Delft University of Technology
Faculty of Civil Engineering
CTB3000 - Bachelor Final Project

Stochastic & Statistical Analysis of Shape Imperfections in Shell Structures

BEP Final Report

Berend Schuit - 4415523

B.J.Schuit-1@student.tudelft.nl

06-46196699

January 22, 2018

Supervisor 1	Dr. ir. P. C. J. Hoogenboom
Supervisor 2	ir. A. Roy

Preface

This report is written for the course CTB3000 'Bachelor Eindproject'. This project is the final project of the Bachelor Civil Engineering. This project is aimed to combine large parts of the skills and knowledge acquired during the Bachelor program. The project integrates the research skills, technical knowledge and taught mathematical and physical problem solving techniques. This final bachelor project enables a student to conduct autonomous research and contribute something small to the research section which issued the project.

This project was conducted for the section Structural Engineering of the Faculty of Civil Engineering of Delft University of Technology. The research theme is 'Shape-imperfections in shell structures' and this project continues on previous research [1] conducted by professors of this department. Last year two other bachelor students also worked on a project on this subject. This project has the same goal, but takes a different approach.
The project was provided by my primary supervisor Dr. ir. Hoogenboom. The exact research subject and the approach was formulated together with Dr. ir. Hoogenboom.

I would like to thank Dr. ir. Hoogenboom for his time and guidance throughout the project. The guidance during the meetings was very constructive and enthusiastic which really helped a lot. I would also like to thank my secondary supervisor ir. Roy for his insights and advice on the statistical part of the project. Finally I would like to thank all people [1] [3] [9] who conducted the research which enabled this small part of a larger whole to be researched by me.

I hope you find this report enjoyable and informative to read.

Summary

First an introduction to the subject is given along with the target statement, the made assumptions and the proposed approach in the first section. Next as the starting point of this project the existing field generator is evaluated and a brief literature study is conducted. The results of this are described in section 2.

In section 3 the process of creating a new field generator from scratch is described. In contradiction to the old field generator, in this version the process parameters will be variable, therefore it will be able to research their behavior and optimize them, this will be done in a later section. The field generator is made based on the principle of fourier series in 2d. The parameters, equations and the code of the field generator itself is explained and example output is generated to test whether the field generator functions as expected.

In the next section the output of this new field generator is compared to the scandata included in the paper describing the research conducted by Elferink et al. [1]. The output of the simulations made using the field generator is compared to the plots of the real scandata. It turns out it is not possible to generate random fields that look exactly like the patterns seen in the scandata with this field generator. Different alterations to the field generator are explored, but these have no mathematical or physical foundation and don't look more like the scandata. The question whether it is logical to expect a certain pattern from a randomly generated field is discussed and the conclusion is drawn it is best to continue with the initial new version of the field generator.

In section 5 the process parameters are researched and optimized. We take a look into each process parameter 'ceteris paribus' and we look at them as a whole. This provides insight in their individual behavior and their combined behavior. Many simulations are ran and the data is stored and analyzed thoroughly. It is attempted to find an analytical solution linking the runtime, the values of the process parameters and the accuracy of the simulation to each other. Unfortunately this doesn't succeed, mainly because two of the three process parameters don't converge nicely because of some remaining noise. In the end optimal values for the process parameters are found and those are tested for different input. The simulation runtime is shorter then expected and therefore it is tested whether increasing the process parameters contributes to the accuracy of the simulations. It turns out this doesn't help much and is not worth the extra runtime. The found optimal values for the process parameters will be used in the statistical analysis and will be used as advised values in the final program.

In section 6 different statistical analysis are conducted on the data generated by the several simulations. A way to efficiently determine the characteristic value of datasets is developed and research into the partial factor is conducted. A start is made to compare the datasets to different distributions but this is abandoned early. It is a requirement that the output values can be used for FEM calculations in combination with snowloads (which are normally distributed) the used partial factor formula is only valid when the distributions of all loads are the normal distribution. Therefor we focus on the normal distribution. The fit of the normal distribution to a large range of datasets is tested and it can be concluded the normal distribution fits well enough for this practical application.

In the final sections the program which is made using all conclusions, knowledge and code of the previous chapters is discussed. The way this program works is described and some images showing the way it works along with a video showing the usage are shown. At last a chapter with all conclusions and a chapter with all recommendations are provided at the end of this report.

Contents

1 Introduction

1.1 Introduction to the subject

One of the major challenges in the design process of concrete shell-structures is the fact that small shape imperfections have a large impact on the overall strength of the shell-structure. Buckling is the most relevant failure-mechanism for concrete shell-structures. Shells are generally designed by the following process: FEM-calculations are performed based on a perfectly shaped shell without imperfections, the FEM-calculation returns the most-likely to occur buckling shape, and buckling length. It is known that buckling tends to occur on locations where this buckling shape is already a little bit present due to an imperfection. This initial imperfection determines the strength of the shell-structure.

Before construction it is not possible to know which imperfections will occur and where they will occur in the concrete shell-structure. Therefore it is not possible to take the imperfections accurately into account during the design process. This means it is necessary to hugely over-design the structure to be sure the structure is safe enough. If a method would exist for calculating an accurate estimate for the characteristic value of the parameters determining this strength this could be taken into consideration during the design process which would result in a less over-designed structure.

A team of researchers from the TU Delft made scans of four existing shell-structures in order to get an insight into the imperfections of those existing shell-structures [1]. Further research was conducted on this data by this team. A Master's Thesis[1] was written on this subject and the acquiring of the data was part of this research.

The imperfections are characterised by the length of the deflection and the largest deflection in the middle of the length. The imperfections occur in a somewhat random way all over the surface, therefore they can be modeled with a stochastic field [5]. In previous research two other bachelor students [9] [3] looked into this project and tried to come up with methods to find the best fitting statistical distribution, which would later be used to find the characteristic value for the buckling load of shell-structures based on the distribution-type. A lot of effort has been put into this, but at this moment a method to accurately estimate the statistical distribution hasn't been found yet. The data doesn't seem to fit particularly well to one of the researched distributions.
A new approach will be used in this project in contrary to the previous research. In this project the goal will not be to find a mathematical and statistical description which describes the problem for all input-values, but a program will be created which evaluates each different set of input-variables independently and returns the desired output very accurately for that particular case.

1.2 Target Statement

The goal of this project is to make the previously conducted research applicable to a wider range of input variables and to take the final step into finding a way to determine the characteristic value for the shape imperfections. Not by finding a nicely fitting statistical distribution but by making a program which runs the Monte-Carlo simulations for given input and then provides the desired output for that particular case. When this succeeds, this method could be turned into a program with a Graphical User Interface. This program could be used by Structural Engineers to find the characteristic value and partial factor of deflections when designing a shell, without the need to understand all underlaying theory and processes.

The target of this research project is:
To combine the knowledge gained by the previous research with new research to create a program which accurately calculates the characteristic value and partial factor of the largest shape imperfection in random fields for a large variety of input. The ultimate goal is to turn this program into a user-friendly and efficient GUI which requires minimal input and delivers accurate output.

The subsection about the Approach will elaborate in more detail how this goal will be achieved.

1.3 Assumptions & Boundaries

In the previously conducted research on this issue some assumptions were made to make it possible to isolate the shape-deflection part of the problem and come up with a "ceteris-paribus" solution for this isolated problem of shape imperfections. In this research we will continue with the assumptions made during the previous research.
Only the shape-imperfections will be taken into account. Other imperfections such as residue-stresses, temperature-stresses, inhomogeneous-material, shrinkage and eccentricity will be neglected in order to be able to come up with a pure "ceteris-paribus" solution for the shape-imperfections problem.
Because this method uses data supplied by measurements of several already constructed shell-structures the results will only be applicable for a shell-structure constructed using the same construction method. Though the goal is to create a program which will also work for new situations when given different input.

1.4 The role of the program within the design process

In figure 1 the role of the program within the design process of concrete shell structures is shown. A larger version of this image is added in Appendix E. The aim of the program is to provide a well founded assumption for the largest occurring shape imperfection in the to be designed shell. This value is one of the required input of the FEM calculations. Nowadays this value is just estimated and usually hugely over-estimated to be able to be on the safe side.

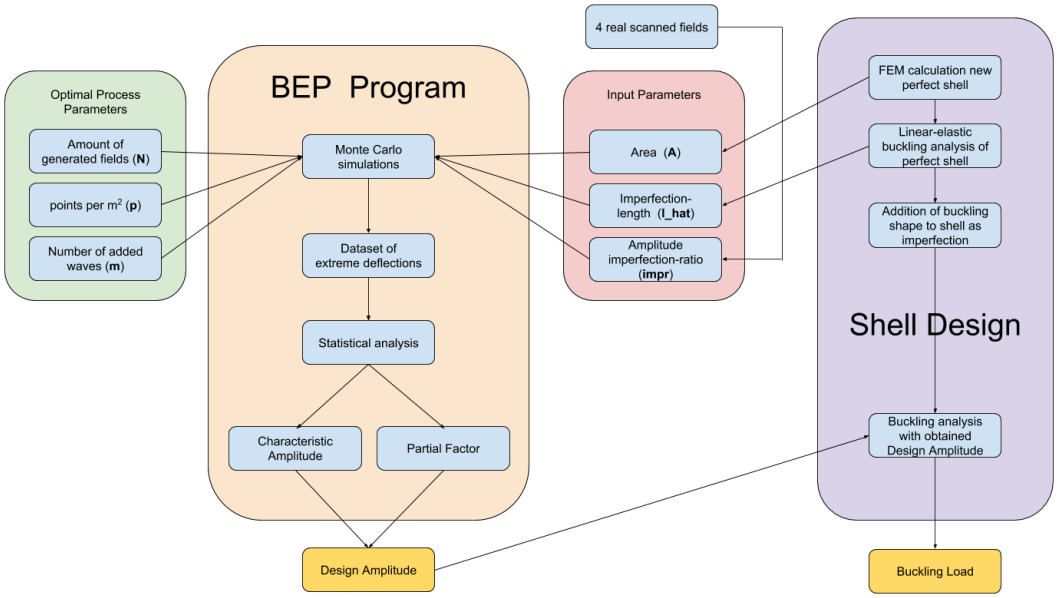

Figure 1: The role of the program within the design process of shell structures.

Two of the three input variables result from the early stages of the FEM analysis. The area and imperfection length are supplied to the program as input. The amplitude imperfection ratio is deduced from the real scandata. With these physical inputs the program runs the Monte Carlo simulations and returns a characteristic value and partial factor for the largest shape imperfection. This value is used as input in the FEM simulation for the shell design instead of a large, unfounded and over-estimated value. The process parameters hold no relation to the physical input, they just influence the internal computation process.

1.5 Approach

The target of this research project will be divided into a few smaller parts in order to be able to have a better overview of the whole project. Some of these parts will result in written chapters in this report, others will not, such as researching the subject with a literature study or creating the final GUI program. The tools or methods used for these applications will also be described below.

The project will be approached in the following way:

- First a literature study will be conducted. Particularly into the previously conducted research and the scandata, but also into new literature.
- Next the already existing Field Generator will be evaluated.
  In this version of the Field Generator the process parameters determining the accuracy and runtime of the program are fixed. These will be made variable in order to have more control over the accuracy of the program.
- When a broad understanding about the subject and the existing program is gained, the program will be adjusted or a new version will be created from scratch.
- When the program is finished it will be thoroughly tested for a wide variety of input-parameters and it will be attempted to recreate the scandata when using values for the input-parameters similar to four shell-structures existing in the real world.
- In contrast to the previous version of the program, the new version will have variable process-parameters which will allow for changes in the internal functioning of the program in order to be able to influence the accuracy and the runtime of the program. These process parameters will also be optimized.
- For the remaining part of the project these optimal values for the process parameters will be used. A lot of different simulations will be ran and a statistical analysis will be conducted on the output in order to find a way to accurately determine the characteristic value and partial factor for the supplied input values.
- Finally all of the preceding code will be combined into a single program in which the user can input values and receive the desired output. When possible this program should feature plots of the internal processes to explain the output.

2 Evaluation of the existing Field Generator

At the start of the project a field generator created by Gijzenberg [3] was available. This field generator was reviewed and studied. This field generator had static process parameters, one of the goals of the project was to make these process parameters variable in order to give the user more influence over the accuracy, the runtime and the way the program functions. Also possibly an error was spotted in the old field generator, but this was never proven nor further investigated because a new version would be created anyway.

Different attempts at updating the existing Field Generator and making the parameters variable were conducted but none were successful. In order to get a better understanding of the way the field generator functions it was adapted to a 1d-version to get more insight in the way the different parameters influence the process of creating a stochastic field. After a while it was decided to start from scratch instead of trying to update the existing version. This proved to be a good strategy. In order to be able to create a new field generator more research was conducted into the subject and different literature was studied. Also some content of Master's courses of the faculties Mathematics and Civil Engineering was studied to acquire the required knowledge for this project.

In the paper about the scanned shell structures[1] it is stated that more research has been conducted on the imperfections of rockets and airplane parts. Therefore some reading into these subjects is also conducted. The studied literature includes:

A report on the process of designing shell-structures using Finite Element Methods to get a grasp of the larger-scale process this project is part of. [4]
A report on the acquisition of the measurement data from the existing shell-structures and a part of the research done on the topic by the research team. [1]
A reader on stochastic vibrations to get a better view on the approach of using stochastic processes on engineering problems. These are lecture notes from a Master's course, therefore the author was not yet familiar with the information. [10]
A reader on buckling in pipes, shells and other constructions. This was read to get a better grasp on the failing mechanism called buckling. [7]
A paper on a stochastic stability analysis of steel tubes with random properties. [11]
A paper on a buckling analysis of imperfect shells with stochastic properties. [8]
Multiple conversations with Mr Hoogenboom on the subject. [6]

The course Probabilistic Design and statistics in Civil Engineering.
The course Monte-Carlo Simulations 1.

Based on the acquired knowledge by studying the existing field generator and the reports of the predecessors in combination with the insights gained by the literature study a new field generator was created. The new field generator is explained in the following section.

3 New Field Generator

The Field Generator is the core part of the program. The field generator simulates the physical problem of shape imperfections which is researched in this project. It will also be the core of the GUI program. All statistical analysis which will be made will be made on data generated by this field generator program. First the different parameters which are the input of the program will be discussed, there are two types of parameters. Next the code will be shown and the underlying formula's and their meaning will be discussed. Next a particular tricky part of the code will be elaborated in more detail. The last part of this section will show example output and some observations based on this output will be discussed.

3.1 Parameters

The parameters of the Field Generator can be divided into two types. Input Variables and Process Parameters. The Input Variables are input-values supplied by the user which represent the physical properties of the shell structure. The Process Parameters are parameters that determine the internal accuracy and runtime of the calculations.

The Input Variables are:

- **A** - Area $[m^2]$
- **l_hat** - Buckling length [m]
- **impr** - Amplitude Imperfection Ratio [-]
- **s** - number of scans of input data [-]

The Process Parameters are:

- **p** - points per m^2 $[1/m^2]$
- **m** - number of added waves [-]
- **N** - number of fields generated [-]

The term Process Parameters and Input Variables will be used many times in this report.

In contradiction to the previous field generator, the process parameters are variable in this field generator. For the final program values for the process parameters which were found to be the optimum will be advised, but ofcourse the user can deviate from these advised values. In order to be able to optimize these process parameters these parameters need to variable in order to be able to iterate over them.

The Input Variables represent the physical properties of the shell which is being designed. The Area is trivial and needs no further explanation. The Buckling Length is the largest possible length of an imperfection. The Amplitude Imperfection Ratio is the ratio between the length and deflection of an imperfection. m is the number of sine waves 'added on top of each other'. The simulations are made based on fourrier series. The number of scans of input data is also relevant for the statistical interpretation and accuracy of the input data. For this application 4 scans were made, thus the spectrum only exists of 4 datasets. This is not that many and therefore there is a large uncertainty which has to be implemented in the amplitudes of the stochastic field. These two Input Variables will be further discussed in a separate subsection, because they are at this moment not relevant for the functioning of the code. First we will look into the way the code functions, and later on we will look deeper into the Amplitude Imperfection Ratio. The Amplitude Imperfection Ratio was implemented as a fixed input value at first, but this turned out to be incorrect as Mr Hoogenboom pointed out.[6] This will be discussed in more detail in section 3.3.

3.2 The Field Generator Code

```
import numpy as np
import numpy.random as rnd


# d = deflection in field
# A = area
# l_hat = buckling length
# impr = Amplitude Imperfection Ratio = np.sqrt(V) / l_hat
# V = variance of measured data
# s = number of scans
# p = points per m2
# m = number of added waves
# N = number of fields generated


def generate_field(A, l_hat, impr, s, p, m, N, field_output_num):

    srA = int(np.sqrt(A))
    srp = int(np.sqrt(p))
    field = np.zeros((srA*srp, srA*srp))
    X_axis = np.linspace(0, (srA-(1/srp)), (srA*srp))
    Y_axis = np.transpose([X_axis])

    C = np.sqrt( (12 / (m - 1 / (s * m)) ) ) * impr

    for i in range(1, m+1):
        l = ((2 * i - 1 ) / (2 * m)) * l_hat
        a = C * l

        d = a * np.sin( (np.pi * X_axis / l) + 2 * np.pi * rnd.rand()) * \
                    np.sin( (np.pi * Y_axis / l) + 2 * np.pi * rnd.rand())
        field = field + d

    if field_output_num == 0:
        return field
    if field_output_num == 1:
        if abs(np.amax(field)) < abs(np.amin(field)):
            return abs(np.amin(field))
        else:
            return abs(np.amax(field))
```

The Field Generator Code looks quite compact and simple, but a lot happens within this single block of code. In Appendix A a Program Structure Diagram describing the process going on within the code is included. The Field Generator can be used to either output a Field with the summation of all waves, or it can be used to output the single maximum absolute value of the field. This value (of N fields) is what we are interested in for the further statistical analysis. This functionality to choose an output is introduced by the extra parameter field_output_num.

$$d = \sum_{i=1}^{m} a_i \sin(\frac{\pi * X}{l_i} + \zeta_i) \sin(\frac{\pi * Y}{l_i} + \xi_i) \tag{1}$$

$$0 < \xi_i < 2\pi, 0 < \zeta_i < 2\pi \text{ (Random Phase Shift)} \tag{2}$$

Equation 1 is suggested to model the random stochastic field in [1]. This formula (1) is the main component of the field generator. This formula is a fourier series of m sines with a random phase shift added on top of each other. The random phase shift in X and Y direction is different and is denoted by ζ_i and ξ_i. The deflection d is calculated for each point on the raster of the X and Y axis (the square roots of the area). The resolution of the X and Y axis is determined by the value of p. The value l_i represents the shape of one of the sines. An amount of m sines is added on top of each other. How the value of a_i is determined will

be elaborated in section 3.3.

Now we described how the function for calculating the deflection at a given point d works. This is iterated in X and in Y direction and by doing this a random field of deflections which takes a semi-continues shape is created. This is also what is observed in reality. Two options for the required output are given. A user can output all values of a single field or a user can output the maximum value of N fields. The first one offers the option to visually inspect a single created random field and the second option allows for the creation of datasets of multiple random fields for the given parameters ready to be statistically analyzed. All that is left to explain about the functioning of the field generator is explaining how a_i is calculated.

3.3 A closer look into the Imperfection-Ratio

At first the Imperfection-Ratio a_i was just implemented as a fixed number into equation 1. Mr Hoogenboom pointed out this is incorrect, therefore he proposed a different approach which takes the amount of summed waves (m) and the amount of scans (s) into account. In equation 1 a_i is still used, but this is no longer one of the Input Variables. Instead of a_i, impr is now one of the new Input Variables. impr is related to a_i but is not the same. They have a relation depending on m and s and on the Variance of the input spectrum.

$$V = \frac{1}{A}\int\int d^2dA = \sum_{i=1}^{m}\frac{1}{4}a_i^2 = (\frac{m}{12} - \frac{1}{48m})C^2\hat{l}^2 \rightarrow C = \sqrt{\frac{12}{m - \frac{1}{4m}}} * \frac{\sqrt{V}}{\hat{l}} \tag{3}$$

$$l_i = \frac{2i-1}{2m}\hat{l} \tag{4}$$

$$a_i = C * l_i \tag{5}$$

$$impr = \frac{\sqrt{V}}{\hat{l}} \tag{6}$$

These equations were provided by Mr Hoogenboom [6]. Now a_i is dependent on s & m & the Input Variable impr instead of a fixed number. The value for impr (equation 6) is itself a product of the scanned data, therefore it can only be used for designing a shell structure which will be built in the same way as the scanned shell structures. In this program it is treated as an Input Variable.

In Section 4.1 we will take a look into the scandata provided by the research conducted by Elferink et al. [1]. The imperfection ratio is the input value which is most difficult to measure and therefore to model. The value for the amplitude imperfection ratio which is mostly used throughout the project is 1/638, this is the mean value of all four scanned shells [1], don't just take the mean value but take for example the 95% value.

This section of the code was slightly adapted in the last week of the project. This is described in section 7.1. The remainder of the project is carried out with the initial version of the new field generator.

3.4 Example output & observations

```
plot_field(A=500, l_hat=3.0, impr=(1/638), s=4, p=400, m=50, N=1)
```

The Field Generator was extensively tested for different parameters. One of the outputs for the input displayed in the box is shown here in figure 2 and figure 3. Both Input Variables as well as Process Parameters. The Field Generator behaves well for a broad range of variables. Problems only arise when they are to be expected, for example when choosing a buckling length which is larger then the square root of the Area. Something strange which occurs with all sorts of chosen parameters are the 45 degree angles in the patterns. This

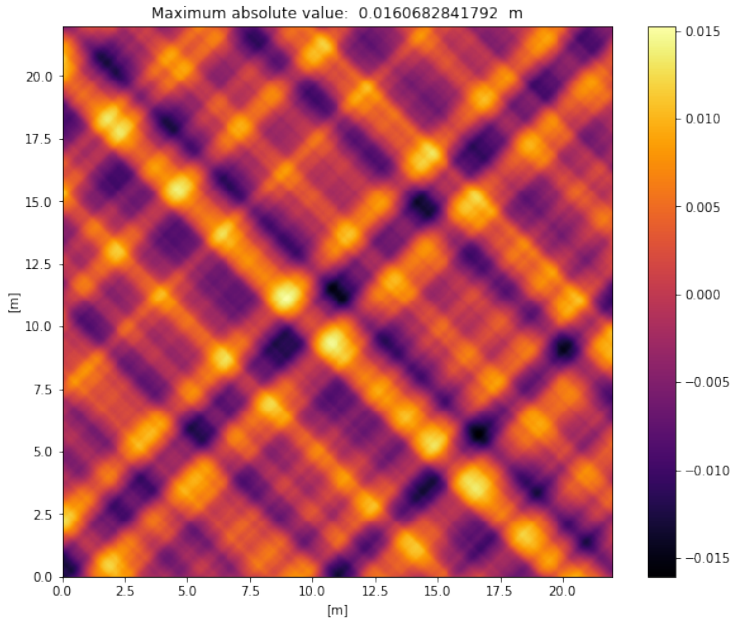

Figure 2: An example of a generated Random Field made using the Field Generator. The maximum deflection value is printed to the title of the plot.

effect isn't visible in the scanned data which raises questions whether this is a good approximation of the reality. This issue will be further explored in section 4.1.

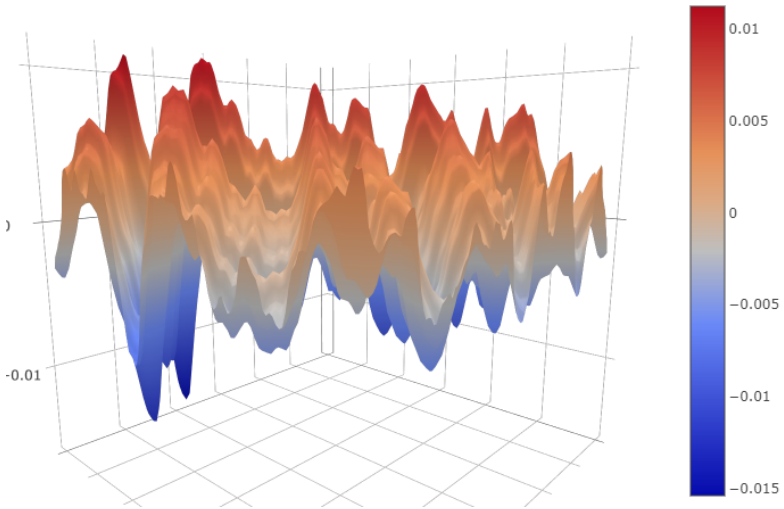

Figure 3: An example of a generated Random Field made using the Field Generator. A 3d view of the random field.

A lot of additional code, next to the field generator, is written in order to display the data, such as for plotting figure 2 and for other applications. In order to fully test the output of the field generator a simplified statistical analysis was performed to test the code as well. In figure 4 a plot of a very simple statistical analysis is shown.

A Normal distribution was fitted onto some test data with the same Parameters as the Field shown in figure 2. This field has a maximum value, such a maximum value is one of the datapoints in the histogram. This is the relationship between both output options of

the field generator. More in-depth statistical analysis will be performed later on in section 6, this is just a simple test to test whether the field generator works well.

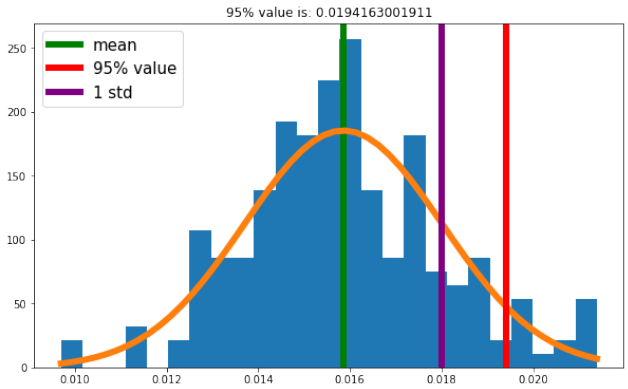

Figure 4: An example of the most basic statistical analysis to test the code.

4 Comparison of the Simulations versus the Scandata

The cause for this project is the research conducted by Elferink, Eigenraam, Hoogenboom, & Rots [1]. In their research they acquired scandata of 4 existing shell structures. This is explained in detail in the introduction.

In order to be able to make sure the simulations represent the reality well it is necessary to compare the output of the simulations with the real scandata. When running the simulation with input values corresponding to one of the real structures, the output of the simulation should be similar to the observed scandata. This wouldn't proof the simulations is a perfect representation of reality, but it would however show the simulation is close to reality. If it wouldn't be possible to even remotely simulate the scandata with the corresponding input values that would be worrisome.

4.1 Scandata of the existing shell-structures

The four plots of the imperfections resulting from the research conducted by Elferink et al. [1] are shown in Appendix B.

Some of the shell-structures have a strange shape, two are trianglular, one is squrare but has a gap in the middle. Thefore the best real shell structure to compare the simulations to is the tennis hall which is displayed in figure 5.

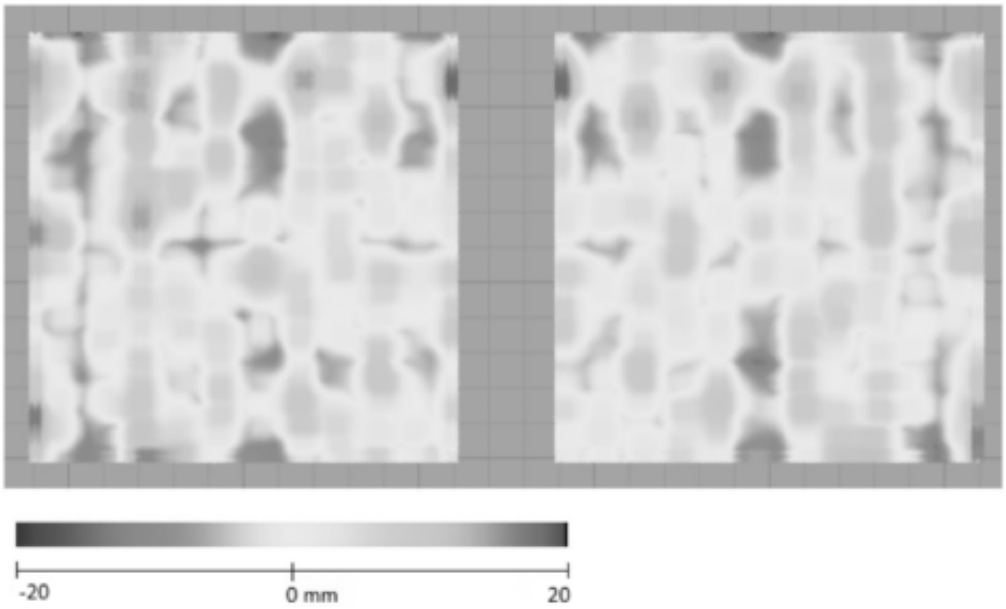

Figure 8. Shape imperfections in the roof of Heimberg tennis hall

Two patches of 12 x 12 m; control point spacing of 0.25 and 4.00 m; largest observed imperfection amplitude is 23 mm

Figure 5: Imperfection plots of scandata of the tennis hall. [1]

It should be noted theses plots were based on the double surface method [1]. The method used in the program is the single surface method. This might cause some inconsistencies when comparing the one to the other.

4.2 Comparing the Simulations to the scandata

When entering input values corresponding to the physical properties of the tennishall, the output should look like the plot of the real scandata in figure 5.
The values for the input parameters for this typical shell structure need to be found. The area is 144 (12x12m). The imperfection length used for all calculations in the paper is 3m. the amplitude imperfection ratio is 1/638 [1] [6]. The amplitude imperfection ratio is explained in more detail in section 3.3. This value is calculated assuming a buckling length of 3m, therefore picking that as the value is mandatory for simulating this shell structure. Note however that the used imperfection ratio of 1/638 is the average of the entire spectrum of scandata (the 4 scanned shell structures). There is a reasonable chance that the (average) imperfection ratio for this typical structure is different.

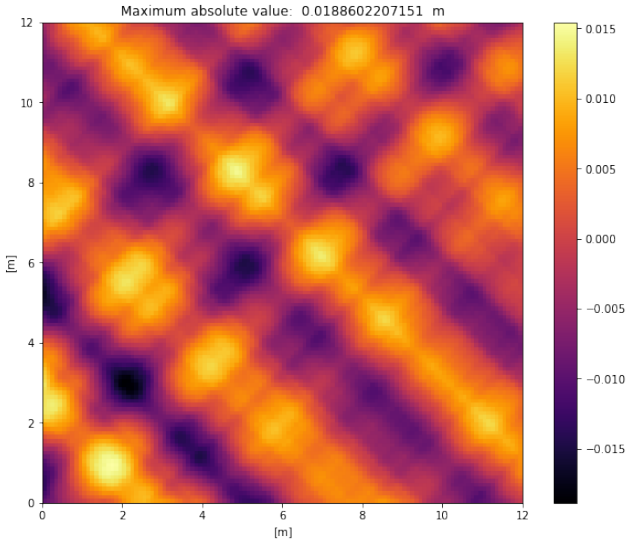

Figure 6: Plot resulting from a simulation with Input Variables corresponding to the scanned tennis hall shell structure

The plot resulting from a simulation with input variables corresponding to the scanned tennis hall shell structure is shown in figure 6. The plot doesn't look quite similar to the scanned data plot. Also the maximum value of the simulation differs from the largest observed imperfection amplitude of the scandata. After some testing it is observed that changing the imperfection ratio to 1/400 results in values similar to the observed maximum amplitude. As stated above the value of the imperfection amplitude ratio (1/638) is the average of the entire spectrum. The value of the tennishall might be different. This could be a possible explanation for the difference in maximum deflections. The maximum deflection is very sensitive to the value of the imperfection ratio.

The maximum deflection is not the only thing that stands out when comparing the simulation result to the scandata plots. The plots don't look that similar either. In the simulations, every time again, a clear 45 degree angle between the x and y direction is observed. This isn't observed in the plotted scandata. The tennishall is the structure which shows the least random deflections, the other structures show even more random patterns These plots can be observed in Appendix B.

In this section we showed the issues and inconsistencies of the simulations compared to the scandata. In the following section it is tested whether changes in the field generator code might solve these issues. In the final section of this chapter we draw conclusions on the reliability of the simulations.

4.3 Explored adjustments to the Field Generator

Different adjustments to the field generator were tried in order to find ways to make the simulation output more similar to the scandata plots. Many different attempts were made, some of them, and their impact on the look and behavior of the plots are shown here.

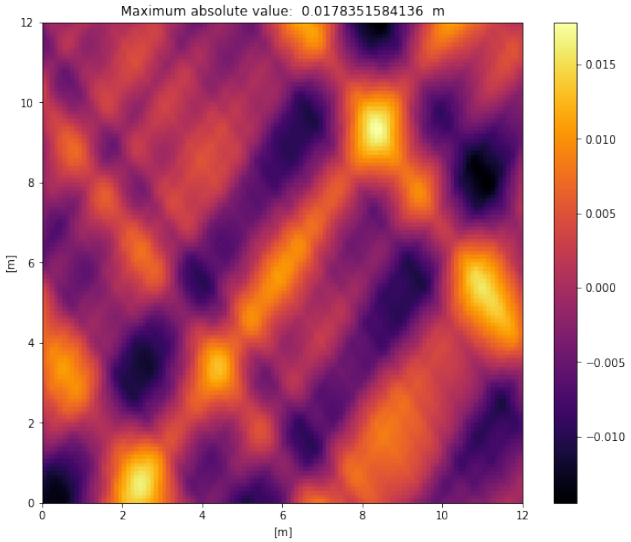

Figure 7: Output of alteration 1 to the field generator.

figure 7 shows what happens when multiplying l_i by 1.5 in the Y direction in equation 1. The 45 degree angle is gone, but a fixed angle (just not 45 degrees anymore) is still visible. When changing the factor 1.5 to another value, the angle changes but the Scottish pattern remains. The values don't change significantly for this change.

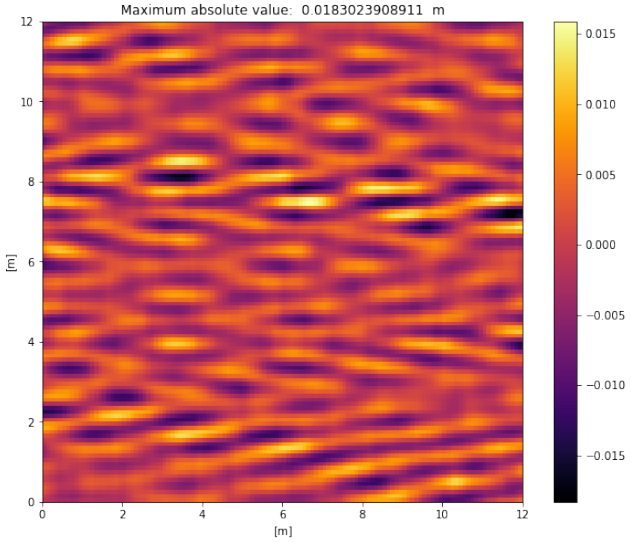

Figure 8: Output of alteration 2 to the field generator.

Figure 8 shows what happens when multiplying l_i by a random number between 0.8 and 1 instead of the fixed 1.5 in the Y direction in equation 1. This makes the output way more random, but random in such a way that the natural waves which do occur in the scandata are gone.

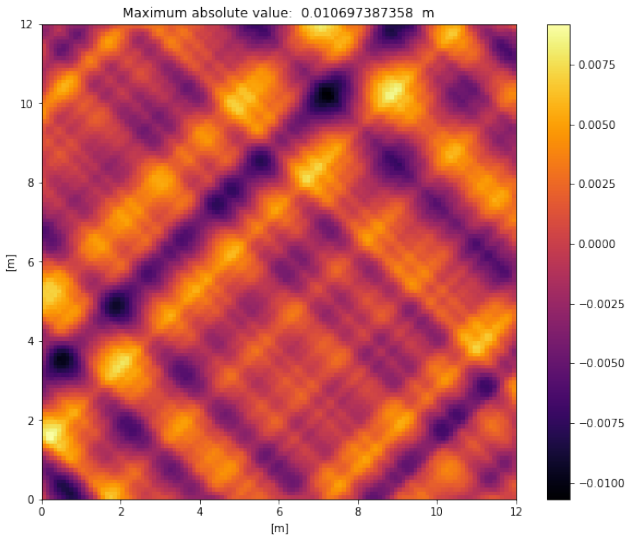

Figure 9: Output of alteration 3 to the field generator.

Figure 9 shows what happens when changing this part of the field generator code described in section 3.2.

```
    for i in range(1, m+1):
        l = ((2 * i - 1 ) / (2 * m)) * l_hat
        a = C * l
```

to

```
    for i in range(1, m+1):
        l = ((2 * i - 1 ) / (2 * m)) * random.uniform(0.5 *l_hat, l_hat)
        a = C * l
```

This results in the same look and behavior of the figure as in the original version, but the deflections are lower because smaller, random values for l_hat are used. Changing this part of the code does not have an impact on the pattern of the output can be concluded.

In the next section conclusion based on these attempts and results will be discussed.

4.4 Conclusions about the reliability of the simulations

It can be concluded it didn't succeed to exactly reproduce the plots of the scandata from the simulations. The part of the field generator which is responsible for the pattern in the output plots was isolated. This is the equation of the fourier series (equation 1) Changes to this function changed the output, but it didn't succeed to reproduce the paterns of the scandata. This might mean the approach of using fourier series for this application is not accurate. But maybe it is just wrong to want to reproduce the patterns of the scandata by a random field. Is it really realistic to expect a certain pattern in a randomly generated field? The deflections in a random field are ofcourse random and maybe the pattern doesn't matter when trying to find the characteristic value. Unfortunately I cannot prove either of these two assumptions.

The maximum deflection from the simulations was also lower compared to the maximum observed imperfection. This could be due to the used input value of the imperfection amplitude ratio. The values of the deflections are very sensitive to the input value of the imperfection amplitude ratio.

From these observations it cannot be concluded with certainty whether or not the simulations are a good representation of reality.

One thing to be certain about is that the tested alterations on the field generator have no

mathematical or physical foundation, therefore we stick with the current version of the field generator.
From all of the tested variations the original one is the closest representation of the scandata. When rotating this output by 45 degrees the output looks somewhat like the plot of the scandata. The axis in the random field generator have no physical meaning, therefore there are no objections for a coordinate system rotation of 45 degrees.
In the remaining part of the project we will stick to the original field generator which is described in chapter 3.2.

5 Optimization of the Process Parameters

The next part of the project is the optimization of the Process Parameters.

The Process Parameters determine the accuracy of the output and the runtime of the program. These parameters will be kept variable, but it is desirable to optimize these parameters and use those values as a standard value. Of course the user can deviate from these recommended values.
The big question of this Section is: What is the optimum between accuracy and runtime. With a very large runtime the results will be very accurate, and with only a few quick iterations the results will not be trustworthy. The optimum lies somewhere in between.

During this optimization the Input Variables will be held fixed. These values were suggested by Mr. Hoogenboom [6] as representative average values for the scanned shell structures.

The fixed Input Variables are:

- **A = 500** - Area $[m^2]$
- **l_hat = 3** - Buckling length [m]
- **impr = (1/638)** - Amplitude Imperfection Ratio [-]
- **s = 4** - number of scans of input data [-]

For the optimization of a Process Parameter the other two will be temporarily fixed. Two Process Parameters will be fixed at these values while the other one is optimized.

The Process Parameters are:

- **p = 400** - points per m^2 $[1/m^2]$
- **m = 30** - number of added waves [-]
- **N = 1000** - number of fields generated [-]

For the optimization of each of the Process Parameters, this Parameter will be made variable again and be replaced by an array of 60 values of this Parameter ranging from a very small to a very large value, initially in 60 steps.

- **p** - ranging from 2 to 10.000
- **m** - ranging from 1 to 2000
- **N** - ranging from 5 to 100.000

A large dataset consisting of 180 csv-files was generated in order to be able to conduct this analysis. This simulation took in total 7 hours to run and several days to prepare. After this analysis was completed Mr. Hoogenboom noted that most of the 60 values of each parameter were picked in the lower end of the range. This was done because the optimum was expected to be in this part of the range, and because the highest values were only intended for comparison means. By the advice of Mr. Hoogenboom more high-end values were added and inspected. This proved indeed to result in a better understanding of the behavior of the parameters, although the optima turned out to be in the expected lower end of the range as was expected.

5.1 Explanation of the simulations

	Nvar27
datetime =	2017-12-09 23:45:15.583223
A =	500
l_hat =	3
impr =	0.001567398119122257
s =	4
p =	400
m =	30
N =	700
runtime =	0:00:22.796641
NaN	0
0	0.02010181757421338
1	0.016509341310940215
2	0.015779413576041276
3	0.013369111707256994
4	0.016420320566625547
5	0.01648651355469036
6	0.01797160745494837
7	0.01732004498777466
8	0.01506670741794999
9	0.01911351054616351

Figure 10: An example of the datasets generated for the optimization of the process parameters.

The simulations are programmed to create csv-files such as the one shown in figure 10. In this figure the first 10 generated values are shown. Looking at the parameters shows there are 700 generated values in reality for this particular dataset. This particular file is the 27th of the simulation where N is variable. The value for N in this simulation is 700. The other parameters (A, l_hat, impr, s, p and m are constant for different N values in different simulations. The program encodes all selected input parameters, all selected process parameter and the runtime to the top of the csv-file. The runtime is important for the analysis of the accuracy, the datetime notation is only there for future reference. Below that the values of the maximum deflections created by the field generator program for each iteration are encoded. This means the program creates a random field such as described in chapter 3. The program takes the largest absolute deflection in that field and appends it to the CSV file. The program repeats this N times. In the dataset in figure 10 only the first 10 deflection values are shown.

The values in the different csv-files are used to analyze the way the parameters, the runtime and the characteristic value (95%-value) are related to each other. The characteristic value is computed by sorting all deflection values (700 in this example) of the field generator and picking the value for which $i \geq \frac{0.95N}{N} * N$. The computations in section 5.3 are based on these characteristic values.

Based on the data obtained from the simulations which is stored in these csv-files it is possible to conduct multiple analyses. The performed analyses and their results will be elaborated in the next subsections.

It should be noted these simulations are based on a fixed set of input variables (A=500, l_hat=3, impr=(1/638) and s=4) Varying these values might change the results of this analysis, but would complicate the analysis by that much that it wouldn't be possible to draw conclusions based on it. This analysis should be seen as a 'ceteris paribus' analysis.

5.2 Process Parameter values vs Runtime

The first step of this optimization analysis is reviewing the effect of a change in Process Parameter on the Runtime. The results of this simulation can be seen in figure 11.

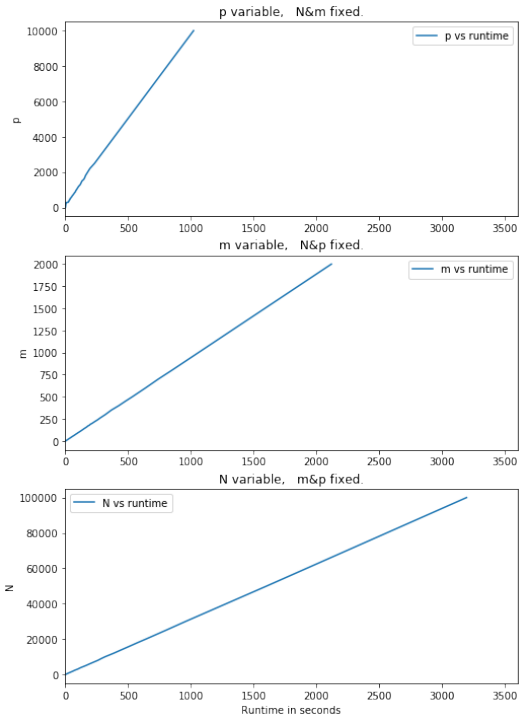

Figure 11: The effect of change in the value of a Process parameter to the Runtime when all other parameters are fixed.

It was to be expected that all parameters would have a linear behavior with the runtime. When doubling the value of a parameter, the runtime also doubles. This was to be expected but we had to make sure of this.

It should be noted though that an increase of a given parameter doesn't necessarily increase the runtime by the same amount as a same increase in another parameter. The slope of the line is meaningless in this figure. The x-axis is the same scale for all three figures, but the scale of the y-axis is different for each parameter, therefore the slopes in this image have no meaning.

From this data formulas linking the runtime of the program and the values of the parameters can be derived. By curve-fitting the function $f(runtime) = a * runtime$ the best fitting value of the slope was determined for each parameter. This resulted in the following formula's linking the runtime to each parameter.

$$p = 10.223 * runtime \tag{7}$$

$$m = 0.954 * runtime \tag{8}$$

$$N = 30.936 * runtime \tag{9}$$

These equations express the extra amount of runtime increasing the corresponding parameter requires. It should be noted this only applies when the other parameters are being kept fixed at the standard values. Nonetheless these formulas present useful insights in the

functioning of the program and will later on prove to be useful. One of the conclusions which can be drawn from these formula's is the fact that, in terms of runtime, it is way cheaper to increase the value of N than it is to increase the value of m. Although the exact numbers in the formula will be different for other input variables, this relationship will remain the same. This insight will proof very usefull in the following sections.
Example: When the one runs a simulation with fixed parameters and next one runs the same simulation but with the value of N 60 higher, the simulation will take 2 seconds longer to complete. When one runs the same first simulation with fixed parameters and next one runs the same simulation but with the value of m 60 higher, the simulation will take 60 seconds longer to complete.
Although this provides some very useful insight, it has no physical meaning yet, but when we link the value of a process parameter to the accuracy of the simulation it will! This is exactly what we will be doing in the following sections.

5.3 Effect of the process parameters on the accuracy

As stated in section 5.1 with the obtained data it is also possible to do an analysis of the relationship between the accuracy and the values of the process parameters. For this purpose quite some assumptions had to be made. For now let's state that we look at all parameters individually. This means all other parameters, including the other two process parameters, are fixed, except for the one we're interested in. Later on we will look closer into this problem and eventually find a partial solution which takes all variables into account in section 5.6.
The second assumption we need to make to be able to look into the accuracy is that "the deflection resulting from the largest parameter is the one true value". This assumption is obviously flawed, because even if it were correct one could always pick an even higher parameter value and then that one would be the true one. Also the question remains whether some noise will remain even for the largest researched values. This is done in order to be able to compare the lower values of the iterated parameter to a value of the parameter which is higher and therefore presumably more accurate. By doing this the data can be normalized. This assumption proves a very useful and necessary tool to be able to normalize the data.

Next we look at all three parameters individually. We look at the plots of the 95& values of the deflections versus the parameter values and examine this data. We also look at a plot of the normalized absolute errors versus the parameter values and examine this data. Finally we discuss the curve-fitted function resulting from the data of the last plot. A proposed optimal value and a minimal value for these parameters will be stated based on the analysis. This is done for all three parameters. In section 5.6 these results will be discussed and an approach to combining the independent variables will be discussed.

5.3.1 Variable N

N is the first process parameter which will be discussed. In order to be able to explain it clearly, a lot of figures are added to this subsection. For the other subsections the figures will be available in appendix C. As discussed in section 5.1 the 95% values of the different simulations will be used for the analysis.

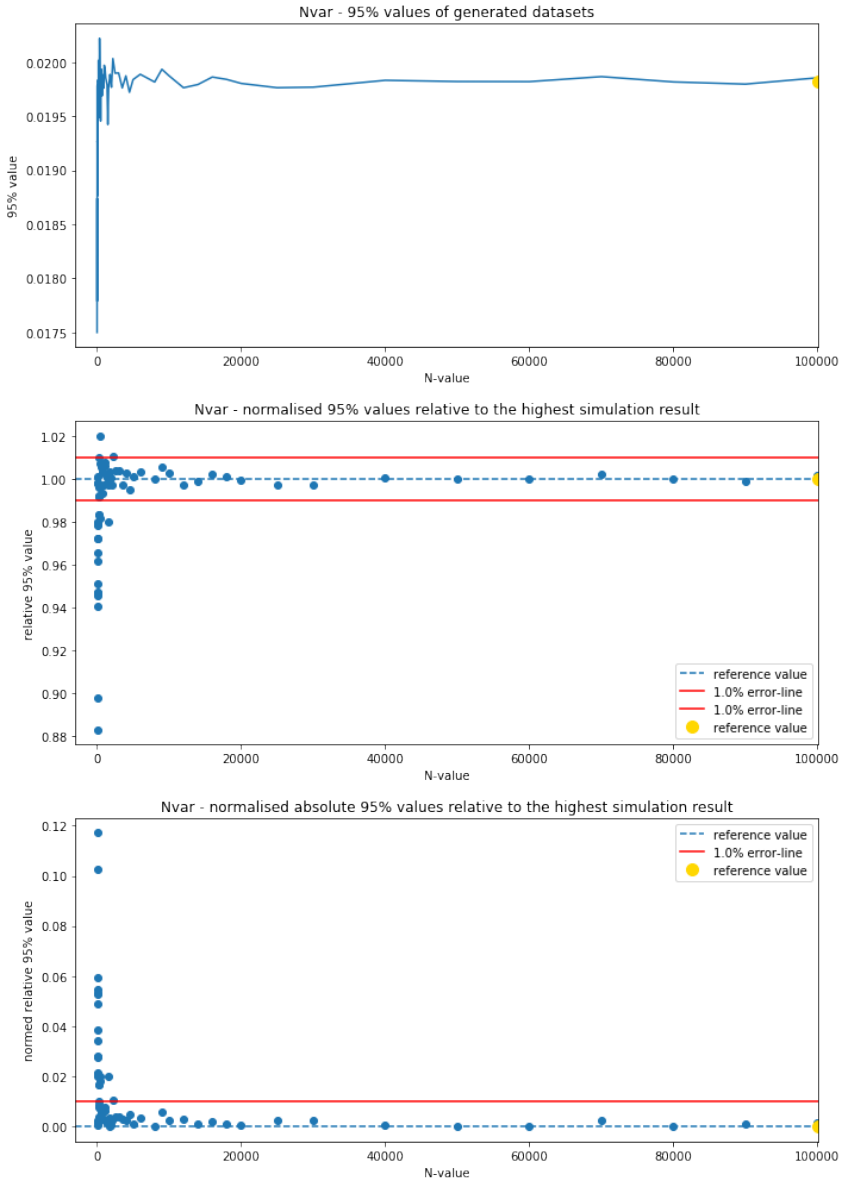

Figure 12: Plots of the simulation data for a variable N. Full range of N values. (a) actual 95% values. (b) Normalized 95% values. (c) normalized absolute 95% values.

Figure 12 shows the first analysis conducted on parameter N. In subfigure a the all simulated values are plotted. The last value, with the highest N (N=100.000) is yellow, this is the reference value which was also discussed in section 5.1. Subfigure b shows the same data, but then normalized to the reference value and shown as dots. Also two 1% error lines are shown to provide a visual reference for inspecting the noise of the data. Subfigure c shows the same data normalized to 0, and made absolute. Assymptotic behavior of the data can be observed.

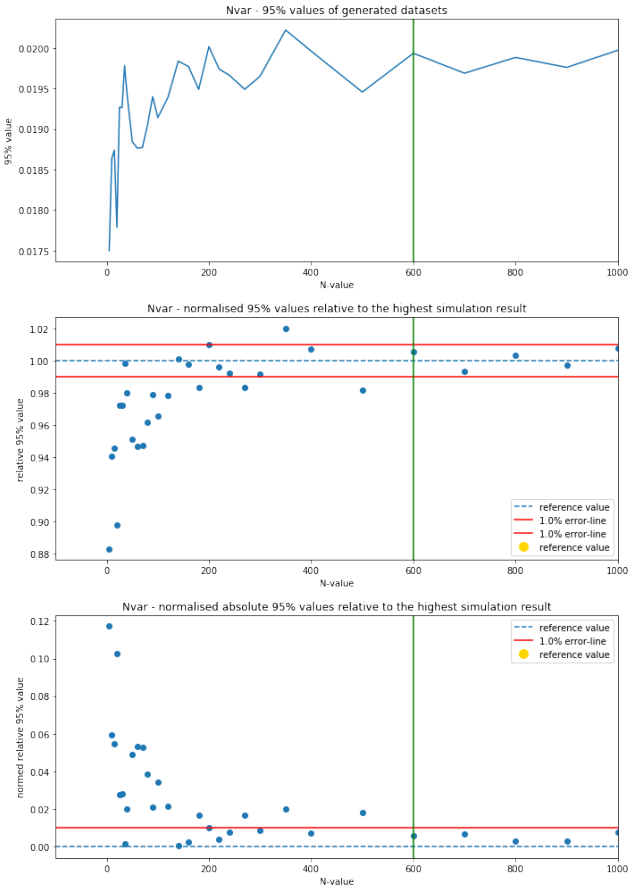

Figure 13: Plots of the simulation data for a variable N. Zoomed range of lower N values end where the optimum is expected to be. (a) actual 95% values. (b) Normalized 95% values. (c) normalized absolute 95% values.

These plots provide a lot of insight, but the lower end of the dataset is probably more interesting to inspect. Therefore a zoomed plot with N values from 0 to 1000 was made. In this image it is clearly visible that also in the lower end of the spectrum the error decreases with an increasing value for N. Inspection shows that a value of 600 should be the minimum value to have a reliable result.

The data of the previous figure lends itself perfectly for curve fitting. A function resulting from such a curve fit would provide a relation between the value of a process parameter and the accuracy of the simulation.

Inspecting the data resulted in two possible functions which could fit the data well. An exponential function and a power function with a negative power.

$$y = a * exp^{-bx}$$

$$y = a * x^{b}$$

Both functions were curve fitted to the data but for the N variable the exponential function proved to be a bad fit, for other parameters this one did work however. The power fit however turned out to fit the data really well. The values of the function parameters a and b are shown in the equation shown below. A variance matrix was also computed for these parameters, but this is not included in the report.

$$error = 0.2655 * N^{-0.5419} \tag{10}$$

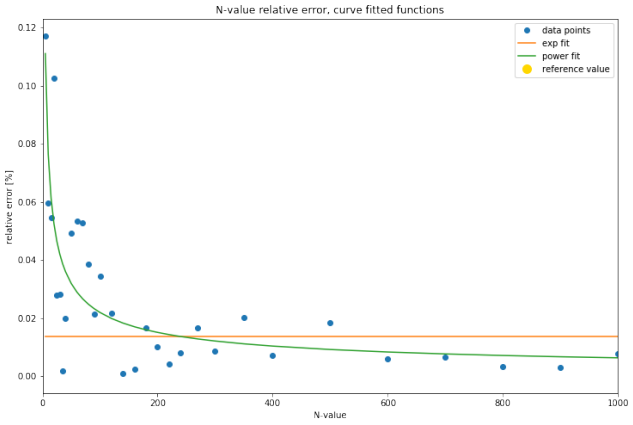

Figure 14: Curve fit of the simulation data for a variable N.

This equation was plotted in figure 14 together with the processed simulation data. This equation shows that the more we increase the value of N the closer the error value gets to 0 (i.e. the reference value). This behavior is to be expected when taking into account the meaning of N in the program. A larger N means a larger dataset of which the 95% value is computed, it is logical this results in a better approximation.

In section 5.4 we will continue with this knowledge, first we will investigate the other process parameters in the next subsections. In order to keep the size of the report limited the plots for the other two variables are not included in the report itself but are instead included in Appendix C.

5.3.2 Variable p

In order to keep the report compact the plots of the p and m variable are included in Appendix C. The analysis applied to these variables is similar to the analysis of parameter N.

Figure 25 a shows that the noise for process parameter p does not decrease for an increasing value of p. This was the case for process parameter N, but not for p. The remaining noise is not noise caused by the value of p but by the, not sufficiently large, constant values of N and m. The error for p remains approximately 2% also for very large values of p. This means there is no reason to further increase p when a certain value is passed. From p=50 the data seems to have converged and from this point on the noise remains the same, about 2 tot 2.5 % as can be seen in figure 26 in Appendix C.

This data was also curve fitted with the power function, but this proved to be less useful because the noise remains for this parameter.

$$error = 0.1431 * p^{-0.4643} \tag{11}$$

The resulting function 11 is useful for the lower part of the p values however.

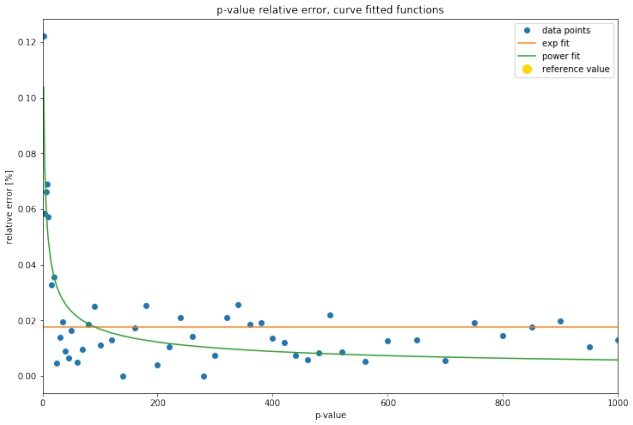

Figure 15: Curve fit of the simulation data for a variable p.

5.3.3 Variable m

Figure 27 and figure 28 show the plotted data for a variable m. In the zoomed figure the data shows a very nice round shape for an increasing value of m. Therefore this is the only parameter on which the exponential function works well. The data converges nicely but some noise is still present. It is expected this is also due to the too low values of p and m. A minimum value of 40 is proposed based on figure 28. For this value of m the nice curve has stabilized. A higher value is recommended of course. Both the exponential and power function work for this data when curve fitting. Although the exponential fit seems to be better in this image, this only holds for the lower values. The power fit has an overall lower standard deviation and it behaves well better for higher values of m. Therefor the power function is chosen for this parameter as well. The resulting parameter values are shown in equation 12.

$$error = 0.5942 * m^{-0.8510} \tag{12}$$

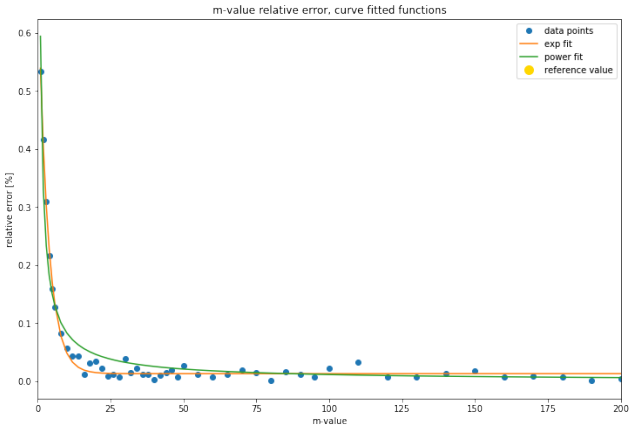

Figure 16: Curve fit of the simulation data for a variable m.

5.4 Linking runtime to accuracy

Now we have 3 equations that link the runtime to the parameter value (equations 7, 8, 9) and 3 equations that link the parameter value to the accuracy (equations 10, 11, 12) it is possible to combine these in order to derive an equation that relates the runtime to the accuracy. In the final program this will be a very useful tool for the user because the user

will be able to pick a desired accuracy and the program will be able to predict the runtime this will take.

The equations could be substituted into each other, this allow the user to specify a desired accuracy. The program can calculate the the PP-value that relates to this accuracy and it can calculate the associated runtime and show that to the user. Next it takes the calculated PP-value and inputs that into the field generator.

This approach looks very promising, but it should be noted this approach assumes that all other variables (the input variables and the other two process parameters) are the same as when we ran the simulations. Also note that not even the optimal process parameters are used in this approach, the initial input values are used. The optimal process parameters are not yet determined. This approach would make more sense when the 3 optimal values, which have yet to be determined, were used in a new simulated and the same analysis was conducted again on this data. However this would start an iterative process which makes this an unfeasible option. These issues will be further discussed in the following section. Also a brief noice analysis will be conducted and discussed.

5.5 Noise analysis

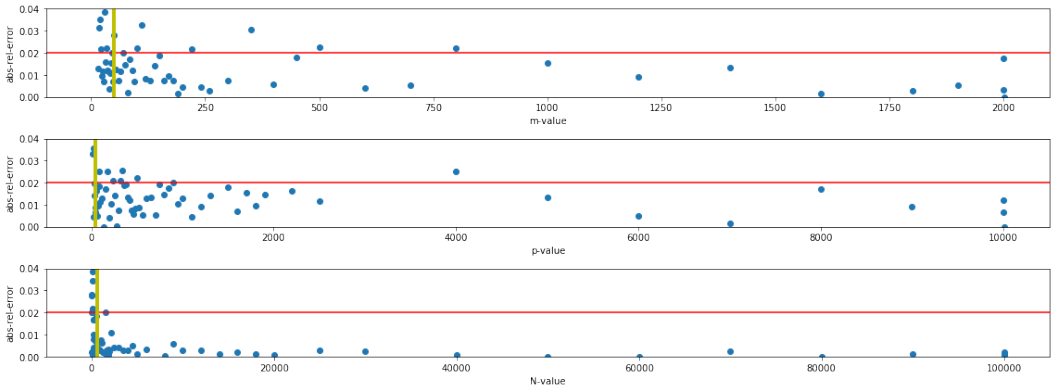

Figure 17: A plot of the noise of all three parameters over the entire range. Including minimum value. (a) parameter m, (b) parameter p, (c) parameter N.

Figure 17 shows the noise of all three process parameter along their entire spectrum. From this image it can be concluded that the accuracy of the process parameters behave in different ways. The proposed minimum value for each parameter is also included in the plot, but this value is better visible in earlier figures in section 5.3.

This figure shows that the previous approach of finding continues equations describing the process are not valid. Look closely to the upper range of the m and p values. Values very close to the maximum value which is used as a reference value are not even close to 0. This proves these parameters do not converge and the noise that is present just remains. When looking at the figure of the N parameter this shows this data does in fact converge nicely. The standard deviations of the curve fitting parameters of the power function support this claim as well.

It can be concluded that the tried approach of finding an analytical solution for this optimization problem is not valid. In the next section we will look into how the parameters are related and what this means for their accuracy and we will finally determine standard values for the process parameters.

5.6 The combined accuracy of all process parameters

Before this section we treated the process parameters as ceteris paribus parameters. We looked at each one individually while keeping the other ones fixed. Ofcourse this is not a realistic approach because the different parameters do influence each other in fact. Running a simulation with N=1000 and m=80 may be more accurate then running a simulation with N=1100 and m=30, although when only looking at the N value one would expect otherwise. We need to link the parameter in some way in order to be able to define the accuracy of the process parameters as a whole.

The relative errors (not absolute) compared to the reference value as discussed in section 5.3 can be treated as a distribution which has a mean and a standard deviation. Look for example at figure 12b and imagine a histogram on the vertical axis. For this situation a mean (which might be close to the reference value) and a standard deviation can be derived. By looking at the errors to the reference value of the simulation data in this way it becomes possible to use statistical properties of distributions and link the accuracy of the different process parameters.

The variance of the sample is equal to the standard deviation squared.

$$Var[X] = \sigma^2$$

Although the parameters do influence each other they are assumed to be uncorrelated here because they are not related in way in which the covariances influence each other. For parameters that are related to each other in this way the following formula holds.

$$Var[X + Y] = Var(x) + Var(y)$$

The error of the parameter related to the reference value is proportional to σ, therefore it holds that for each parameter the error value multiplied by a given scalar is the standard deviation.

$$parameter\_error \sim \sigma$$

Combining these three equations and relations results in the following equation:

$$Accuracy = \sqrt{N\_acc^2 + p\_acc^2 + m\_acc^2} \tag{13}$$

This equation describes the accuracy of the entire process based on the accuracy of all three input parameters. If reliable analytical solutions for each of the process parameters had been found these could have been implemented in this formula and the accuracy of the simulation could be determined based on the values of the process parameters. None the less this equation provides a lot of useful insight into the way the process parameters influence the accuracy of the simulation as a whole, this is called error propagation. For example when one chooses the values for the process parameters such that the ceteris paribus accuracy is 2%, the total accuracy which can be derived form the found equation is approximately 3.5%. Ofcourse not all three parameters need to have the same accuracy value, it is also possible to get the same resulting total accuracy for different values of the process parameters. When choosing N to have an accuracy of 1%, p of 2.3% and m of 2.5%, the combined accuracy is also 3.5%. Combining this with the equations linking the runtime (figure 11) to the parameter value and the individual accuracy's could be very useful. Unfortunately the curve fitting functions for the accuracy don't hold, but in the following subsection when we choose the optimal values this way the parameters influence the total accuracy plays a great role.

5.7 First proposal for Optimal values of the Process Parameters

Unfortunately the attempts to find an analytical solution for the accuracy versus the process parameter value didn't succeed. Therefore we are forced to pick optimal values for the process parameters by inspection, but based on the extensive acquired knowledge from the prior sections. Absolute minimum values were already derived for all three parameters in section 5.3 (N=700, m=40, p=50). Below these minimum values strange things happen to the accuracy and the results are generally bad in this region of the spectrum. These minimum value however don't provide the desired accuracy yet, a higher value should be picked to be able to guarantee the accuracy of the results of the simulation. Unfortunately the search for an analytical solution did not succeed, so we are stuck to using the noise analysis in figure 17 and the formula's linking the runtime to the parameter value (equations 7, 8 and 9) together with the acquired knowledge and understanding of the problem.

Because increasing the parameter N is relatively cheap in terms of runtime and contributes a lot to the upcoming statistical analysis a high value will be picked for N. The noise in the data of the variable parameters m and p doesn't seem to decrease at all past a certain point, therefore there is no point in picking a very large value for this parameter.

It should also be noted the simulations were made with the the following standard values for the process parameters: N=1000, p=400 & m=30. The runtime for this standard situation is 31.5 seconds.

Based on the noise analysis and the runtime equations the following standard values for the process parameters are advised:

First proposal for standard N value = 10000
First proposal for standard m value = 50
First proposal for standard p value = 100

Based on the derived formula's for the change in runtime for a change in parameter value the expected runtime for this simulation is: 31.5 + 9000 * 1/30.936 + 20 * 1/0.954 + -300 * 1/10.223 = 314 seconds. This is quite an increase, but this is mainly due to the increase in N, halving N would half the total runtime of the program, but later on in section 6 we will see a large dataset contributes greatly to the accuracy of the statistical analysis. In the next section we will see whether this estimate holds or not.

A rough approximation for the total accuracy with these values based on the figure in the noise analysis and equation 13 is:
N:0.5% m:2.5% p:2.3% and total: 3.4% (derived from formula 13). This error can be decreased by increasing the process parameters, but this will result in a longer runtime.

In the next section we will look at the way this first proposal for the standard process parameter values was tested.

5.8 Testing the proposed PP for different Input Variables

Now that we have proposed standard values for the process parameters it is time to test these parameters for different input values. For this purpose the procedure described in section 5.1 is slightly altered. Now we don't iterate over one process parameter with fixed input variables, but now we iterate different sets over the input variables with fixed process parameters. The approach remains the same.

Set number	A	l_hat	impr
Set 1	500	3	1/638
Set 2	300	5	1/400
Set 3	400	6	1/100
Set 4	486	5.3	1/575

Table 1: The different sets of Input Variables used for testing the process parameters.

The different sets of Input Variables that were used are displayed in table 1. These sets are selected in such a way that the values represent a large part of the spectrum of possible input variables, some values are nicely rounded, some are more random and some are floats. Set 1 are the input values which were used as fixed input values in the previous sections.

Another overnight simulation was ran for these parameter sets combined with the proposed standard process parameters. Of each of the sets of input variables 25 simulations were made, summing to a total of 100 simulations. The csv files resulting form these simulations look the same as the dataset depicted in figure 10. The data resulting from these simulations was thoroughly analyzed and a lot of new insights was obtained from this.

The first thing that was quite notable was the runtime, this was way less than the runtime estimated in the previous section. For the sets in increasing order the runtimes were approximately: 41, 29, 36, 41 seconds. This was way less then expected and below the reasonable desired runtime. This raised questions whether the process parameters m and p could be increased while still remaining within acceptable values for the runtime, and whether this would have a useful effect on the accuracy of the results or not.

A few quick tests showed that doubling p to 200 and m to 100 with the same input variables resulted in runtimes between 100 and 150 seconds. These runtimes are still acceptable but the question whether these process parameters result in better results arises. This was tested by also running 25 simulations of each set but with the higher process parameters. The two resulting collections of data were compared.

	set1	set2	set3	set4	set1_norm	set2_norm	set3_norm	set4_norm
0	0.019888	0.048396	0.232812	0.036609	0.000113	0.000451	0.005353	0.001217
1	0.019901	0.048260	0.231408	0.036470	0.000783	0.003265	0.000711	0.002566
2	0.019926	0.048168	0.230485	0.036863	0.002022	0.005166	0.004697	0.008187

Figure 18: The first few entries of the normalized 95% values of the different datasets for the low process parameters (p=100 and m=50).

In figure 18 the first three results of the simulations with the low process parameters are shown. In the first four columns the 95% value of each simulation for each set is shown. In the last four columns these values are normalized against the mean of the 25 values of the corresponding set. These values give insight in the spread of the sample of 25 values. When comparing the characteristic values of all 25 simulations for each dataset it turns out the values are very close. This is the way it should be, when running another simulation with same input variables, the same characteristic value should be returned as output by the

program. Although the program is made up of random processes, the output should not be random anymore but should always be the same to a certain extend because of the amount of simulations that are ran. Next, for each set the average of these normalized differences to mean is taken. These values, for the simulation with the low PP as well as the simulation with the high PP, are shown in table 2.

	Set 1	**Set 2**	**Set 3**	**Set 4**
Low PP	0.002248	0.002293	0.002188	0.002368
high PP	0.002206	0.003541	0.002813	0.002525

Table 2: The average relative normalized spread of the 95% values of the 25 simulations for each set with high and low PP values.

Table 2 shows that for each of the sets and PP values the average normalized spread of the 95% values is about 0.2-0.3%. The spread actually increased when using the high PP, this is logical because this simulation is 'more random' because of the higher m-value. No significant increases in accuracy is observed when using the high PP instead of the low PP. The absolute real difference in 95% values between the low and high PP was also computed. Again the average was taken of these 25 difference values for each set. The resulting average absolute differences are shown in table 3. This difference is very low.

	Set 1	**Set 2**	**Set 3**	**Set 4**
avg diff high-low	0.000104	0.000146	0.000711	0.000165

Table 3: The absolute difference of the 95% values of the 25 simulations of the high and low PP values compared.

From these two tables it can be concluded that increasing the process parameter values from m=50 to m=100 and from p=100 to p=200 does not contribute much to the accuracy of the simulations. This holds for all 4 datasets. The runtime does double for this increase in the PP values, therefore it can be concluded it is not worth it use the higher values for the process parameters. The low values of the process parameters will be used as Optimal values for the Process Parameters. These values will be used in the statistical analysis in the following chapter and they will also be the standard values in the final program.

Optimal N value = 10000
Optimal m value = 50
Optimal p value = 100

To conclude this chapter I'd like to state that I spent way too much time on this single chapter. The optimization of the process parameters could have been done in way less time when the approach of the last section was done from the start. Less insight would have been acquired in the way changes in the process parameters influence the accuracy and the runtime, but the same optimal values would have been found in way less time. This time could have been better spend exploring the statistic in further depth. Acquiring a better understanding about the statistics would have been more useful in retrospect because this knowledge is applicable outside of the final program. The optimization of the process parameters is only important within the scope of the program which will be made for this project.

6 Statistical Analysis

6.1 Characteristic Value

In previous sections the characteristic value was already used to perform multiple analysis on some datasets. The characteristic value is computed by sorting all deflection values of a dataset (10000 when using the optimal process parameters) and picking the value for which $i \geq \frac{0.95N}{N} * N$. This returns the 95% value of that given dataset. This value is the real characteristic value.

When assuming a normal distribution it is also possible to calculate the theoretical characteristic value with the mean and standard deviation of the data. Comparing the theoretical characteristic value to the actual 95% value gives insight in whether assuming the data as a normally distributed is a good approximation.

In figure 19 the statistical values of one of the 25 simulations of dataset 1 is shown (read section 5.8 for details about this data). The input values are also the same as the ones used in the first few sections of the chapter about the optimization of the process parameters.

```
mean:    0.0160467759821
std:    0.00218758077426
theoretical char_val:    0.0196344084519
sorted char_val:    0.0198492381738
real vs theoretical char value 1.0941 %
partial factor: 1.09804578967
NormaltestResult(statistic=276.49227864010413, pvalue=9.129869058276648e-61)
KstestResult(statistic=0.02758786529296775, pvalue=4.9009869493021106e-07)
```

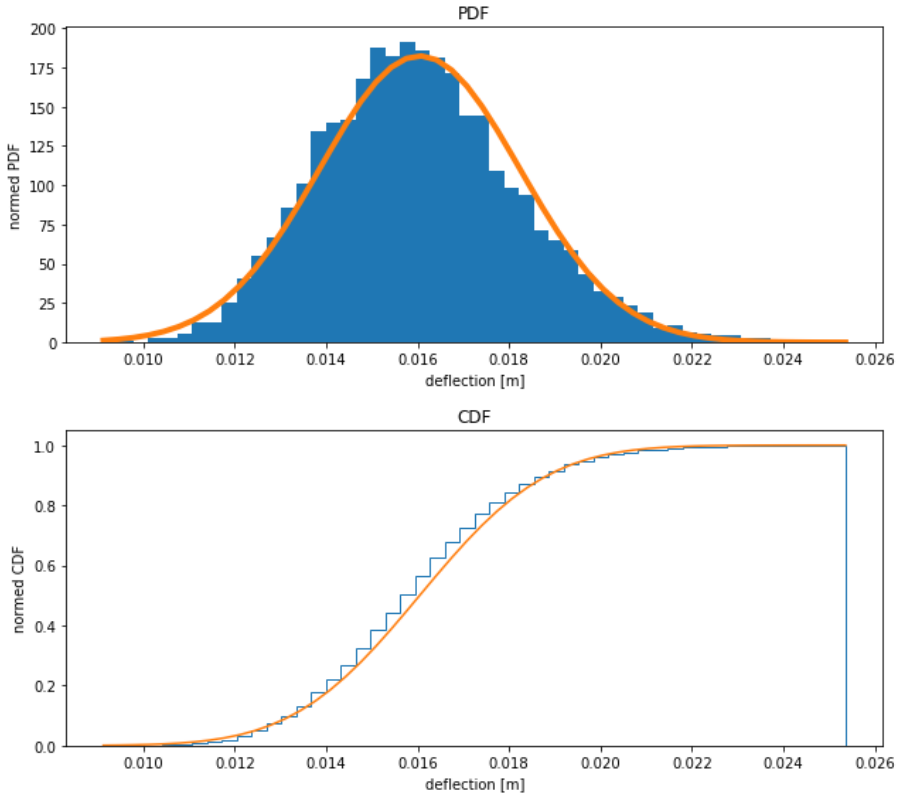

Figure 19: Results of a statistical analysis.

In this figure the values for the mean, standard deviation, theoretical characteristic value and the actual characteristic value for this dataset are shown. The difference in percentages

of the two theoretical and actual characteristic value is shown as well. This is 1.0941%, the number is positive which means the actual characteristic value is slightly higher. The distribution is more 'tail-heavy' then a normal distribution.

In figure 19 along with these numbers two plots are shown. The upper one is the probability density function along with a histogram of the data. The histogram is made up of 10000 datapoints. This plot shows the data resembles the normal distribution quite well, but the data is slightly denser in the 40-50% range compared to the 50-60% range. It also shows the upper tail is a little heavier then the normal distribution. Overall it can be concluded for this case the data seems to fit a normal distribution quite well. The 95% value, the single value we are interested in, is close to the theoretical value which is computed like this: $mean + 1.64 * std$. Visually and based on these numbers the data seems to fit the normal distribution quite well. At least well enough for practical applications. Why this is important will be discussed in the next section.

There are however tests which are made to determine what the chance is this data is originated from a normal distribution. These tests give a more objective view on the issue. Two of those testresults are also listed in figure 19. These are both functions from the scientific computing package Scipy which is one of python's many libraries. The tests are called 'Normaltest' and 'KStest'. These tests have as test-hypothesis that the dataset is drawn from a normal distribution, the output p-value is the chance this hypothesis is true. Both methods work in a different way, therefore the different values, but they both calculate moments of the data-distribution and compare this to those values of a normal-distribution of the same size with the same mean and standard deviation. The p-values seem to be very low, but when I ran these tests with an actual dataset of the same length drawn from a normal-distribution the p-value turned out to be below 1% quite a few times.

Mainly based on the values and the plots it can be concluded the data fits the normal distribution well enough for the practical application within this program. With more time it could have been possible to investigate different statistical distributions, but this would have caused complications for the partial factor as we will see in the following section. For this reason this was not further researched.

Ofcourse this analysis was not just conducted on a single generated dataset. All 100 simulations (25 simulations per set) of the sets used in section 5.8 were evaluated. The first five rows of results of the 100 simulations are shown in figure 20. This section was written based on the analysis of just one of the datasets as an example because that is clearer to explain and enabled me to include plots, but ofcourse the test was conducted on all datasets. The values listed in figure 20 are the same values that are listed above the plots. The plotted data is the data shown in row 4 of figure 20.

	mean	std	thr_cv	srtd_cv	cv_diff [%]	normtest	kstests
0	0.016021	0.002207	0.019641	0.019888	1.2580	1.165411e-57	1.896659e-11
1	0.015984	0.002195	0.019583	0.019901	1.6221	1.150374e-84	9.152294e-10
2	0.015997	0.002214	0.019629	0.019926	1.5120	3.068659e-53	8.496774e-08
3	0.015980	0.002202	0.019590	0.019893	1.5466	3.274205e-56	3.921009e-08
4	0.016047	0.002188	0.019634	0.019849	1.0941	9.129869e-61	4.900987e-07

Figure 20: First few results of the statistical analysis of all simulated data.

6.2 Partial Factor

In the previous subsection we found a way to find the characteristic value and we showed that assuming the data to be normally distributed holds well enough for our practical implication for the program. Why this is of importance will be elaborated in this section.

In order to be able to use the standard formula for the partial factor in the FEM program which is out of the scope of the project, both the load and the (load due to the) imperfections need to be of the same distribution type. The imperfections work as a load on the structure. The load which will be placed on the structure in the FEM simulation is a snowload distributed over the area of the shell. This load is normally distributed. In order to be able to use the known and trusted formula for the partial factor both loads need to be normally distributed. Therefore the shape-imperfections which are added to the model as a load also need to normally distributed.[6]

If the shape-imperfections are not assumed to be normally distributed, the formula for the partial factor does not hold. In that case another formula for the partial factor has to be derived which takes this other combination of distributions into account. There are formula's for the partial factor where both distributions are for example from a log-normal distribution or from a gumbell distribution, those formula's exist. But the fact remains the snowload which is modeled in the FEM simulation is normally distributed. Therefore showing the shape-imperfections fit best to a gumbell distriubtion for example doesn't help us any further because then we are stuck with two loads of which one is normally distributed and one is from a gumbell distribution. No formula's exist for these combinations to my knowledge.

Since we showed in the last section that assuming the shape-imperfections are normally distributed is a reasonable assumption, it holds that for the application in this program it is best to assume that the shape-imperfections are normally distributed. The errors resulting from this assumption are small enough to be acceptable in the scope of this project. Mr Hoogenboom also advised to just stick to the normal distribution for this project [6].

Now that we explained why we assume the shape-imperfections to be normally distributed and we showed that this is accurate enough for our application we can continue to the formula for the partial factor we can use in the program [2].

$$\gamma_s = \frac{1 - \alpha_s \beta V_s}{1 - k V_s} \tag{14}$$

Because the shape-imperfections act as a load on the 'perfect' structure, the partial factor for the sollicitation is used. The symbol V_s stands for the Variance of the dataset, this is computed by dividing the standard deviation by the mean. The symbol k is 1.64 (95% value for a normal distribution). The symbol β is the index of reliability. This value is usually 3.6, but the user will be free to enter another value. The symbol α is the influence coefficient of the load. There are two possible values for α being 0.7 when the shape-imperfections are the dominant load and 0.28 when another load (the snowload for example) is the dominant load. The user will also be free to choose from these values.

With the derivation of the formula for the partial factor all pieces of the puzzle of the program are created and clarified. In the next section all individual pieces of code will be combined and the final program will be created.

7 Program

In this last chapter all previously acquired code and knowledge will be combined into a program in which users can enter input values and will receive the corresponding characteristic value and partial factor as output.

At the time of writing the final version of the report a proof-of-concept version of the back-end code which receives the input, performs the computations and returns the output and the GUI-code which can receive input within an interface and perform computations with these values and return the results to the same interface as output have been made. Therefore it is clear that the program as it was intended is possible to make. At the moment of writing the final version of the report these two proof-of-concept versions still need to be linked and some technical difficulties need to be overcome. For example including multiple plots in the interface and updated the figures with new input.

The code of a minimalistic version of the program which can be executed by a python interpreter in the command prompt without a graphical user interface is added to the report in Appendix D. This code consists of many of the isolated parts of code which were written in previous sections. Although it lacks an interface, this minimalistic program has the same functionality as the final program.

The program first outputs a plot of one of the generated random fields. This looks like figure 2. This plot is shown immediately and can be inspected while the rest of the simulations are computed. When this is finished the output of the statistical analysis is returned. This looks the same as figure 19.
Please note in that in this version of the program α, β and s are chosen to be fixed. Ofcourse those variables could be made to be variable as well. In this version α is 0.7 and β is 3.6. s is 4 because four shells were scanned.

A video showing the functionality of the program was made. This video can be watched via this link:

`https://youtu.be/wr8Ei5m77Hw`

The code which is executed by the program() command in this video is appended to Appendix D.

7.1 Alteration to the field generator

After handing in the concept version of the report a change was made on the field generator based on the advice of mr Hoogenboom [6]. Because there was not enough time this change was not incorporated in the entire report, but only in the final program. The analysis and optimizations in the report are based on the early version of the field generator, only this chapter is about the newest version of the field generator.

The amplitude imperfection ratio was fixed for each of the N simulations. This is not a good represenatation of reality because there is actually a lot of randomness and uncertainty involved with the input variable impr. This had to be incorporated into the field generator.

This was done by replacing the fixed impr for each generated field by a randomly changing value for impr. At first just the average value of the scandata was used. This was 1/638. This value has a coefficient of variance of 0.29. This variance was neglected and just the mean value was used. This neglected the variance in the scandata.

This was changed to a new value for each simulation. For each new simulation four values were drawn from a normal distribution with a mean of 1/638 and a coefficient of variance of 0.29. The mean and the variance of these 's' values (4 in this case) were computed. From a new normal distribution with the new mean and variance one new value was drawn. This value was used as impr. For each generated field this value for impr was computed again. This was the uncertainty of the scandata was incorporated into the field generator. The

updated code of the Field Generator can be found in Appendix F.

Instead of just providing a single value for impr as input now two values have to be provided to the program. The mean and the coefficient of variance of the amplitude imperfection ratio.

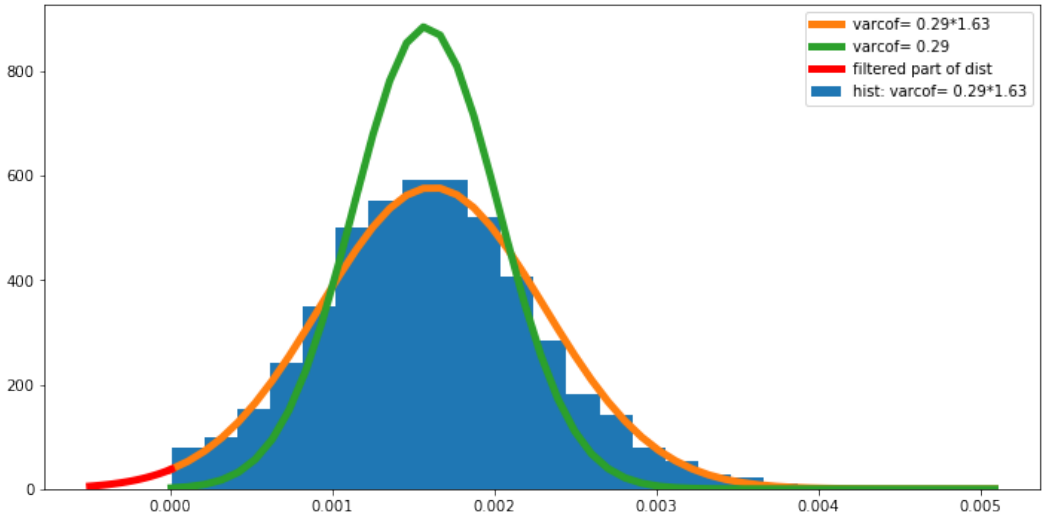

Figure 21: 10000 tests of the function for computing the input variable impr

Because of the small number of scanned shell a factor of 1.63 has to be added to the coefficient of variance to cover this extra uncertainty. This posed some problems With this larger variance the tail of the distribution surpassed 0 and some of the values for impr became negative (red part of the line in the figure). This is impossible and this resulted in a lower characteristic value because some unrealistic situations were taken into account in the simulations. Therefore the function was altered to only accept values higher then 0. In figure 21 the effect of multiplying the coefficient of variance with 1.63. When not doing so this problem doesn't arise. This figure also shows the possible range of imperfection ratio's which are used in different fields.
This figure clearly illustrates what changed compared to the previous version. In the new version impr is one of the values of the histogram for each of the generated fields. In the previous version impr was the same mean value of the histogram for every single generated field.

The altered field generator was tested and compared to the last version. These tests showed that the characteristic value slightly increased. This is mainly due to the fact that more extreme situations can occur because of lower values for impr. The behavior of the generator doesn't seem to be severely impacted, the shape of the statistical distribution remains the same and the values don't seem to be affected that much as well. There was no time to conduct all the analysis and optimizations in the previous chapters of the report again. By some quick inspections it seems these results are not affected that much, but this was not thoroughly researched and proven. This corresponds to the theory of Monto Carlo simulations that adding another stochastic variable doesn't interfere much with the results.

8 Conclusion

The end product of this project is not a conclusion but the program which is discussed in section 7. However to get to this program which includes parts of the code and the gained knowledge of the other parts of the project many different conclusions about these smaller parts had to be drawn. These conclusions will be discussed briefly in this section. For detailed information please read the corresponding chapters. This section will not be a summary of the project, just the conclusions. The summary can be found at the start of this report.

Generating simulated output which is similar to the scandata of one of the scanned shell structures described in [1] when using input values which represent the physical properties of these existing shell structures didn't work out. The observed patterns in the scandata cannot be simulated using this field generator. Multiple alterations to the field generator code were tried, but these were also not successful. The maximum deflection value is also a bit different. The observed spectrum falls within the observed spectrum of the simulations, but is a very high end value. However this is possible it would have been more encouraging if the observed value would be somewhere near the mean of the spectrum. It should be noted the average amplitude imperfection ratio of the observed spectrum (4 shell structures) was used in the calculation, not he average for this single shell structure. Changes in the input value imperfection amplitude ratio have a huge impact on the deflection values. Whether the conclusion can be drawn the field generator is not a good representation of reality because the exact observed pattern cannot be recreated is unclear. Personally I think the pattern doesn't really matter that much and conclusions on the pattern cannot be drawn because it is randomly generated.

The optimum process parameters are m=50, p=100 & N=10000. These process parameters make up the details of the internal processes in the program. Higher values yield more accurate results but require a longer computation-time. These parameters are separate from the physical input. In chapter 5 deep research was conducted in to the behaviour of the process parameters, to all three process parameters combined aswell as ceteris parabius analysis for each of them isolated. The runtime of the simulations linearly increases when increasing the process parameters. The simulations get more accurate when increasing the values of the process parameters. N keeps on getting more accurate for increased values, p and m only until a certain value, after this value the accuracy doesn't increase further due to remaining noise. The accuracy of the simulation is made up of the combination of the accuracy of the process parameters as described in equation 13. The optimal process parameters as a set were tested with multiple sets of input variables, this was compared to a set of higher process parameters. The accuracy almost didn't increase at all by this, therefore this is not worth the extra runtime. The advised optimum process parameters are m=50, p=100 & N=10000. Ofcourse the user is free to deviate from these values when using the program.

The normal distribution is not a perfect fit for the data generated by these simulations. The theoretic 95% value differs about 1 to 1.5% from the actual 95% value for almost all simulations. This seems to be an acceptable accuracy, please note this is only used for comparison, ofcourse the actual 95% is used. The data was tested with the normal-test and the Kolmogorov-Smirnoff test of the Scipy python library. This proved the data doesn't fit the normal distribution very accurately, but it does show it is not that far off either. For the practical application within this project it is fine describe the data by a normal distribution. By assuming the normal distribution to be a reasonable fit the partial factor formula which describes loads that are normally distributed makes it possible to add the shape-imperfections in combination with snowloads in FEM simulations (shape imperfections work as a load on the structure). This is one of the requirements of the program.

Based on these conclusions the program which is described in section 7 was made. All conclusions, knowledge and code which were acquired while working on this project are used for this program. The program is the end-product of this project.

In retrospect it would have been more productive to spend less time optimizing the process parameters in order to have had time to explore the statics in more depth. The process parameters are only part of this single project, knowledge about the statistics of stochastic fields representing shape imperfections are broader applicable.

In the last week some changes to the field generator were proposed by the supervisor. These changes were processed, but there was not enough time to repeat all analysis and simulations that were conducted in the sections 4 through 6. The changes were incorporated in the final program in section 7 and were extensively elaborated in the same section. Please note the conclusions are drawn based on the field generator in section 4.1. It is not certain the same conclusions hold for the version of the field generator which is incorporated into the final program.

9 Recommendations

During the project a few times the decision had to be made to continue to a next part before having the current subject explored to full detail. It might be useful to further investigate some parts of this project into more detail in further research.

Before writing recommendations about further research it's also good to state that it might be useful to have someone with more background knowledge about the research areas covered within this project compared to a Civil Engineering Bachelor's student look into these issues. During the project it became clear that understanding about Monte-Carlo simualtions, advanced statistics and probabilistic design which are not covered in the bachelor were required. Although it was certainly very useful for me to learn about these subjects, I only scratched the surface of these subjects and gained only a brief understanding about the specific parts I needed. Someone with more knowledge in these areas might reach better conclusions than I did.

The parts of the project which could be explored in more detail are:

The field generator. It might prove useful to explore different mathematical methods to generate random fields. In this field generator summed Fourrier Series were used. There are a lot of other ways to generate random fields. With the applied approach it proved to be impossible to accurately recreate the scandata, however it might be possible to do this when using another mathematical approach to create the random fields.

The process parameters. The research into the optimal values of the process parameters was stopped at a certain time because this only minor important part of the project took too long to finish and endangered being able to finish the entire project in time. The research was not finished and therefore further research might be very useful. The 'ceteris paribus' research into the isolated process parameters lends itself very good for an iterative process. In this project it was conducted for randomly selected standard values of m, p & N, but it might prove insightful to repeat it for the found optimal values for the process parameters. This might lead to new, better optimal values which could be analyzed again which initiates the iterative process. After a while when the iterative process has converged it might be possible to find an accurate analytical solution to describe the relation between process parameter value and accuracy as was attempted in section 5.4 through section 5.6.

The statistical analysis. In this project only the normal distribution was explored. Partly because of a lack of time and partly because this was almost required to be able to make the partial factor for the shape-imperfections work in combination with the normally distributed snowload which need to be able to be added in the FEM simulations. In this project it is demonstrated the normal distribution fits the data well enough for these practical applications but it is also shown that the normal distribution doesn't fit the data really that well. It would probably provide a lot of useful insight to test the data against other distribution types. A way to do this was explored but shortly thereafter abandoned because of the arising problems with the partial factor when the distributiontype of the data is assumed to be something else then the normal distribution.
Based purely on observations the Gumbell max distribution seems like a promising candidate because the data tends to be heavier in the 40-50% region compared to the 50-60% region and the upper tail is heavier than the normal distribution.

When these three parts of the project would be explored into more details and would lead to new knowledge, conclusions or code it would be very simple to change these parts in the program.

In the last week some changes to the field generator were proposed by the supervisor. These changes were processed, but there was not enough time to repeat all analysis and

simulations that were conducted in the sections 4 through 6. The changes were incorporated in the final program in section 7 and were extensively elaborated in the same section. Please note the conclusions are drawn based on the field generator in section 4.1. It is not certain the same conclusions hold for the version of the field generator which is incorporated into the final program. It would be useful to repeat the entire project with the new field generator.

10 References

References

[1] B. Elferink, P. Eigenraam, P. C.J. Hoogenboom, and J. G. Rots. Shape imperfections of reinforced concrete shell roofs. *Heron*, 61(03):177–192, 2016.

[2] Civieltechnisch Centrum Uitvoering Research en Regelgeving. *Kansen in de Civiele Techniek*. CUR, Delft, eerste dru edition, 1997.

[3] C J Gijzenberg. *BEP: Extreme waarde verdeling van stochastische velden voor variërende parameters*. Bachelor thesis report, Delft University of Technology, 2016.

[4] PCJ Hoogenboom. Finite element analysis of a conoid shell. Technical report, TU Delft, Factulty of Civil Engineering, Structural Engineering, Delft, 2015.

[5] PCJ Hoogenboom. *BSc Eindwerk opdracht - Extreme waarden van stochastische velden*. Opdracht bsc eindwerk, Delft University of Technology, Delft, 2016.

[6] P.C.J. Hoogenboom. Meetings with Dr. Ir. Hoogenboom, 2017.

[7] P.C.J. Hoogenboom. Notes on shell structures. Technical report, Delft University of Technology, Faculty of Civil Engineering and Geosciences, Delft, 2017.

[8] Vissarion Papadopoulos, George Stefanou, and Manolis Papadrakakis. Buckling analysis of imperfect shells with stochastic non-Gaussian material and thickness properties. *International Journal of Solids and Structures*, 46:2800–2808, 2009.

[9] T Van De Velde. *BEP: Extreme waarden van stochastische velden*. Bachelor thesis report, Delft University of Technology, 2016.

[10] A.C.W.M. Vrouwenvelder. Reader on Random Vibrations. Technical report, Delft University of Technology, Faculty of Civil Engineering and Geosciences, Delft, 2018.

[11] Isaak Vryzidis, George Stefanou, and Vissarion Papadopoulos. Stochastic stability analysis of steel tubes with random initial imperfections. 2013.

11 Appendix

11.1 Appendix A

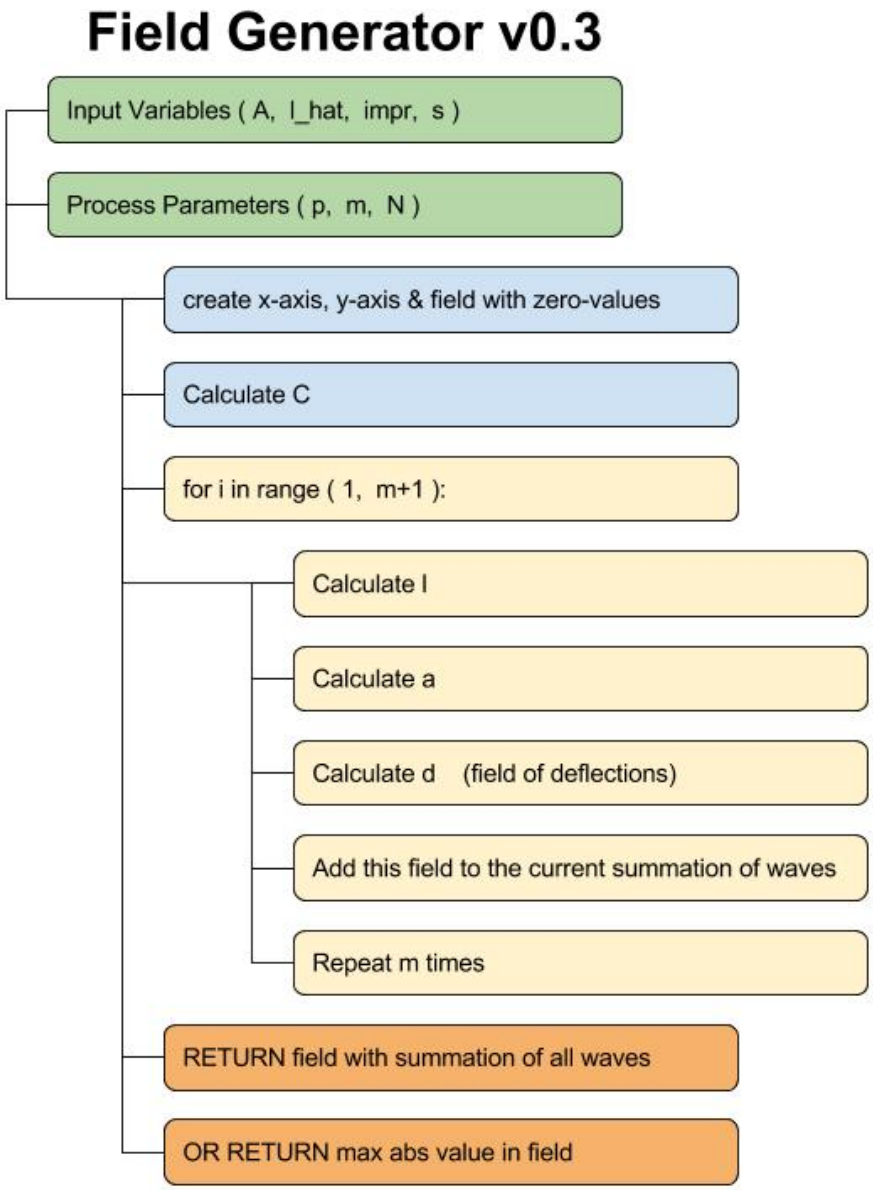

Figure 22: A schematic overview (PSD / Program Structure Diagram) of the functionality and parameters of the Field Generator program.

11.2 Appendix B

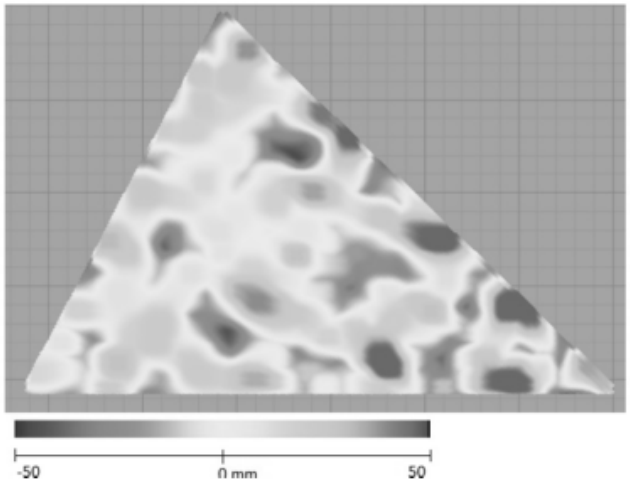

Figure 5. Shape imperfections in the North roof of Deitingen petrol station
Patch of 33 x 21 m; control point spacings of 0.25 and 4.70 m; largest observed imperfection amplitude is 64 mm

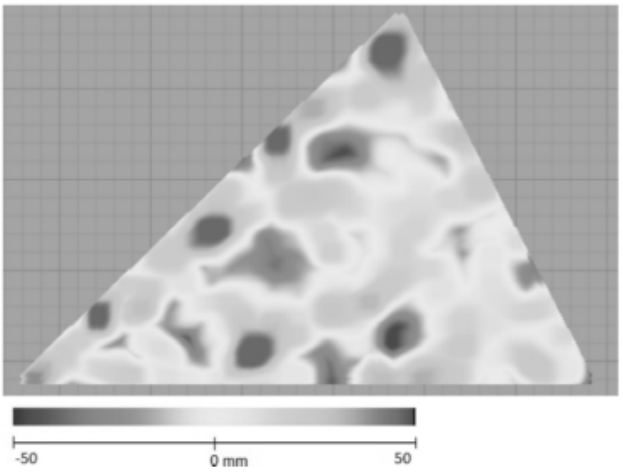

Figure 6. Shape imperfections in the South roof of Deitingen petrol station
Patch of 33 x 21 m; control point spacings of 0.25 and 4.70 m; largest observed imperfection amplitude is 60 mm

Figure 23: Imperfection plots of scandata of both petrolstation shellstructures. [1]

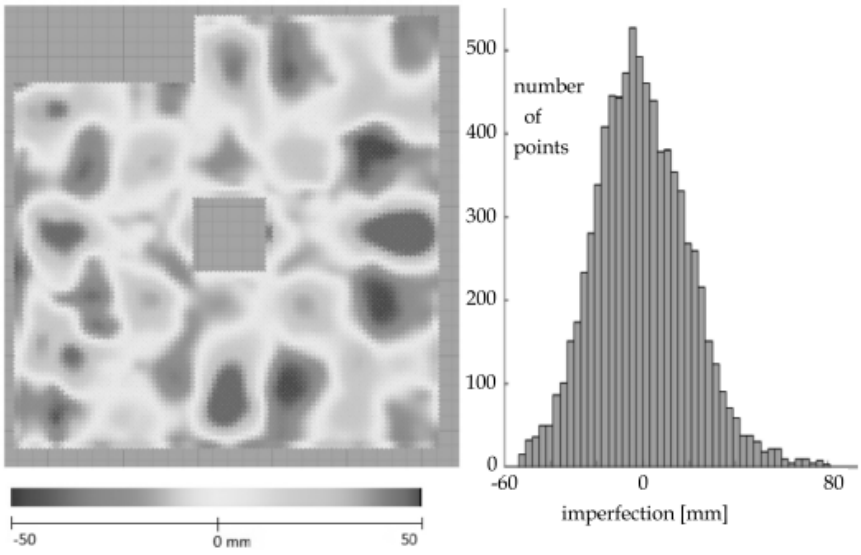

Figure 7. Shape imperfections in the roof of Heimberg swimming pool
Patch of 29 x 28 m; control point spacing of 0.25 and 5.00 m; largest observed imperfection amplitude is 80 mm

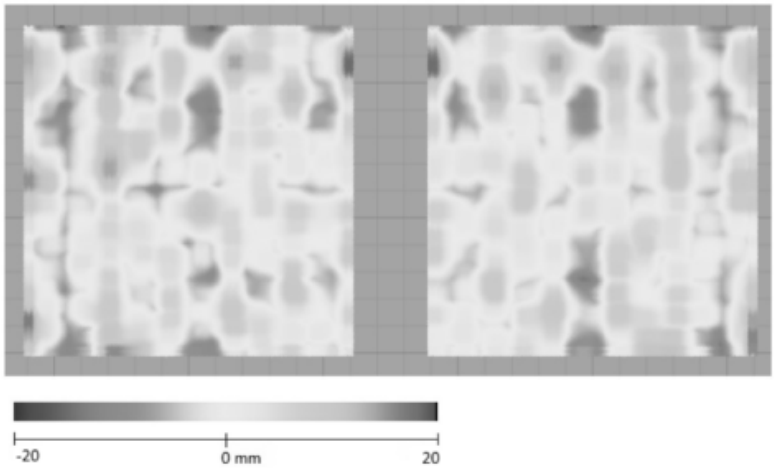

Figure 8. Shape imperfections in the roof of Heimberg tennis hall
Two patches of 12 x 12 m; control point spacing of 0.25 and 4.00 m; largest observed imperfection amplitude is 23 mm

Figure 24: Imperfection plots of scandata of the tennishall and swimmingpool. [1]

11.3 Appendix C

Variable p - full range

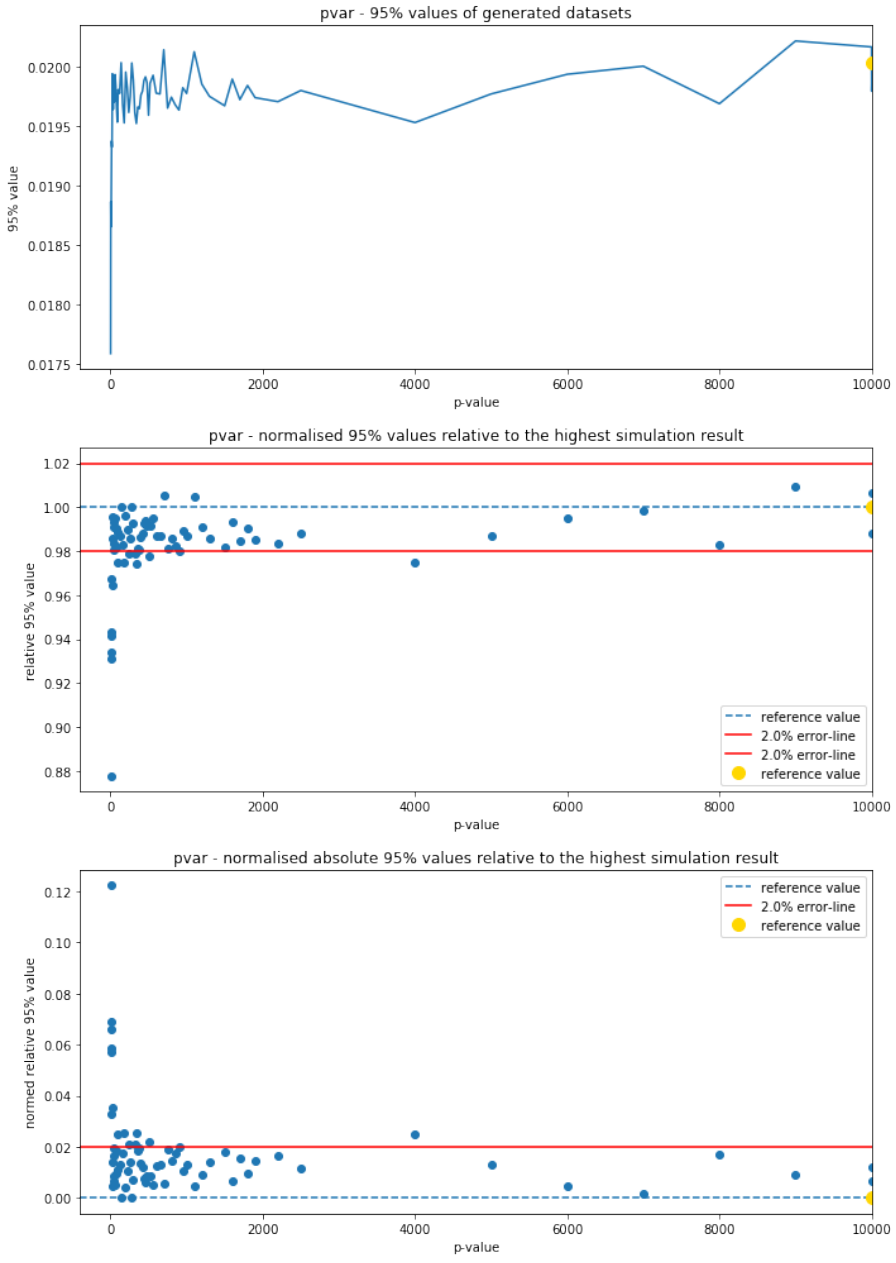

Figure 25: Plots of the simulation data for a variable p. Full range of p values. (a) actual 95% values. (b) Normalized 95% values. (c) normalized absolute 95% values.

Variable p - zoomed range

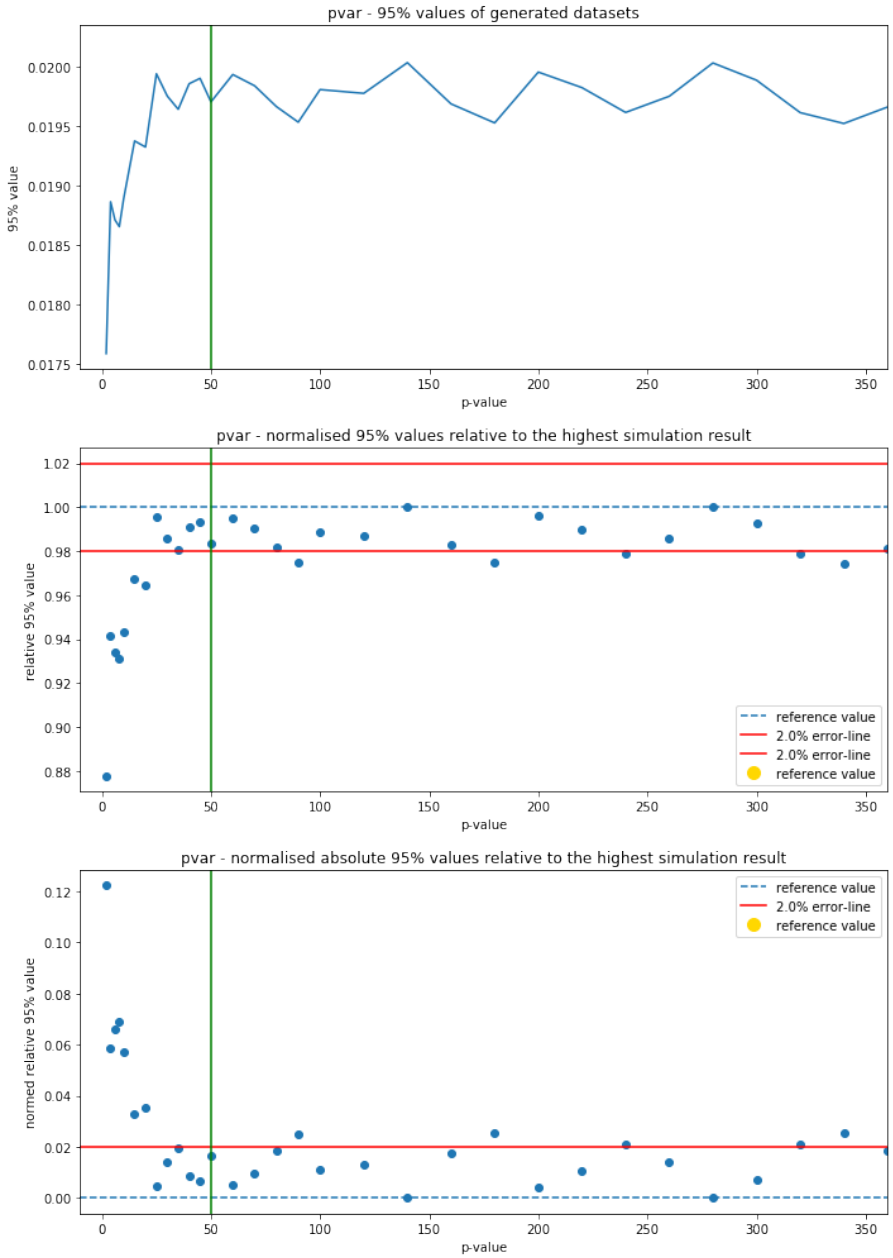

Figure 26: Plots of the simulation data for a variable p. Zoomed range of lower p values end where the optimum is expected to be. (a) actual 95% values. (b) Normalized 95% values. (c) normalized absolute 95% values.

Variable m - full range

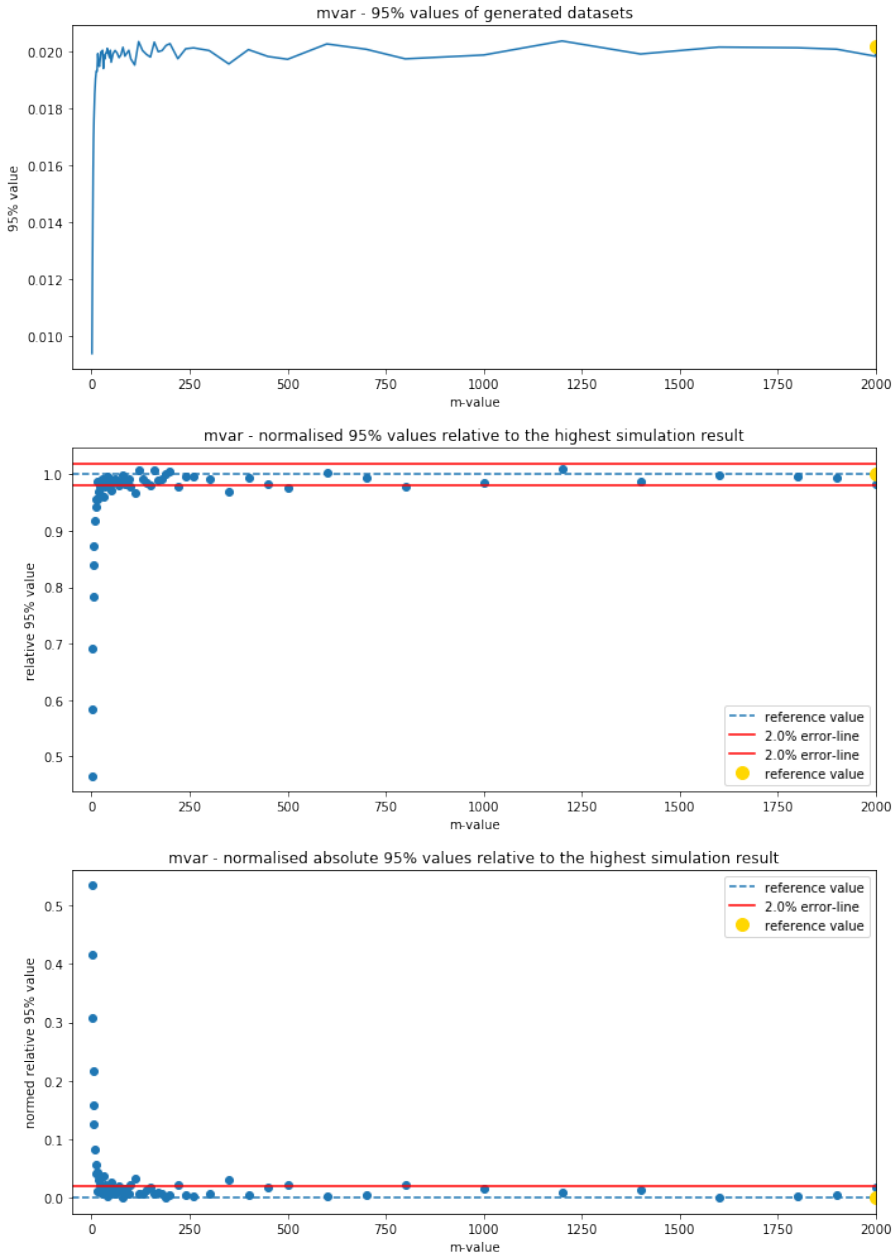

Figure 27: Plots of the simulation data for a variable m. Full range of m values. (a) actual 95% values. (b) Normalized 95% values. (c) normalized absolute 95% values.

Variable m - zoomed range

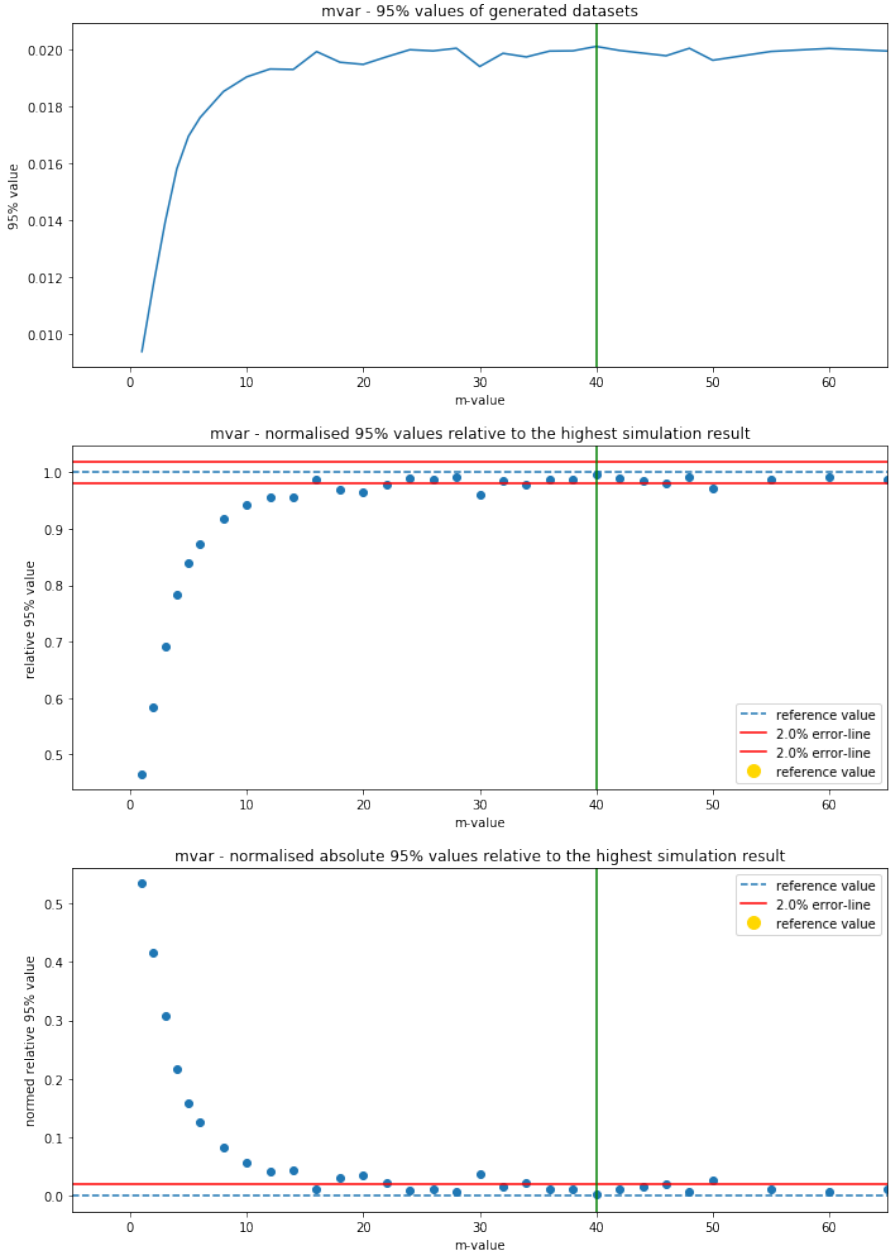

Figure 28: Plots of the simulation data for a variable m. Zoomed range of lower m values end where the optimum is expected to be. (a) actual 95% values. (b) Normalized 95% values. (c) normalized absolute 95% values.

11.4 Appendix D

```
# Python 3.6.1 (Anaconda 4.4.0)
# numpy==1.13.3
# matplotlib==2.1.1
# Scipy within anaconda package


import numpy as np
import numpy.random as rnd
import matplotlib.pyplot as plt
import scipy.stats


def impr_func(mean, varcof, s):

    while True:

        sigma = mean * varcof
        sample = rnd.normal(mean, sigma, s)

        samp_mean = np.mean(sample)
        samp_std = np.std(sample)
        samp_varcof = samp_std / samp_mean

        output = rnd.normal(samp_mean, samp_std, 1)

        if output > 0:
            return float(output)



def generate_field(A, l_hat, impr_mean, impr_varcof, s, p,
                   m, N, field_output_num):

    srA = int(np.sqrt(A))
    srp = int(np.sqrt(p))
    field = np.zeros((srA*srp, srA*srp))
    X_axis = np.linspace(0, (srA-(1/srp)), (srA*srp))
    Y_axis = np.transpose([X_axis])


    for i in range(1, m+1):

        impr = impr_func(impr_mean, impr_varcof, s)

        C = np.sqrt( (12 / (m - 1 / (s * m)) ) ) * impr

        l = ((2 * i - 1 ) / (2 * m)) * l_hat
        a = C * l

        d = a * np.sin( (np.pi * X_axis / l) + 2 * np.pi * rnd.rand()) * \
                   np.sin( (np.pi * Y_axis / l) + 2 * np.pi * rnd.rand())
        field = field + d

    if field_output_num == 0:
        return field
    if field_output_num == 1:
        if abs(np.amax(field)) < abs(np.amin(field)):
            return abs(np.amin(field))
        else:
            return abs(np.amax(field))



def plot_field(A, l_hat, impr_mean, impr_varcof, s, p, m, N):
    z = generate_field(A, l_hat, impr_mean, impr_varcof,
                       s, p, m, N, field_output_num=0)
    mv = max(abs(np.amax(z)), abs(np.amin(z)))
```

```
    plt.figure(figsize=(10,8))
    plt.imshow(z, interpolation='nearest', extent=[0,int(np.sqrt(A)),\
                                0,int(np.sqrt(A))],cmap='inferno')
    plt.title('Maximum absolute value:  '+str(mv)+'  m')
    plt.xlabel('[m]')
    plt.ylabel('[m]')
    plt.colorbar()
    plt.show()


def generate_dataset(A, l_hat, impr_mean, impr_varcof,
                     s, p, m, N, field_output_num):
    dataset = np.zeros(N)
    for i in range(N):
        dataset[i] = generate_field(A, l_hat, impr_mean, impr_varcof,
                                    s, p, m, N=[i], field_output_num=1)
    return dataset


def char_val(data, percentile):
    sorted_array = np.sort(data)
    x = len(sorted_array)
    y = int(np.floor(percentile * x))
    z = sorted_array[y]
    return z


def partial_factor(alpha, beta, mean, std):
    return (1 + alpha * beta * (std / mean)) / (1 + 1.64 * (std / mean))
# alpha and beta should be positive numbers


def plot_stat(data):
    data0_array = data

    plt.figure(figsize=(9, 8))
    plt.subplot(211)
    a = plt.hist(data0_array, bins=50, normed=True)
    x = np.linspace(min(data0_array), max(data0_array))
    y = scipy.stats.norm.pdf(x, np.mean(data0_array), np.std(data0_array))
    plt.plot(x, y, linewidth=4)
    plt.title('PDF')
    plt.xlabel('deflection [m]')
    plt.ylabel('normed PDF')

    plt.subplot(212)
    a = plt.hist(data0_array, bins=50, normed=True, cumulative=True,
                 histtype='step')
    x = np.linspace(min(data0_array), max(data0_array))
    y = scipy.stats.norm.cdf(x, np.mean(data0_array), np.std(data0_array))
    plt.plot(x, y)
    plt.title('CDF')
    plt.xlabel('deflection [m]')
    plt.ylabel('normed CDF')

    plt.tight_layout(h_pad=1.5)

    plt.show()

    mean = np.mean(data0_array)
    std = np.std(data0_array)
    th_char_val = np.mean(data0_array) + 1.64 * np.std(data0_array)
    sort_char_val = char_val(data0_array, 0.95)
    diff = np.round((sort_char_val - th_char_val) / th_char_val * 100, 4)
    # a positive difference means the real charval is higher then theoretical

    print('mean:    ', mean)
    print('std:    ', std)
    print('theoretical char_val:    ', th_char_val)
```

```
    print('sorted char_val:   ', sort_char_val)
    print('real vs theoretical char value', diff, '%')
    print('partial factor:', partial_factor(0.7, 3.6, mean, std))
    print(scipy.stats.mstats.normaltest(data0_array))
    print(scipy.stats.kstest(rvs=data0_array, cdf='norm',
                             args=(mean, std), N=10000))


def program():
    A_input = input('What is the value for A? ')
    A_input = float(A_input)
    l_hat_input = input('what is the value for l_hat? ')
    l_hat_input = float(l_hat_input)
    impr_mean_input = input('What is the value for impr_mean? ')
    impr_mean_input = float(impr_mean_input)
    impr_varcof_input = input('What is the value for impr_varcof? ')
    impr_varcof_input = float(impr_varcof_input)
    p_input = input('What is the value for p? p=100 is adviced. ')
    p_input = float(p_input)
    m_input = input('What is the value for m? m=50 is adviced. ')
    m_input = int(m_input)
    N_input = input('What is the value for N? N=10000 is adviced. ')
    N_input = int(N_input)

    plot_field(A=A_input, l_hat=l_hat_input, impr_mean=impr_mean_input,
               impr_varcof=impr_varcof_input, s=4, p=p_input, m=m_input, N=
                                                        N_input)

    data = generate_dataset(A=A_input, l_hat=l_hat_input, impr_mean=
                                          impr_mean_input,
                            impr_varcof=impr_varcof_input, s=4, p=p_input,
                            m=m_input, N=N_input, field_output_num=1)

    plot_stat(data)

    pass


program()
```

11.5 Appendix E

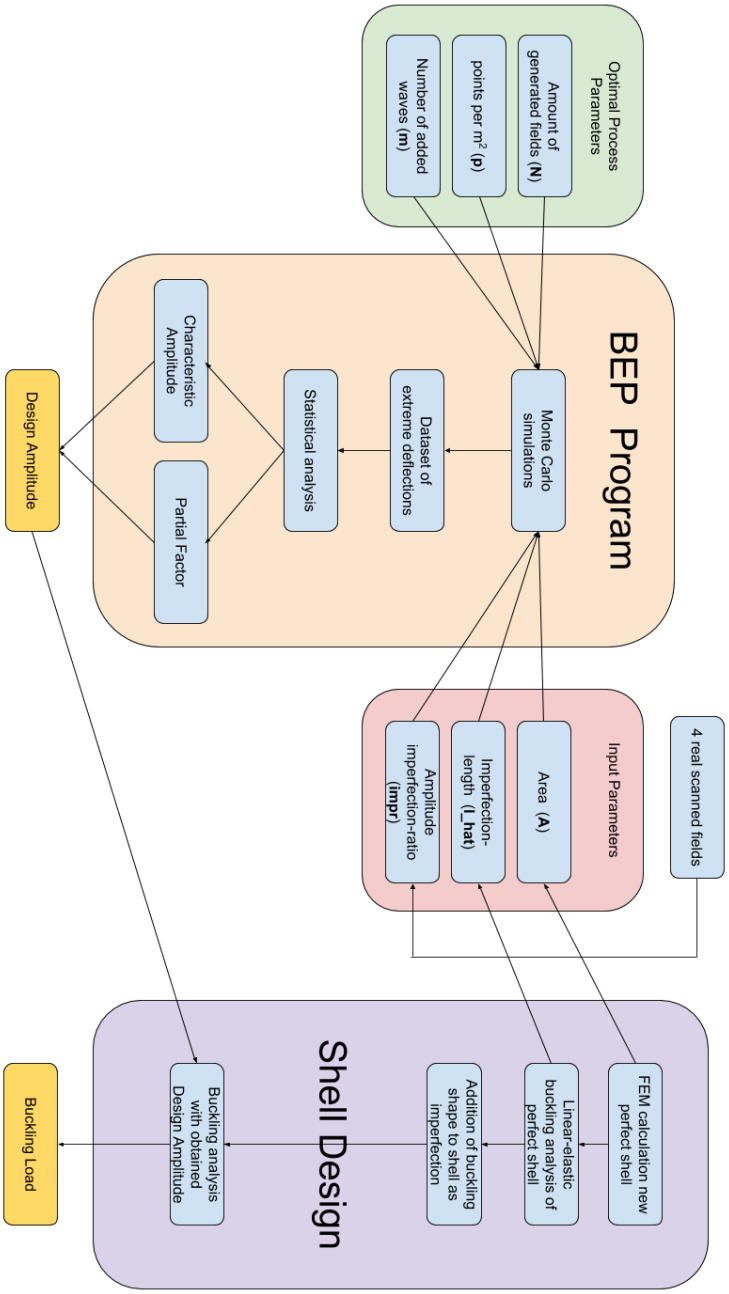

Figure 29: The role of the program within the design process of shell structures.

11.6 Appendix F

Updated version of the field generator code. Now with a distribution of impr instead of just a single value. This incorporates the variance of the scandata into the field generator.

```
import numpy as np
import numpy.random as rnd
import matplotlib.pyplot as plt
import scipy.stats


def impr_func(mean, varcof, s):

    while True:

        sigma = mean * varcof
        sample = rnd.normal(mean, sigma, s)

        samp_mean = np.mean(sample)
        samp_std = np.std(sample)
        samp_varcof = samp_std / samp_mean

        output = rnd.normal(samp_mean, samp_std, 1)

        if output > 0:
            return float(output)


def generate_field(A, l_hat, impr_mean, impr_varcof, s, p,
                   m, N, field_output_num):

    srA = int(np.sqrt(A))
    srp = int(np.sqrt(p))
    field = np.zeros((srA * srp, srA * srp))
    X_axis = np.linspace(0, (srA - (1 / srp)), (srA * srp))
    Y_axis = np.transpose([X_axis])

    for i in range(1, m + 1):

        impr = impr_func(impr_mean, impr_varcof, s)

        C = np.sqrt((12 / (m - 1 / (s * m)))) * impr

        l = ((2 * i - 1) / (2 * m)) * l_hat
        a = C * l

        d = a * np.sin((np.pi * X_axis / l) + 2 * np.pi * rnd.rand()) * \
            np.sin((np.pi * Y_axis / l) + 2 * np.pi * rnd.rand())
        field = field + d

    if field_output_num == 0:
        return field
    if field_output_num == 1:
        if abs(np.amax(field)) < abs(np.amin(field)):
            return abs(np.amin(field))
        else:
            return abs(np.amax(field))
```