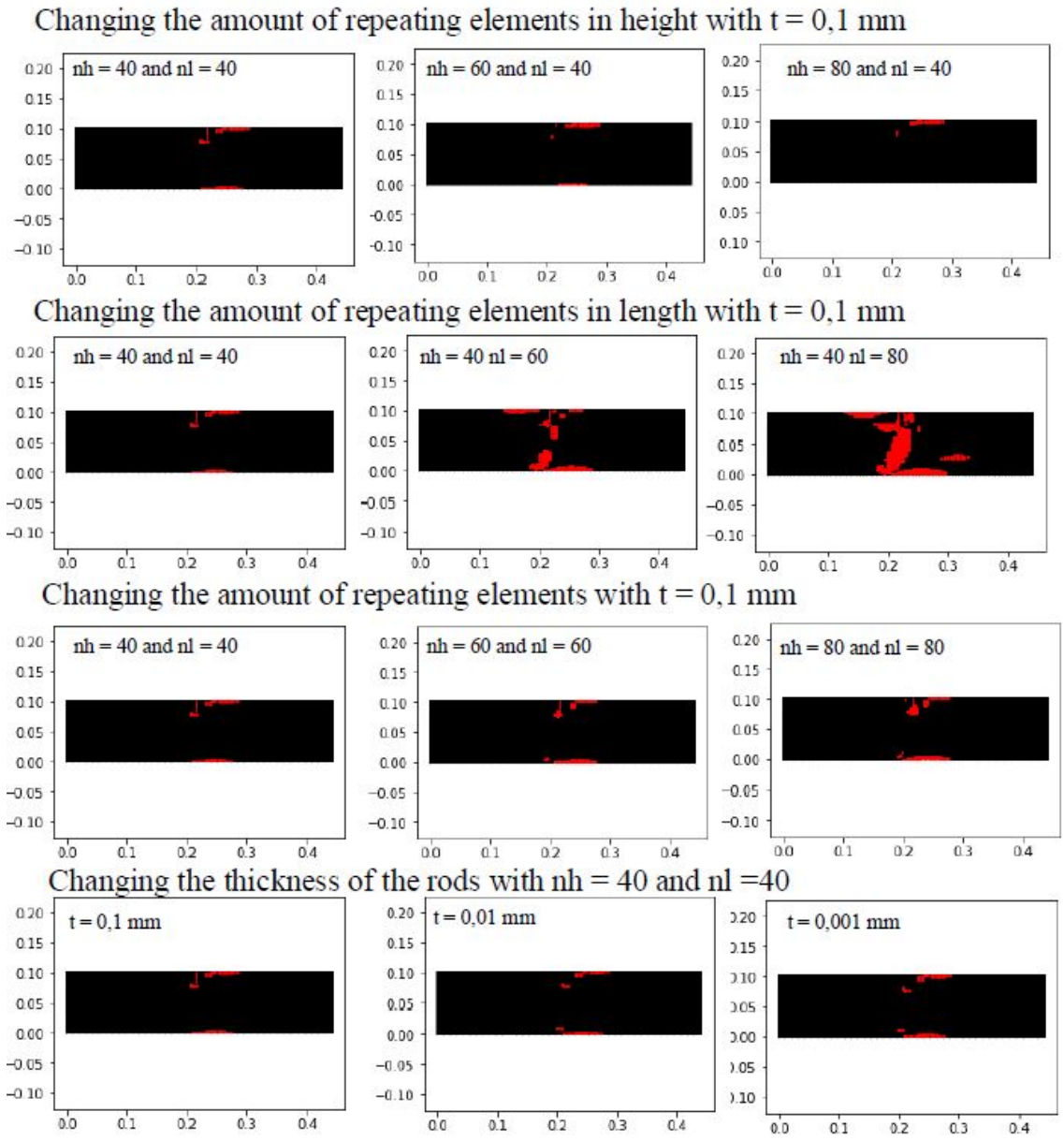

Figure D.4: The variable analysis of a combination model

D.3.2 Combination with diagonals to the right

The first thing that is striking about the results obtained in figure D.5 is the difference with the other combination model: when the diagonals are in the right direction, it can be seen that the model behaves weaker. When a variables analysis is done, it can be seen though that the variables have the same influences on both systems. If the elements are increased in the height direction, it can be seen that the red area shrinks. If the amount of elements in the length direction is increased, it can be seen that the crack develops further towards each other. The crack becomes larger. When the total amount of elements in the system are increased, it can be seen that the red area grows around the notch. Thus, it is recommended to increase the amount of elements in the height direction and keep the amount of elements in the length direction slightly lower than the elements in the height direction. When the thicknesses of the rods are made smaller, it can be seen that the crack in the middle of the beam start to grow. It is recommended to keep the thickness at 0.1 mm, to reduce the red area in the model.

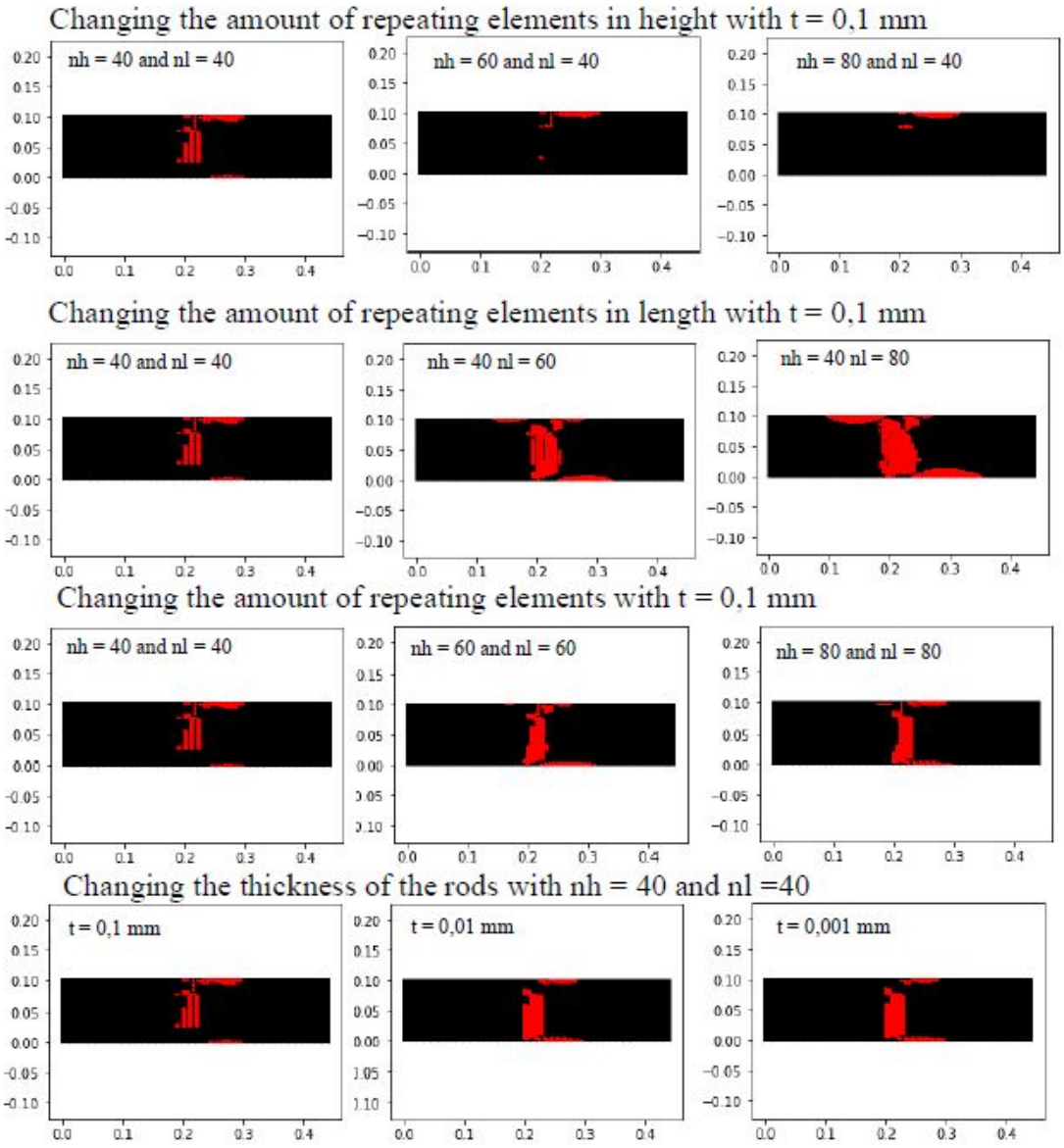

Figure D.5: The variable analysis of a combination model

D.3.3 Combination with diagonals in both directions

Using the latter model, the results in figure D.6 were found. This model had four different orientations for the rods. It is a combination of the two previous models. As variables are changed, the same trend is seen as with the previous two combinations. When the amount of elements in the height direction is increased, it can be seen that the crack retreats more towards the notch. When the amount of elements is changed in the length direction, it can be seen that the crack develops more. Compared to the two previous models, the shape is more representative: fewer cracks develop in the area around the notch. For the combination model it is recommended to set the amount of elements in height direction high and the amount in length direction not too high. Because thickness has little effect on the change in results, no advice is given.

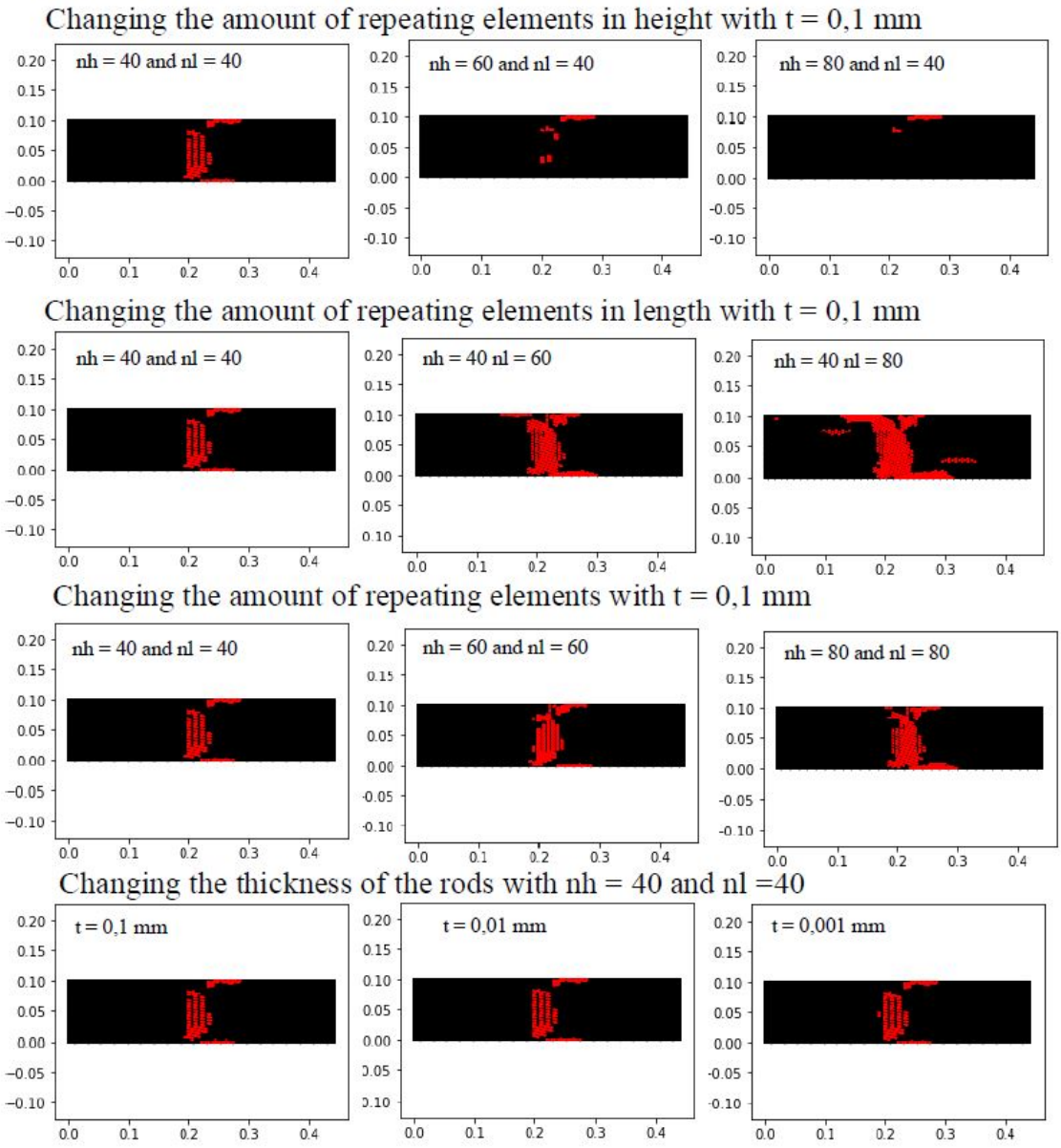

Figure D.6: The variable analysis of a combination model

References

[1] *Alsam project aluminium lattice structures: Am additive manufacturing metal powder bed fusion*. [Online]. Available: https://resources.renishaw.com/details/ALSAM+project+aluminium+lattice+structures(254282)(93788) (cit. on p. 2).

[2] T. C. Gasser and G. A. Holzapfel, 'Modeling 3d crack propagation in unreinforced concrete using pufem,' *Computer Methods in Applied Mechanics and Engineering*, vol. 194, pp. 2859–2896, 25-26 Jul. 2005, ISSN: 00457825. DOI: 10.1016/j.cma.2004.07.025 (cit. on pp. 1, 3, 4).

[3] C. Hartsuijker, *Spanningen, vervormingen, verplaatsingen*. Academic Service, 2000 (cit. on pp. 21, 22).

[4] L. J. Malvar and G. E. Warren, 'Dtic f;le copy o e l mixed mode crack propagation in concrete ltn,' 1990 (cit. on p. 5).