

Jonas Ronglan Lindgård

# Parameter optimization of sheet piles at Drammen Hospital

June 2020





Norwegian University of  
Science and Technology

# Parameter optimization of sheet piles at Drammen Hospital

**Jonas Ronglan Lindgård**

Civil and Environmental Engineering

Submission date: June 2020

Supervisor: Arnfinn Emdal

Norwegian University of Science and Technology  
Department of Civil and Environmental Engineering



# Preface

This master's thesis has been carried out at the Department of Civil and Environmental Engineering at the Norwegian University of Technology and Science (NTNU) during the spring of 2020. The thesis is set at 30 credits.

The topic for the master thesis was suggested by Kristian Aunaas and Andreas Brathetland at Norconsult AS, Sandvika. I would like to thank my supervisor from NTNU, Arnfinn Emdal, for advising me on the contents of the thesis and helping to concretize the project. I would also like to thank Lars Gudmund Botnen at Norconsult, Ivan Dapina at NTNU and SINTEF, as well as Miquel Lahoz from Bentley for their advise on programming in Plaxis 2D.

Finally, a big thank you to Emil and Andreas for making the writing process in Emil's kitchen bearable during the corona period. Your support and companionship has been greatly appreciated.

Jonas Ronglan Lindgård

June 5, 2020



# Abstract

With the implementation of options to couple programming languages and geotechnical finite element method software, the potential for optimized design in geotechnical engineering has arguably never been greater. However, knowing the situations in which these possibilities are the most useful is not always clear. This thesis has attempted to answer this by using Python in PLAXIS 2D to model and cost optimize single anchor sheet piles at the new Drammen Hospital in Drammen, Norway. Along with the analysis itself, the general viability of modelling and analysis using Python in PLAXIS 2D for regular geotechnical engineering practice has been evaluated.

The study starts by looking at previous optimization studies within the field of geotechnical engineering, evaluating the different methods with regards to their benefits and drawbacks. Some general optimization theory has also been presented. For a comparative analysis, three optimization methods were selected: an optimization algorithm from a Python library, a simplistic optimization script and an automatized brute force method. The modelling and analysis of the profiles in the thesis has been done purely using Python scripting, in order to give a good idea of its potential in regular engineering practice. A focus throughout the study has been to look at where and when the use of scripting is the most effective.

The results from the study show that there is good potential for cost optimization using Python in PLAXIS 2D if the different construction costs are available to the engineer. The results indicate that finding a solution in the transition between the two most dominant failure mechanisms yields the most cost-efficient solutions.

Modelling using Python in PLAXIS 2D is found to have limited viability for regular practice in geotechnical engineering, as many of the actions, such as drawing irregular geometry, changing boundary and water conditions and creating materials are all easier done manually. Materials especially should always be created manually, as the available manual tools are doing it through scripting can be tedious and is prone to error messages. Overall, scripted modelling is found to have more drawbacks than possible advantages. That said, for more research-based projects, automatized modelling could still be highly useful.

Analysis using Python in PLAXIS 2D is found to have a lot of potential and great prospects. While there are a few drawbacks that must be accounted for, the possibilities offered by automatized analysis are found to outweigh these. Being able to run highly detailed analyses overnight is in itself thought to be very useful in terms of efficient analysis. Combining this with robust optimization scripts can allow for overall better designs and lower design costs. The benefits of cost-optimized design will increase as project size increases, making it particularly useful for large projects. In the case of small projects, the extra time spent optimizing design must be weighted against the potential gain.





# Contents

<b>Preface</b>	<b>i</b>
<b>Abstract</b>	<b>iii</b>
<b>1 Introduction</b>	<b>2</b>
1.1 Background . . . . .	2
1.2 Aim of Thesis . . . . .	2
1.2.1 Limitations . . . . .	3
1.3 Outline of Thesis . . . . .	3
<b>2 Optimal design in Geotechnical Engineering</b>	<b>5</b>
2.1 Optimization by domain . . . . .	5
2.2 Solid Isotropic Material with Penalization method (SIMP) . . . . .	6
2.2.1 Drawbacks of the SIMP-method . . . . .	8
2.3 Reliability-based design . . . . .	8
2.3.1 Motivation for using reliability-based design . . . . .	9
2.3.2 Uncertainties in Geotechnical engineering . . . . .	9
2.3.3 The performance function . . . . .	9
2.4 Robust geotechnical design . . . . .	11
2.4.1 Pareto frontier . . . . .	12
2.4.2 Knee point . . . . .	13
2.4.3 Difficulties with reliability-based design in geotechnical engineering . . . .	13
2.5 Simplified-robust geotechnical design . . . . .	13
2.5.1 Sensitivity Index . . . . .	14
2.6 The problem with robust design . . . . .	15
2.7 Cost efficient design using optimization algorithms . . . . .	15
2.8 The potential of Python as an efficient modelling and analysis tool in PLAXIS 2D	16
<b>3 Optimization Theory and Choice of Methods</b>	<b>18</b>
3.1 Optimization theory . . . . .	18
3.1.1 Optimization methods to be tested . . . . .	19
3.1.2 Choice of optimization algorithm . . . . .	20
3.2 Viable design space . . . . .	21
3.3 Sensitivity analysis . . . . .	22
<b>4 The Excavation at New Drammen Hospital</b>	<b>24</b>
4.1 The project . . . . .	24

4.2	The Area . . . . .	25
4.3	Site Investigations . . . . .	25
4.4	Chosen design . . . . .	27
<b>5</b>	<b>Modelling &amp; Framework</b>	<b>29</b>
5.1	Choice of soil models . . . . .	29
5.2	Choice of input parameters . . . . .	29
5.2.1	Hardening Soil Small . . . . .	29
5.2.2	NGI-ADP . . . . .	32
5.2.3	Sheet piles . . . . .	34
5.3	Modelling the zones in Plaxis 2D . . . . .	35
5.3.1	Rock contour . . . . .	35
5.3.2	Loads . . . . .	35
5.3.3	Meshing . . . . .	35
5.3.4	Ultimate limit state . . . . .	37
5.3.5	Phases . . . . .	38
5.4	Cost estimate of the anchored sheet pile system . . . . .	39
<b>6</b>	<b>Analysis and Results</b>	<b>41</b>
6.1	Optimization formulation . . . . .	41
6.2	Problems with the SLSQP algorithm . . . . .	41
6.3	Zone 1 . . . . .	42
6.3.1	Initial testing . . . . .	42
6.3.2	Optimization . . . . .	44
6.4	Zone 3 . . . . .	48
6.4.1	Initial testing . . . . .	48
6.4.2	Optimization . . . . .	49
<b>7</b>	<b>Discussion</b>	<b>54</b>
7.1	Found solutions . . . . .	54
7.1.1	Optimization scripts versus Automatized Brute Force . . . . .	55
7.2	Modelling with Python . . . . .	57
7.3	Analysis with Python . . . . .	58
<b>8</b>	<b>Conclusion and further work</b>	<b>60</b>
8.1	Conclusion . . . . .	60
8.2	Further work . . . . .	61
<b>A</b>	<b>Site investigations</b>	<b>64</b>
<b>B</b>	<b>Optimization script</b>	<b>75</b>
<b>C</b>	<b>Instructions for Python in PLAXIS 2D</b>	<b>79</b>
C.1	Structures . . . . .	81
C.2	Mesh . . . . .	84
C.3	Flow conditions and staged construction . . . . .	84

# List of Figures

2.1	Topology optimization of a design problem. From Pucker and Grabe (2011) . . .	5
2.2	Shape optimization of a design problem. From Pucker and Grabe (2011) . . . . .	6
2.3	Shape development after a) 1, b) 15, c) 50 and d) 500 iterations. From Pucker and Grabe (2011) . . . . .	7
2.4	Deformations a) before and b) after the shape optimization. From Pucker and Grabe (2011) . . . . .	8
2.5	Visualization of the reliability index. From Prästings (2019) . . . . .	10
2.6	Testing different designs for robustness and cost. Notice especially the jump in cost compared to robustness from design 7 to design 8. From Gong et al. (2017) .	12
2.7	Visualization of the Pareto frontier. From Juang et al. (2014) . . . . .	12
2.8	Determination of the knee point. From Juang et al. (2014) . . . . .	13
2.9	Shortest distance from the utopia point as the optimal design. From Gong et al. (2017) . . . . .	15
3.1	The progress of a trust-region algorithm. From <a href="http://www.applied-mathematics.net/optimization/optimizationIntro.html">http://www.applied-mathematics.net/optimization/optimizationIntro.html</a> . . . . .	20
3.2	Viable design space . . . . .	21
3.3	Illustration of an example path taken by the optimization algorithm . . . . .	22
4.1	Illustration of the new Drammen Hospital. Photo: LinkArkitektur/Helse Sør-Øst	24
4.2	The construction area of the new hospital. Snippet from Finn.no/kart . . . . .	25
4.3	Overview of the site investigations and zones for the excavation. From Norconsult.	26
4.4	Principle drawing of the chosen design. From Norconsult. . . . .	27
5.1	PLAXIS Models of zones 1 and 3 of the excavation . . . . .	35
5.2	Mesh quality for the two zones . . . . .	36
5.3	Ultimate limit states . . . . .	37
5.4	The phases used in the optimization process for both zones . . . . .	38
6.1	Change in safety factor with sheet pile length in zone 1 . . . . .	42
6.2	Change in safety factor with anchor length in zone 1 . . . . .	43
6.3	Change in failure mode for a 20m anchor as the sheet pile length decreases . . . .	43
6.4	Path of the optimization for different starting points for zone 1 . . . . .	44
6.5	Cost versus safety factor in the automatized brute force optimization for zone 1 .	45
6.6	Visualization of the lengths tested in the automatized brute force optimization for zone 1 . . . . .	45
6.7	Development of the failure mechanism as the anchor length increases . . . . .	47

6.8	Sheet pile movement in the failure mechanism of the final solution in zone 1 . . .	47
6.9	Change in safety factor with sheet pile length in zone 3 . . . . .	48
6.10	Change in safety factor with anchor length . . . . .	48
6.11	Development of the failure mechanism for 14m sheet piles as the anchor length increases . . . . .	49
6.12	Path of the optimization for different starting points for zone 3 . . . . .	49
6.13	Cost versus safety factor in the automatized brute force optimization for zone 3 .	50
6.14	Visualization of the lengths tested in the automatized brute force optimization for zone 3 . . . . .	51
6.15	Development of the failure mechanism as the system reaches the final solution . .	52
6.16	Sheet pile movement in the failure mechanism of the final solution in zone 3 . . .	52
7.1	Equal parameter range but with different amounts of available analyses . . . . .	57
A.1	Total sounding from borehole 24 in zone 3 . . . . .	65
A.2	Total sounding from borehole 54 in zone 3 . . . . .	66
A.3	Total sounding from borehole 69 in zone 1 . . . . .	67
A.4	Shear profile based on a CPTU test from borehole 24 in zone 3. Profile interpreted by Norconsult . . . . .	68
A.5	Shear profile based on CPTU tests from borehole 54 in zone 3. Profile interpreted by Norconsult . . . . .	69
A.6	Shear profile based on a CPTU in from borehole 69 zone 1. Profile interpreted by Norconsult . . . . .	70
A.7	Borehole profile for borehole 24 in zone 1 . . . . .	71
A.8	Borehole profile for borehole 24 in zone 1 . . . . .	72
A.9	Borehole profile for borehole 54 in zone 1 . . . . .	73
A.10	Borehole profile for borehole 69 in zone 1 . . . . .	74
C.1	Using the echo command to find plate input parameters . . . . .	83

# List of Tables

4.1	Soil materials in zone 1 . . . . .	26
4.2	Soil materials in zone 3 . . . . .	26
5.1	Soil parameters for the granular soils . . . . .	31
5.2	Soil parameters for the cohesive soils in the HSS model . . . . .	32
5.3	Soil parameters for the granular soils in the HSS model . . . . .	34
5.4	Input parameters for the sheet piles used in the excavation . . . . .	34
6.1	Final solutions for different starting points in zone 1 . . . . .	44
6.2	Final solutions for the automatized brute force optimization in zone 1 . . . . .	46
6.3	Final solutions for the two optimization methods in zone 1 . . . . .	46
6.4	Final solutions for the two optimization methods in zone 1 . . . . .	46
6.5	Final solutions for different starting points in zone 3 . . . . .	50
6.6	Final solutions for the automatized brute force optimization in zone 3 . . . . .	51
6.7	Final solutions found by the two optimization methods . . . . .	51
6.8	Maximum forces in the different components in zone 3 . . . . .	52



# Chapter 1

## Introduction

### 1.1 Background

The concept of optimized design has always been an area of focus in engineering. More efficient material usage, leading to lower costs, has always been a preferred design choice in any project, and the possibility for cost-efficient design has arguably never been greater than the last two decades. With a large amount of user-friendly, available, robust computational tools, avoiding overdimensioned designs and excessive material usage is highly feasible. Coincidentally, the desire of modern times for greener designs with low carbon footprints makes optimization more relevant than ever.

As noted in the prestudy to this thesis, a relatively new option has been introduced in the form of using scripts for modelling and analysis in geotechnical engineering. The coupling of programming languages with FEM-software offers huge potential in terms of optimization, but it remains slightly unclear in what situations and to what extent these new possibilities can be used most efficiently. This thesis will look to provide insight to the topic, specifically the coupling of Python with PLAXIS 2D.

In standard geotechnical engineering practice, analyses are usually run during the day, and one at a time. This can be tedious and inefficient if struggling to obtain a satisfactory safety factor, having to make small adjustments to the profile and also running the risk of missing a potentially more optimal solution. Having an automatized process able to be run overnight could therefore have big potential in terms of efficient analysis.

### 1.2 Aim of Thesis

The aim of this thesis is to evaluate the potential in using Python scripting in PLAXIS 2D for modelling and analysis of geotechnical problems, with a focus on optimization and cost-efficiency. The thesis will also aim to provide a cost optimized solution to the sheet pile design at Drammen Hospital. Summarized, the thesis will attempt to provide an answer to the following questions:

*What is the optimal solution of a single-anchor sheet pile wall in sandy and clayey ground conditions with regards to cost-efficiency? In general, what are the prospects of using Python scripting in PLAXIS 2D for optimized modelling and analysis?*

In order to answer the questions, a couple of objectives for the thesis have been established:

1. Perform a literature study of general optimization concepts in geotechnical engineering, looking at advantages and disadvantages
2. Present the situation at Drammen Hospital, including the thought-process behind the chosen design
3. Model the situation at Drammen Hospital and explain choice of material parameters, as well as establish the cost framework for the excavation
4. Analyse and optimize the design through Python scripting in PLAXIS 2D

### 1.2.1 Limitations

This thesis will only look at an optimized solution of a single anchor sheet pile wall. While looking at multiple design concepts would be interesting, it is considered more valuable to get a more in-depth understanding of the situation at hand.

## 1.3 Outline of Thesis

The thesis consists of 8 chapters. Chapters 2 and 3 take a look at optimization theory, with Chapter 2 focusing on previous optimization studies in geotechnical engineering and Chapter 3 looking at more generalized optimization theory.

Chapters 4 and 5 present the excavation at Drammen Hospital. Chapter 4 gives a brief overview of the project itself and explains the choice of design concept. Chapter 5 presents the models, explaining the choice of material parameters and model geometry, while also establishing a simple cost framework for the excavation.

Chapter 6 presents the results from the analyses, and Chapter 7 is a discussion of these results as well as the viability of Python as an efficient tool in PLAXIS 2D. Finally, Chapter 8 concludes the discussion and presents some ideas for further work.





## Chapter 2

# Optimal design in Geotechnical Engineering

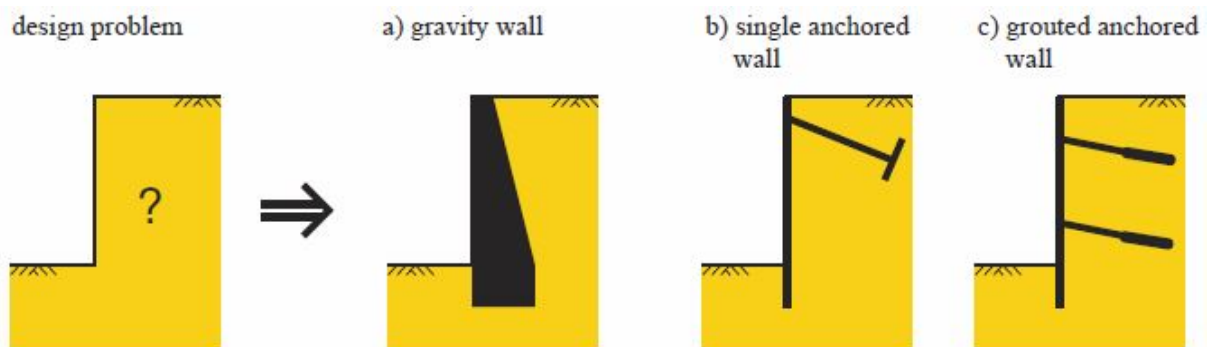
This chapter will take a look at some optimal design concepts in geotechnical engineering. Reliability-based design and uncertainty in geotechnical engineering is examined, along with the concept of robust geotechnical design and simplified robust geotechnical design. Finally, the concept of cost-efficient design using optimization algorithms and the potential of Python in PLAXIS 2D is discussed.

### 2.1 Optimization by domain

In order to discuss optimal design, it may be useful to establish some terminology for the different stages of optimization. Pucker and Grabe (2011) divide the optimization process into three domains: topology optimization, shape optimization and dimension optimization. The parts are done individually of each other, and once the part is set, it should remain unchanged during the rest of the optimization process.

#### Topology optimization

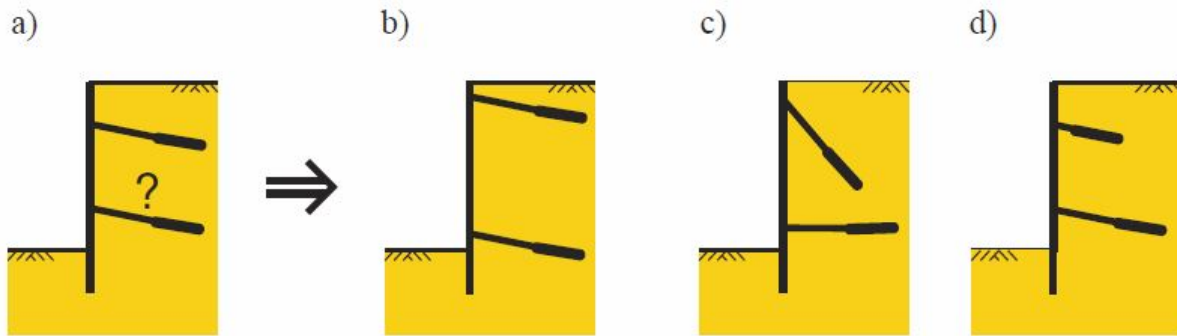
The topology is considered as the chosen solution to the problem itself. This form of optimization is used in the initial phase of the design to try finding the optimal material distribution within the design space available. Example of topology optimization can be seen in Figure 2.1 below.



**Figure 2.1:** Topology optimization of a design problem. From Pucker and Grabe (2011)

### Shape and dimension optimization

After the topology of the solution is chosen, the shape has to be optimized. This can be accomplished in numerous ways, therefore possibly presenting the biggest potential for change. For example, using a grouted anchor wall to reduce maximum wall deflection, one could vary the positioning of the anchors, the inclination of each anchor or simply the anchor lengths. Different shape designs can be seen in Figure 2.2.



**Figure 2.2:** Shape optimization of a design problem. From Pucker and Grabe (2011)

Shape optimization is usually the most time consuming when it comes to geotechnical engineering, due to all the possibilities and variables. However, the development of use-friendly FEM-software has markedly increased efficiency and helped speed up this process.

After topology and shape are selected, the dimensions of the components have to be optimized. Where the shape optimization is concerned with optimizing lengths, angles or positioning of different components, dimension optimization is about selecting the type sheet piles, anchors or other system components. This is usually given by the forces resulting from the shape and topology, and this form of optimization is therefore in most cases the most straightforward to implement.

## 2.2 Solid Isotropic Material with Penalization method (SIMP)

Pucker and Grabe propose using the SIMP method, initially proposed by Bendsøe and Kikuchi (1988), as a way to optimizing a footing for minimal deformations.

The SIMP method is already used for several engineering problems, for instance structural compliance, fluid-structure-interaction or thermoelectricity problems. The method uses an iterative process to find the optimal distribution of material within a design space  $\Omega$ . The algorithm is based on the idea that the material of the optimized structure already exists in the design domain  $\Omega$ , but is not optimally distributed. Therefore, the material is equally distributed in the design domain  $\Omega$  at the beginning of the optimization process. The material distribution changes during the optimization process and the material compacts in areas where it is needed to achieve the optimization task.

The design domain  $\Omega$  is discretized with finite elements. The material parameters are specified individually for each element depending on the material distribution. The virtual density  $\rho_e$  at a point  $a$  has to be between 0 and 1:

$$\rho_e(a) = \begin{cases} 0 & \rightarrow \text{no material} \\ 1 & \rightarrow \text{material} \end{cases}$$

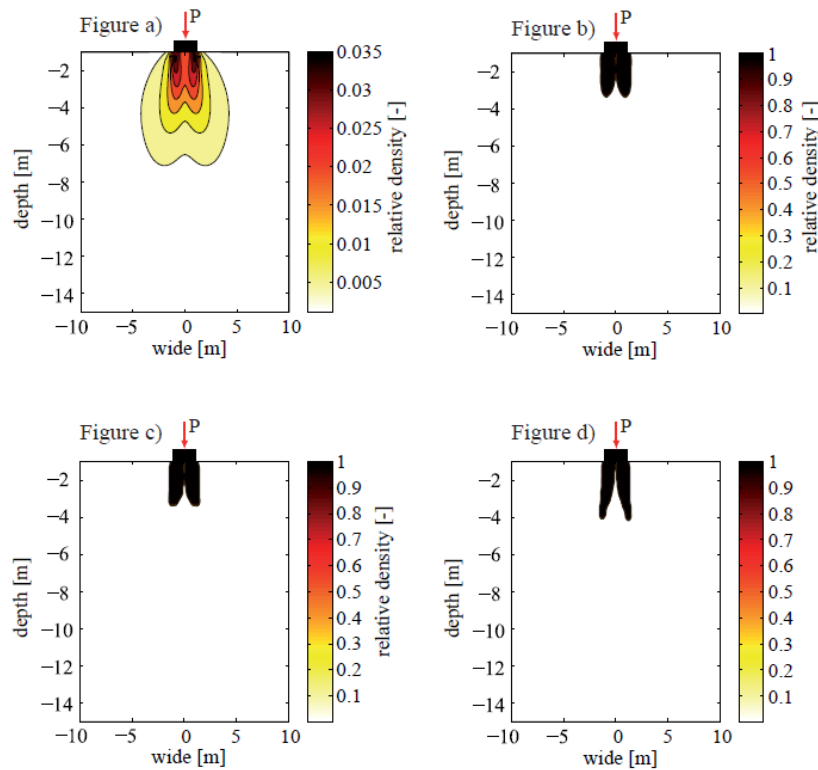
The material change-over is controlled by a penalty term  $p$ , which is commonly chosen to  $p = 3$ . In the element  $a$ , the Young's modulus  $E$  changes from  $E_1$  to  $E_2$  depending on the virtual density  $\rho_e$  during the optimization:

$$E_2 = E_1 \cdot (\rho_e)^p$$

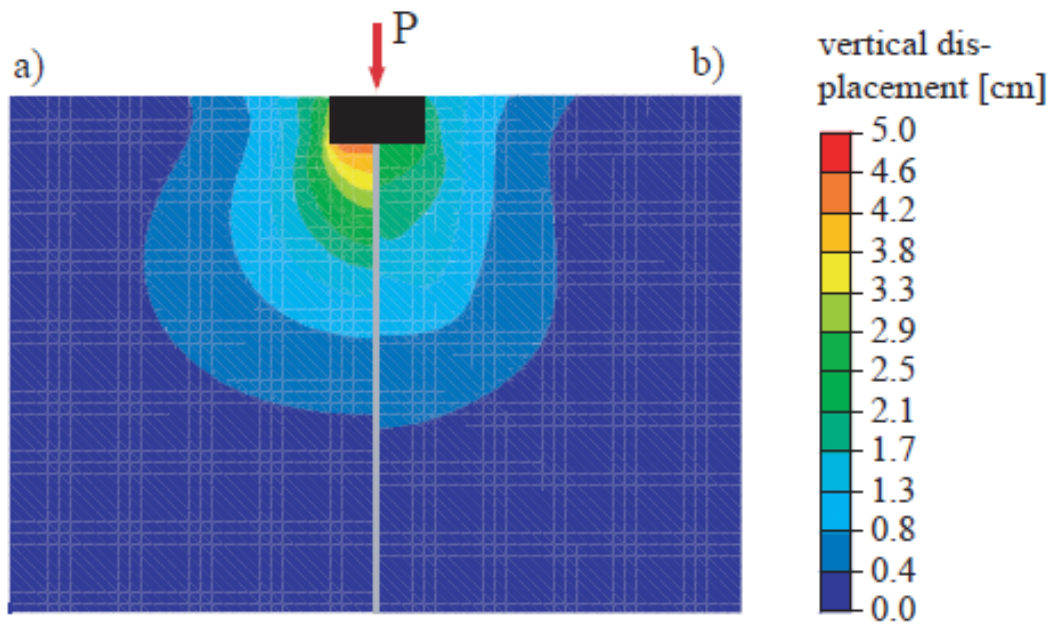
Using this change-over principle, the algorithm attempts to maximize the system stiffness. Each iterated solution must meet two criteria, the first being that the system must be in equilibrium an every step of the optimization process. A solution must also ensure that the volume of the material distributed in the design domain remains constant during the optimization process.

Seeing as a main assumption of the SIMP-method is a homogenized (and usually linear elastic) material, the method has to be adapted to function for the footing problem, both for the footing-soil interaction, as well as the non-linearity of the soil itself.

The result of the optimization process can be seen in Figure 2.3 below. The figure shows how the shape of the foundation is altered with increasing amounts of iterations. As the SIMP-algorithm discovers which parts of the foundation contribute to reducing settlement and which don't, the material is shifted in order to maximize efficiency.



**Figure 2.3:** Shape development after a) 1, b) 15, c) 50 and d) 500 iterations. From Pucker and Grabe (2011)



**Figure 2.4:** Deformations a) before and b) after the shape optimization. From Pucker and Grabe (2011)

As can be observed, the settlements are greatly reduced through the optimization, by about 50%, showing the potential of the method.

### 2.2.1 Drawbacks of the SIMP-method

While the SIMP-method appears to have potential in the design of footings, it still seems difficult to implement such a method efficiently for general geotechnical engineering, due to the complexity that is usually required in a solution. It is also a framework that is difficult to set up technically. Many geotechnical solutions also use several different materials, making setting up the framework even more difficult.

A final point, as Pucker and Grabe (2011) state themselves, is that these type of optimization algorithms also have the problem of not taking the practicability of the solution into account. This means many proposed optimizations may be much too expensive compared to the added benefit. Although this can be worked around by modifying the solution to something more technically feasible, such a solution will not be using the method to its full potential.

## 2.3 Reliability-based design

First seen in the 1950s, the concept of structural reliability attempts to express the safety of different structural design criteria through probability of failure. The probability of failure is defined as the probability of unsatisfactory performance, and can be applied to both serviceability limit states as well as ultimate limit states. The term *reliability* is basically the opposite to probability of failure, commonly defined as the probability of satisfactory performance.

### 2.3.1 Motivation for using reliability-based design

The main motivation for using reliability-based design is to take into account the uncertainties that occur in design parameters through probability rather than a fixed material factor (Prästings, 2019). The material factor is in many cases argued to have limited practicality, as it is unable to alter its value for varying uncertainty. This may be unfortunate, because large uncertainties in a project may require a material factor that exceeds the usual set value, or the material factor might be too conservative if the uncertainties are small.

### 2.3.2 Uncertainties in Geotechnical engineering

In all domains of civil engineering, there exists the need to deal with uncertainty. Load factors, material factors and safety factors are all frequently used in order to deal with a lack of knowledge and ensure that a design is safe.

However, within civil engineering, geotechnical engineers arguably face the most uncertainty of them all. Where the materials used in structural engineering, such as steel and concrete, have defined stiffness and strength parameters, a set of unique soil parameters must be determined for any geotechnical project at a new site. The strength of steel and concrete can certainly have slight variations that must be compensated using material factors, but overall these attributes are chosen to fit the structural design, whereas a geotechnical design is chosen to fit the attributes of the soil.

When the parameters of the "building materials" themselves have to be approximated as well, this means the overall solution faces many stages of uncertainty. Optimal design in geotechnical engineering should therefore look to take uncertainty into consideration while simultaneously producing the most cost-efficient solutions.

#### Aleatory versus epistemic uncertainty

Uncertainty in determining soil parameters is typically divided into two categories: aleatory and epistemic (Baecher and Christian, 2003). Aleatory uncertainty comes from the inherent randomness of a variable, for example wind or wave loads. The spatial variation of a soil parameter within a soil layer is also an aleatory uncertainty. An important fact for aleatory uncertainty is that it cannot be reduced or eliminated.

Epistemic uncertainty represents uncertainty due to lack of knowledge on the variable. Model uncertainty, measurement uncertainty and statistical uncertainty are all examples of epistemic uncertainty. Measurement errors inherent from equipment, operator and random testing effects, and statistical uncertainty arises because soil parameters are estimated from a limited set of data. Transformation uncertainty is introduced when measurements are transformed to the sought-after parameter from field and laboratory measurements of a property by site-specific or empirical transformation models (Prästings, 2019). Epistemic uncertainty can be reduced by collecting more data or improving measurement and transformation methods.

### 2.3.3 The performance function

The reliability design method is based on a defined *performance function* which compares the capacity  $R$  to the load  $S$ :

$$G(R, S) = R - S$$

or

$$G(R, S) = \frac{R}{S} - 1$$

A failure is defined as the performance function attaining a negative value:

$$P_f = P(G < 0)$$

The capacity and loads are calculated by describing the uncertain parameters using statistical distributions. The determination of the distributions itself is done through fitting the collected data to a matching distribution.

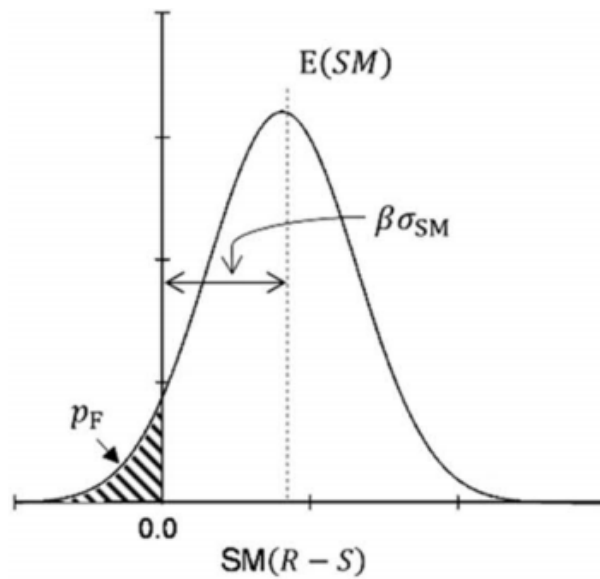
By describing the resistance and loads using statistical distributions, the probability of failure can be determined through a mean and a standard deviation of the performance function. This enables the analysis to take into account the uncertainties that are present in the parameters. In the simple case when the performance function  $G$  is normally distributed, the most common way of expressing this is using a reliability index,  $\beta$ , defined as:

$$\beta = \frac{\mu_g}{\sigma_g}$$

Where  $\beta$  then relates to the probability of failure by:

$$P_f = \Phi(-\beta)$$

An illustration of the reliability index can be seen in Figure 2.5.



**Figure 2.5:** Visualization of the reliability index. From Prästings (2019)

In a general case, the performance function may be nonlinear and not normally distributed. In this case, the failure probability or reliability index can be calculated by different numerical methods, such as the first order second moment-method (FOSM), the first or second order reliability-method (FORM/SORM) or Monte carlo-simulations (Prästings, 2019).

## 2.4 Robust geotechnical design

Proposed by Taguchi (1986), the concept of robust design revolves around minimizing the effect of uncertainties in a design process. Originally used for improving quality in industry engineering, the concept has since been adapted to other engineering fields, such as aviation (Paiva et al., 2014) and structural engineering (Doltsinis and Kang, 2004). The last couple of years, the concept has also made its way to geotechnical engineering (Juang et al., 2014)(Gong et al., 2014).

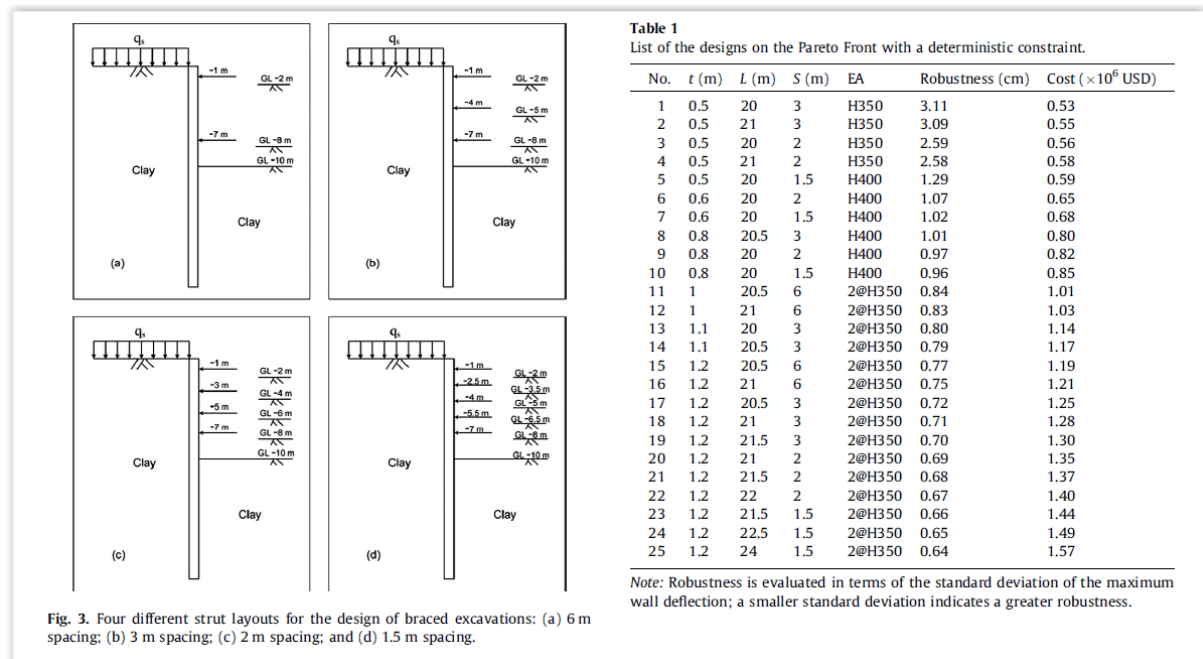
Robust geotechnical design (RGD) is a newer form of reliability analysis, which looks to provide a solution that is somewhat "desensitized" to possible parameter variability for the studied case. These parameters are divided into *easy-to-control* design parameters, such as the dimensions of different structural components and *hard-to-control* parameters (referred to as noise factors), such as soil parameters. A solution is considered robust if the system response varies little with noise factor variation. The goal of the RGD-method is therefore to find a safe and cost-efficient design, that is also robust against parameter variability.

For geotechnical engineering, parameter variability is as mentioned already present in a major way, as the determination of soil parameters always presents uncertainty. Because of this, the concept of robust design might become more relevant in geotechnics in the future. However, this will probably rely on the implementation of the concept becoming more available, as the process to set up the RGD framework can be quite technical and time-consuming. In addition, one could argue that the use of material, load and safety factors already contribute to making a design more conservative, although these factors as discussed are unable to implement improved measurements.

In their study of robust design of braced excavations in clay, Juang et al. (2014) measure the robustness of the system using the standard deviation of the maximum deflection of the excavation wall. A low standard deviation of wall deflection means the system is robust and will respond little to smaller changes in soil parameters. Multiple systems are tested with regards to the stability and serviceability requirements. The systems satisfying these requirements are then attempted optimized in terms of their response to uncertainty in the soil parameters.

The results from the study is displayed in Figure 2.6, showing how the cost changes with change in robustness. Most notable is the jump in cost compared to robustness from design 7 to design 8, and this jump is defined as a knee point. A plot of cost versus robustness as well as further explanations on knee points can be found in section 2.4.2.

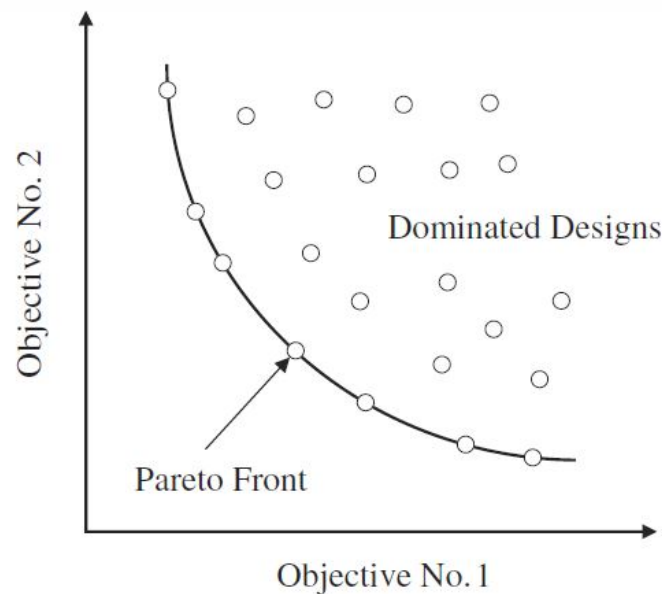




**Figure 2.6:** Testing different designs for robustness and cost. Notice especially the jump in cost compared to robustness from design 7 to design 8. From Gong et al. (2017)

### 2.4.1 Pareto frontier

In a problem with multiple, often conflicting objectives, obtaining a unique, fully optimized solution is highly unlikely. Instead, one will instead obtain a so-called Pareto frontier consisting of different optimized solutions. A Pareto frontier presents a set of optimal designs, in which further optimization of one objective cannot be accomplished without a sacrifice in another (non-dominated designs). A typical bi-objective Pareto frontier could consist of robustness versus cost. See Figure 2.7 below for a visualization of the Pareto frontier.



**Figure 2.7:** Visualization of the Pareto frontier. From Juang et al. (2014)

As can be observed from the figure, the Pareto frontier defines a trade-off relationship between the objectives, and it is up to the engineer to choose a solution that satisfies the desired cost or robustness level.

### 2.4.2 Knee point

If a requested solution has a set cost or a desired robustness level, picking the optimal solution will be a simple task, as a non-dominated design at the Pareto frontier that is the closest to the defined objective will be the most optimal. However, if such criteria are not specified, the decision is not as easy to make. For these cases, Juang et al. (2014) have suggested the use of a *knee point*, which presents the best compromise between design robustness and cost-efficiency, and therefore the preferred solution to a problem. See Figure 2.8 below for an illustration of the knee point.

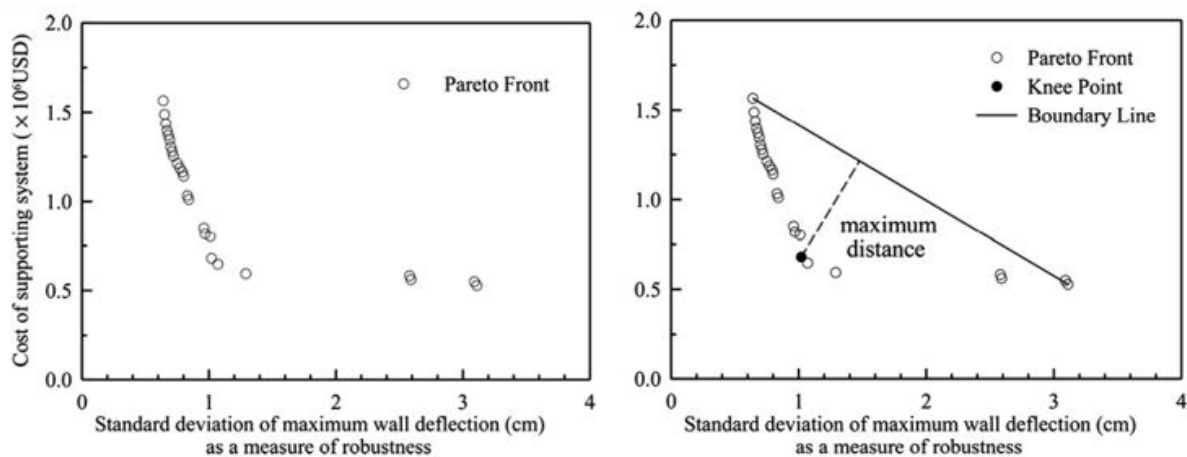


Figure 2.8: Determination of the knee point. From Juang et al. (2014)

As can be seen from the figure, when at the knee point, an increase in robustness will lead to a much higher cost, whereas a decrease in cost will lead to a drastic fall in robustness. As such, the knee point can be seen as the most optimal design.

### 2.4.3 Difficulties with reliability-based design in geotechnical engineering

In general, reliability analyses in geotechnical engineering can be problematic to implement. For example, the performance function describing the system response can be difficult to express mathematically due to complicated system response. In addition, creating representative distributions for the parameters require a lot of data, of which there may not be enough in smaller projects. Combined with the lack of available geotechnical reliability-based FEM-software, reliability based design might have trouble being implemented for commercial use in the near future.

## 2.5 Simplified-robust geotechnical design

As mentioned, while the RGD method is an interesting approach to account for uncertainties in geotechnical design, the technicality and time needed to set up the framework might make it difficult to implement efficiently in geotechnical firms. In addition, as mentioned, the

robust geotechnical design is based on a probabilistic approach that requires accurate statistical characterizations of the input parameters, to a degree that would seem unobtainable in most geotechnical problems due to constraints in budget.

With this in mind, Gong et al. (2017) presented the concept of simplified-robust geotechnical design, introducing several new ideas.

### 2.5.1 Sensitivity Index

Firstly, the simplified-RGD method adopts a new way of finding robustness, using the gradient of the system response to the noise factors in order to create a sensitivity index. A high sensitivity index equals a lower robustness, and vice-versa. The gradient of the system response is defined as (Gong et al., 2017):

$$\nabla f|_{\theta=\theta'} = \left\{ \frac{\partial f(\mathbf{d}, \boldsymbol{\theta})}{\partial \theta_1} \Big|_{\theta=\theta'}, \frac{\partial f(\mathbf{d}, \boldsymbol{\theta})}{\partial \theta_2} \Big|_{\theta=\theta'}, \dots, \frac{\partial f(\mathbf{d}, \boldsymbol{\theta})}{\partial \theta_n} \Big|_{\theta=\theta'} \right\} \quad (2.1)$$

Here the vector  $\mathbf{d}$  consists of the *easy-to-control* design parameters, such as anchor angle, anchor length and sheet pile length, and the  $\boldsymbol{\theta}$ -vector consists of the *hard-to-control* noise factors, such as friction angle. The derivatives are defined numerically as:

$$\frac{\partial f(\mathbf{d}, \boldsymbol{\theta})}{\partial \theta_1} \Big|_{\theta=\theta'} = \frac{f(\mathbf{d}, \theta'_1 + d\theta_1) - f(\mathbf{d}, \theta'_1 - d\theta_1)}{2d\theta_1} \quad (2.2)$$

To make the vector unitless, each derivative is multiplied with its corresponding value. The new vector is defined as the normalized gradient vector ( $\mathbf{J}$ ):

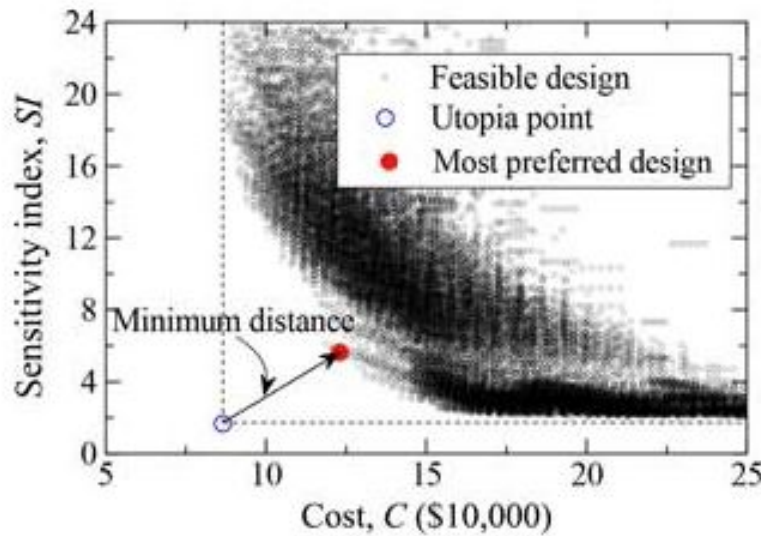
$$\mathbf{J} = \left\{ \frac{\theta'_1 \partial f(\mathbf{d}, \boldsymbol{\theta})}{\partial \theta_1} \Big|_{\theta=\theta'}, \frac{\theta'_2 \partial f(\mathbf{d}, \boldsymbol{\theta})}{\partial \theta_2} \Big|_{\theta=\theta'}, \dots, \frac{\theta'_n \partial f(\mathbf{d}, \boldsymbol{\theta})}{\partial \theta_n} \Big|_{\theta=\theta'} \right\} \quad (2.3)$$

Finally, the Euclidian norm of the normalized gradient vector, indicating the length of the vector, is defined as the *sensitivity index* of the design:

$$SI = \sqrt{\mathbf{J}\mathbf{J}^T} \quad (2.4)$$

This method of measuring robustness presents an important change to the standard RGD, as the method no longer requires statistical distributions of the input parameters, instead applying deterministic values directly.

Secondly, a new method of finding the knee point is presented, using a utopia point to find most optimal design, see Figure 2.9. This removes the need to generate a Pareto frontier, speeding up the optimization process markedly.



**Figure 2.9:** Shortest distance from the utopia point as the optimal design. From Gong et al. (2017)

The alterations introduced by Gong et al. (2014) contribute to a great extent in making simplified-RGD viable as a geotechnical design approach, by overcoming one of the big downsides of reliability based design: having to establish statistical distributions for the parameters. As any geotechnical design has to establish deterministic values for the input parameters, the method is readily available if the framework can be set up properly.

## 2.6 The problem with robust design

Unfortunately, as mentioned, most projects are run on a set budget, and this is usually the parameter of choice when selecting designs. Because of the high economic focus of construction projects, the optimization of most interest is arguably one that strictly minimizes cost while simultaneously satisfying any constraints set by the designer, such as maximum deformations or safety factor. Spending time to set up a framework for calculating robustness could be difficult to argue for, when a design already satisfies the criteria set by the contractor. In addition, the extra analyses that have to be run to generate the Pareto front is also likely to make the design process less efficient. While highly interesting, it may seem that the commercial market is not yet ready for concepts such as robust design.

## 2.7 Cost efficient design using optimization algorithms

Previous studies on this form of optimization is limited, but it does in theory have a big potential. Many programming languages have libraries containing numerous optimization algorithms for use in mathematical optimization problems. Coupled with FEM-analysis software, this should give the possibility to optimize design with regards to chosen parameters. If this coupling can be made work, it will provide a quick and easy optimization method for almost all cases in PLAXIS 2D.

This thesis will be examining the options available within Python, but other programming languages will also be able to perform the same type of optimization, as long as the coupling with

a FEM-analysis software can be made. Some relevant optimization algorithms will be further discussed in the following chapter.

## 2.8 The potential of Python as an efficient modelling and analysis tool in PLAXIS 2D

While the more advanced optimization techniques mentioned above might not be applicable in a commercial setting as of yet, the potential in including scripts for modelling and analysis could be considerable. As discussed in the prestudy to this thesis (Lindgård, 2019), modelling through scripting in PLAXIS 2D can be highly useful if many similar-looking cross sections have to be modelled. It allows for quick changes and alterations, creating a clean new model instead of using *move* commands, which in some cases can lead to slightly messy models.

However, using scripts for modelling does also have some drawbacks. Firstly, writing the script can at times be overly time-consuming to be worth the benefits. If a quick analysis is to be done on a simple cross section without much need for running multiple analyses, this is more conveniently done through the user interface in PLAXIS 2D. Additionally, creating irregular geometry can be tedious, so generating cross sections of highly irregular geometry might be more efficient when done manually. On the other hand, once a certain type of cross section (e.g. sheet pile wall, retaining wall) has been set up, the script can quite easily be tweaked and reused for other designs of a similar nature.

Using scripts for analysis is arguably where the biggest potential lies. While the benefits still have to be weighted against the extra time spent to set it up, the possibility of running analyses overnight means a high degree of efficiency can be obtained. This effect will be especially prominent if a single analysis takes a considerable amount of time, as it allows testing of multiple designs without the user having to pay attention to when an analysis finishes, in order to tweak the design parameters and rerun the analysis.

That said, for simpler analysis without much need for checking multiple designs or without much potential gain in optimizing design, running the analyses through the user interface is probably the better choice.



## Chapter 3

# Optimization Theory and Choice of Methods

This chapter will take a short look at general optimization theory. First, a generalized mathematical formulation of an optimization problem is introduced, and the choice of optimization methods is presented. Choice of optimization algorithm is then explained. Finally, the concept of viable design space and sensitivity analysis is demonstrated.

### 3.1 Optimization theory

Optimization theory has been of interest in all quantitative disciplines, from computer science and engineering to operations research and economics, due to its enormous potential. And in light of advances in computing systems, optimization techniques have become increasingly important and popular in all fields of engineering (Ding-Zhu Du, 2009).

According to Christensen et al. (2008), a general mathematical form of a structural optimization problem always consist of the following:

- *Objective function ( $f$ )*: A function used to classify designs. For every possible design,  $f$  returns a number indicating the goodness of the design. The objective function can be chosen to measure different objectives, such as displacement, stresses or cost of design.
- *Design variable ( $x$ )*: A function or vector that describes the design, and which can be changed during optimization. It may represent geometry or material choice.
- *State variable ( $y$ )*: For a given structure, i.e., for a given design  $x$ ,  $y$  is a function or vector that represents the response of the structure.

A general structural optimization (SO) problem will then take the form:

$$\text{SO} \left\{ \begin{array}{l} \text{minimize } f(x,y) \text{ with respect to } x \text{ and } y \\ \text{subject to } \left\{ \begin{array}{l} \text{behavioral constraints on } y \\ \text{design constraints on } x \\ \text{equilibrium constraint} \end{array} \right. \end{array} \right.$$

For this thesis, the objective function  $f(x,y)$  will be the cost of construction, and the aim of the optimization will be to minimize the cost while satisfying different constraints put on the model. The objective function will be given in more detail in Chapter 6.

The design variable  $x$  could theoretically contain every single design parameter, but in general increasing the amount of parameters will increase the computational time notably (although certain algorithms are tailor-made for handling a large amount of parameters). Therefore, neglecting parameters of little importance to the overall result will be important to help speed up the optimization process. Examples of possible design parameters in the studied case at Drammen Hospital include sheet pile length, sheet pile stiffness or anchor length.

The values of the state variable  $y$  will be given by PLAXIS 2D. Simulations in PLAXIS 2D will also ensure that equilibrium is satisfied.

### 3.1.1 Optimization methods to be tested

The optimization of the sheet piles at the new Drammen Hospital will be attempted in three ways: using an optimization algorithm from a publicly available Python library, a created optimization script as well as an automatized brute force method.

#### Optimization algorithm

Within the Python language there exist numerous libraries containing optimization algorithms. While these types of algorithms are normally used for purely mathematical expressions, they could in theory present themselves as good optimization tools in PLAXIS 2D if the coupling can be done correctly.

This thesis will be taking a look at the options available within the `scipy.optimize` library. Choice of optimization algorithm can be found in section 3.1.2.

#### User-created optimization script

In addition to the algorithm from the Python library, a simplistic optimization script has been written in order to compare results with the algorithm. The script is based on a sensitivity principle, coupling the change in safety factor to the change in cost in order to find the optimum step for each iteration. The script can be summarized as:

1. Calculate safety factor  $Msf_0$  for the current step
2. Altering one parameter at a time, calculate new safety factors  $Msf_i$
3. Use new safety factors to find change in safety factor per parameter,  $dMsf_i = Msf_i - Msf_0$
4. Use the cost function to find change in safety factor per change in cost,  $\frac{dMsf_i}{dC}$
5. The most optimal change is used for the next step

The optimization script in its entirety can be found in Appendix B.



### Automatized brute force

The final optimization method to be tested is an automatized brute force method. This method implies testing of a large amount of different designs in an automatized and systematic manner, and through this finding a suitable alternative. This means setting a range for each parameter, including the increment with which to increase/decrease the parameter within that range.

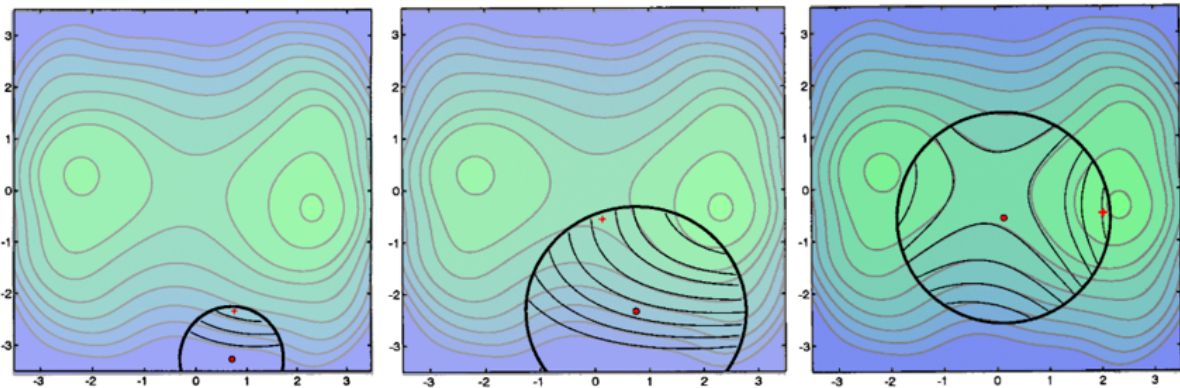
The automatized brute force method will be the most straightforward to implement, but should offer a decent alternative to setting up a more sophisticated optimization framework.

### 3.1.2 Choice of optimization algorithm

There exist an enormous amount of methods to optimize different problems, with gradient-based, gradient-free, unconstrained and constrained optimization algorithms all useful for their particular cases. For the optimization problem in this thesis, a cost optimization which satisfies a certain safety factor is sought after. This means that an optimization algorithm with the possibility of setting constraints is required.

In Python, the `scipy.optimize` library provides several commonly used optimization algorithms. Of the algorithms for constrained minimization, the Trust-Region Constrained algorithm, Sequential Least Squares Programming (SLSQP) algorithm and Constrained optimization by linear approximation (COBYLA) algorithm are all viable options.

The Trust-region Constrained algorithm works by first defining a region around the current best solution, in which a certain model (usually a quadratic model using a Taylor expansion) can to some extent approximate the original objective function. It then searches for a local minimum in the confidence region which becomes the new best solution.



**Figure 3.1:** The progress of a trust-region algorithm. From <http://www.applied-mathematics.net/optimization/optimizationIntro.html>

The COBYLA algorithm is a gradient-free algorithm, which also uses a trust region to find a minimum. Each iteration forms linear approximations to the objective and constraint functions by interpolation at the vertices of a simplex and a trust region bound restricts each change to the variables. However, unlike the Trust-region Constrained algorithm, the radius of the trust-region is never increased. Therefore, the algorithm can be inefficient for complex problems or problems containing a lot of variables (M.J.D., 1994).

Finally, the Sequential Least Squares Programming (SLSQP) algorithm solves a sequence of optimization subproblems, each of which optimizes a quadratic model of the objective subject to a linearization of the constraints. The algorithm is gradient-based, using a quasi-Newton method to calculate the Jacobian of the objective function.

As more in-depth knowledge of the algorithms go outside the scope of this thesis, they will not be further examined. Because of this, they will have to be treated as black boxes and the results must be tested accordingly.

After initial testing on mathematical expressions, it is found that the SLSQP algorithm needs the fewest function evaluations to converge for a simple mathematical case. This is also found through extensive testing by Schittkowski (Schittkowski, 1980) on a wide variety of test examples, who concludes that sequential quadratic programming methods is one of the most efficient algorithms when it comes to function evaluations and computational time. Seeing as each function evaluation equals a PLAXIS analysis, reducing this number should greatly reduce total computational time. The SLSQP algorithm is therefore selected for the optimization process.

### 3.2 Viable design space

For the optimization case at hand, any solution must lie within the viable design space as shown by Figure 3.2. The originally proposed design will naturally fall somewhere above the theoretically most cost-efficient design boundary and to the right of the safety factor line. If the solution finds itself outside this design space, it is either not satisfactory, unfeasible or both. The viable design space can in theory be increased to an infinite amount of dimensions, depending on the amount of constraints set on the system.

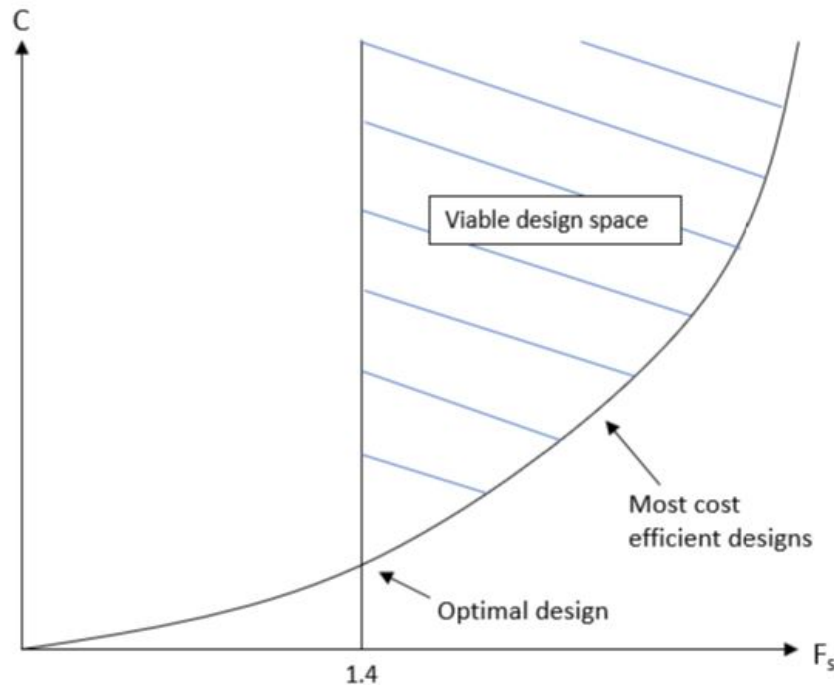
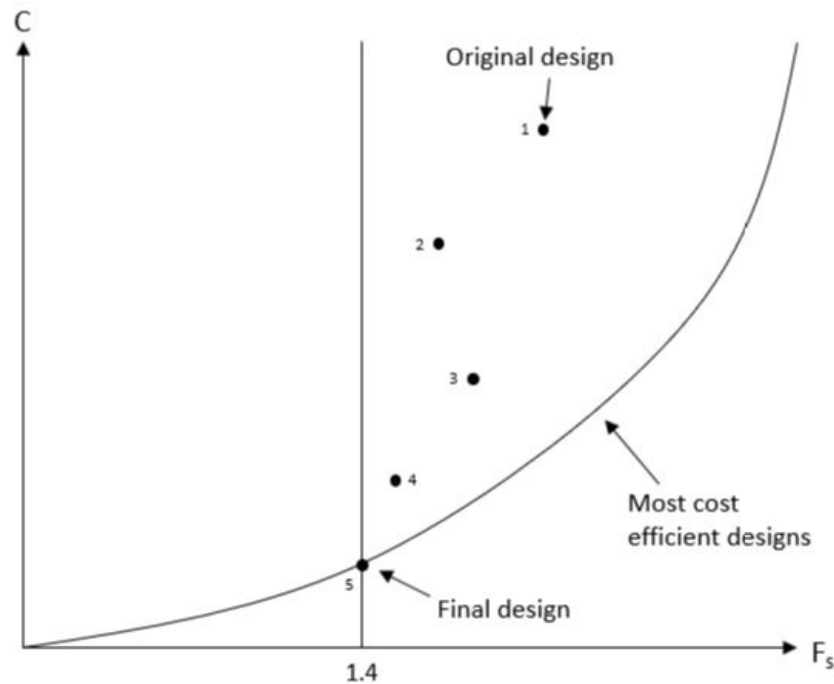


Figure 3.2: Viable design space

While there in theory are several designs satisfying a certain safety factor and other design constraints, they will not necessarily be the most cost-efficient. The aim of the optimization process will be to find the intersection point between the safety factor line and the cost-efficient design line, as shown by point 5 in Figure 3.3 below.



**Figure 3.3:** Illustration of an example path taken by the optimization algorithm

While the solution for a case of few design parameters could be done manually but tediously (brute force), once more parameters are added, this could become a difficult and time-consuming task. Optimization algorithms offer an automatized way of solving the issue.

### 3.3 Sensitivity analysis

Sensitivity analysis is a key concept within optimization theory, and is used in some way by virtually all forms of optimization algorithms. In general, sensitivity analysis is used in a wide range of fields, and is especially useful in the study and analysis of a “Black Box Process”, where the exact correlation between input and output is unknown. A typical example here is weather models, which can be very complex, but it can also be applied to many finite element models in engineering, where the system responses can be complicated.

A sensitivity analysis determines how different values of an independent variable affect a particular dependent variable under a given set of assumptions. By performing sensitivity analyses for relevant parameters, it becomes possible to evaluate the effect a change in input will have on the computational result, which essentially is the criteria used when algorithms choose which paths to take. A sensitivity analysis makes use of derivatives, either in an analytical form, or numerically in the form of finite differences.



## Chapter 4

# The Excavation at New Drammen Hospital

This chapter will give information and describe the situation at the location of the new hospital in Drammen. It will take a look at the site investigations performed in the area, as well as describe the choice of topology for the sheet pile wall of the excavation.

### 4.1 The project

The construction of the new hospital in Drammen started the 14. October 2019, and is planned to finish in 2024. The hospital will have integrated premises for somatics, mental health care and multidisciplinary specialized drug treatment, as well as radiation therapy (VestreViken, 2019). The hospital will be replacing the old Drammen hospital as well as Blakstad hospital.



**Figure 4.1:** Illustration of the new Drammen Hospital. Photo: LinkArkitektur/Helse Sør-Øst



## 4.2 The Area

Brakerøya, the area in which the hospital is to be constructed, is located next to the E18 road running through Drammen, as well as Brakerøya Train Station. The area has previously been used for small-scale industry and smaller businesses.



**Figure 4.2:** The construction area of the new hospital. Snippet from Finn.no/kart

As can be seen from Figure 4.2, the construction site is situated right next to the estuary of Drammen River. The groundwater in the area is therefore expected to be highly influenced by the water level of the river.

## 4.3 Site Investigations

Using data from site investigations over three periods in 2018-2019, the ground conditions at Brakerøya have been analyzed and determined, and the results summarized in a data report (Hult, 2019).

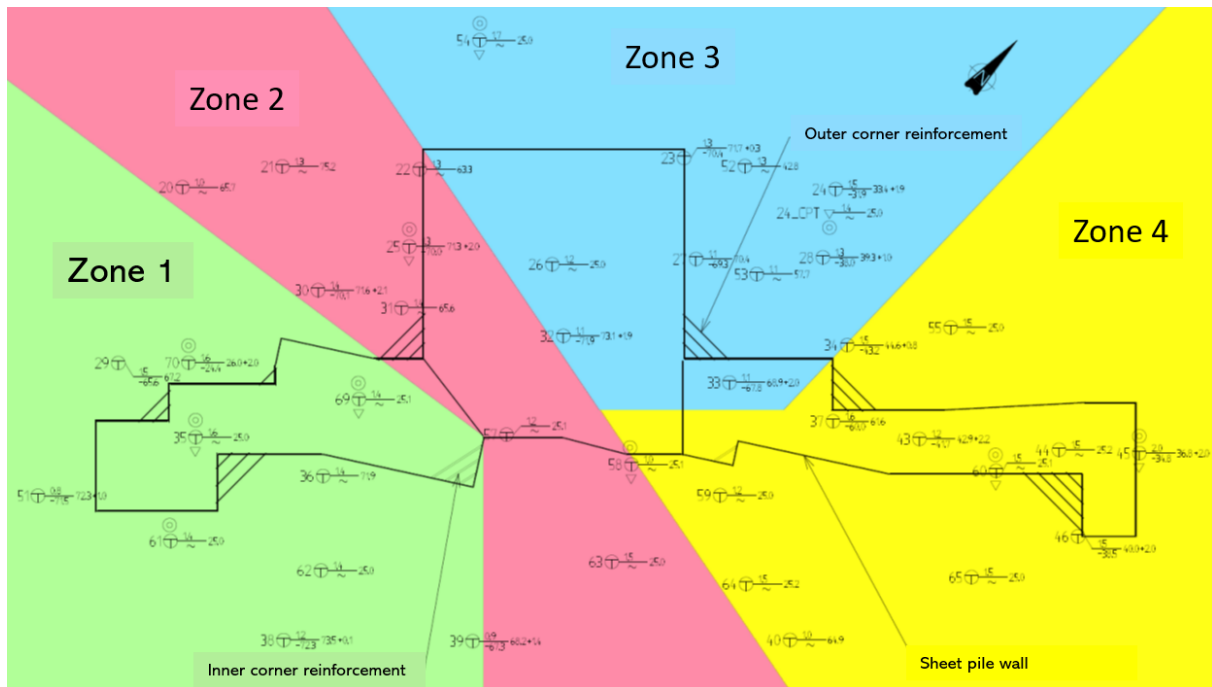
The contour of solid rock at Brakerøya is varied, with heights ranging from -21,3 meters above sea level in some areas to almost -72 meters above sea level in the area where the majority of the hospital is to be constructed.

Total soundings done in the eastern half of the hospital construction area show high resistance in the upper 1-3 meters, increasing to upwards of 11 meters in the western half. Below these layers, the soundings show a more constant resistance all the way from -55 to -70 meters above sea level, before being stopped for the deep areas or solid rock being assumed for the shallower areas.

Field samples analyzed in laboratory show varied results when it comes to soil compositions. Tests collected from the southern part of the site show varying layers of clay, gravel, silty sandy materials and sand down to around -12 meters, followed by clay all the way to the lowest sample at -19 meters, with no quick clay detected. The western area has mostly gravel and sand, with

some organic material, down to -7 meters, followed by silty clay and clay to the lowest sample at -22 meters. The eastern area 4 meters of mostly clayey sand and silty clay over soft clay down to the lowest sample, collected at -22 meters. The northern area of the site consists almost exclusively of clay from -3 all the way to the lowest sample at -22 meters.

Based on the data from the site investigations, the ground conditions at Drammen Hospital are divided into 4 zones, each with differing amounts and types of soil materials. An overview of the site investigations and zones for the excavation can be seen in Figure 4.3.



**Figure 4.3:** Overview of the site investigations and zones for the excavation. From Norconsult.

In this thesis, zones 1 and 3 will be analyzed. Zone 1 is characterized by approximately 11 meters of varying layers of sandy and sandy, silty, clayey materials, over roughly 20 meters of clay. Zone 3 consists of silty clay only, with quantities of up to 30 meters. The upper layer of clay in zone 3 is firm, but from roughly -8.5 meters the clay becomes softer. An overview of the assumed soil materials for different depths can be seen in Table 4.1 and Table 4.2.

**Table 4.1:** Soil materials in zone 1

Height (masl)	Soil material
+1.7 → +0.5	Fill material
+0.5 → - 2.0	Sandy, silty, clayey material
- 2.0 → - 4.0	Sand
- 4.0 → - 8.0	Sandy, silty, clayey material
- 8.0 → - 10.0	Sand
- 10.0 → - 30.0	Clay

**Table 4.2:** Soil materials in zone 3

Height (masl)	Soil material
+1.7 → - 1.0	Fill material
+0.5 → - 8.5	Clay 1
-8.5 → - 30.0	Clay 2

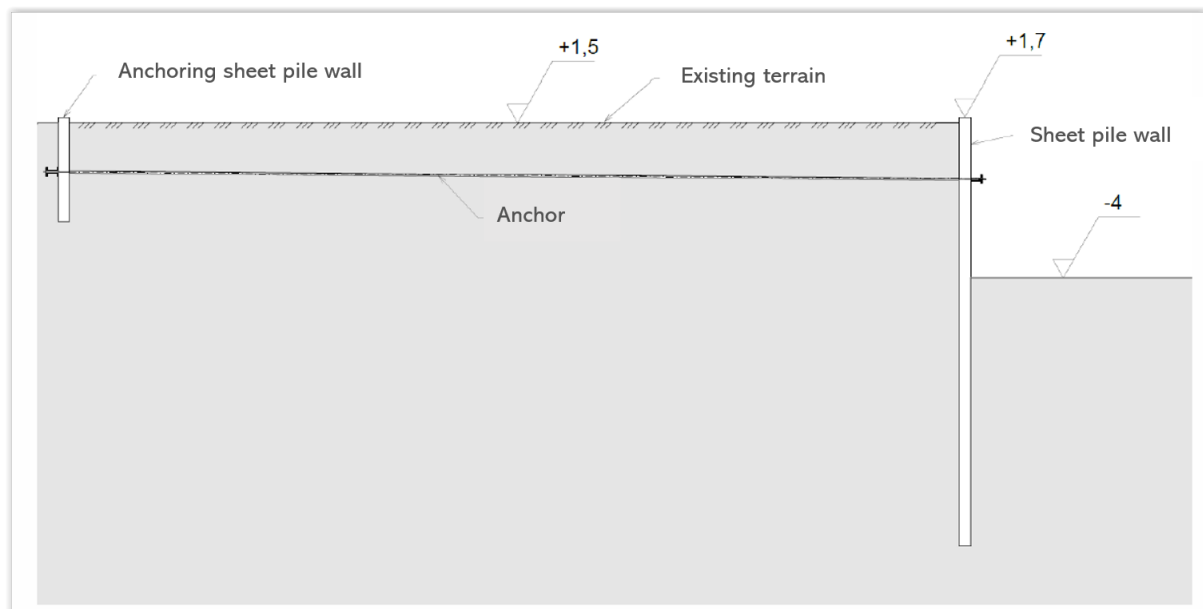
The groundwater in the area is largely affected by Drammen River, which is located near the construction site. As the upper soil layer in some areas of the construction site mainly consist of

sandy materials, such as zone 1, the groundwater level could here be expected to vary considerably throughout the year. For the more clay-dominated areas, such as zone 3, the groundwater level be expected to change more slowly. An average expected groundwater level is set at +0.5 meters above sea level.

## 4.4 Chosen design

A few different topologies were considered as a solution to the excavations. However, the availability of large amounts of space behind the sheet piles, as well as considerations of the practicality of pile driving later in the project, made single anchor sheet piles seem the most efficient choice.

In this type of solution, the main sheet pile wall of the excavation is supported by a smaller anchoring sheet pile wall further back. The two sheet pile walls will be connected through a horizontal 7-wire anchor. Each anchor is to be installed with a center distance of 4.2 meters. An illustration of the design can be seen in Figure 4.4.



**Figure 4.4:** Principle drawing of the chosen design. From Norconsult.

The solution is established in several steps. First, the sheet piles of the excavation are established, along with the supporting sheet pile wall. The soil is then excavated to 0.5 meters below where the anchor is to be established, and the anchor is drilled through and fastened in the supporting sheet piles. Finally, the remainder of the soil is removed. As seen from the illustration, the excavation will be close to 6 meters deep when complete.

Due to the large amount of open space around the excavation, no serviceability or deformation criteria have to be met for the excavations in the project.





## Chapter 5

# Modelling & Framework

This chapter presents the modelling and economic framework of the sheet piles. Choice of soil models and input parameters for the granular and cohesive materials are explained, and relevant loads around the excavation are considered. Mesh size and overall mesh quality for the two zones are discussed and the phases utilized are presented. Finally, the estimated cost function for the excavation is established.

### 5.1 Choice of soil models

The two zones, presented in Chapter 4, are each analyzed using a Hardening Soil Small (HSS) soil model as well as an NGI-ADP soil model. The reason for using the NGI-ADP model is that the HSS model cannot model the anisotropic strength properties of clays, and will give an unrealistically high safety factor. The NGI-ADP model can model the anisotropic properties and is therefore used to give a more realistic safety factor. However, the HSS model gives a larger estimate of the forces in the support system, making this model more conservative when selecting the dimensions of the support system.

### 5.2 Choice of input parameters

The input parameters for the soils have been selected based on a combination of site investigations and previous experience. As seen in section 4.3, the two zones have quite different amounts and types of soils, and the input parameters have to be determined using the site tests from that particular area.

#### 5.2.1 Hardening Soil Small

All the soils are modelled using the HSS model, for the reasons described in section 5.1. However, as the NGI-ADP model is used for cohesive soils only, the granular soils are modelled exclusively with an HSS model. The granular soils are here defined as all soils in the two profiles except the clay. These materials have a relatively large fraction of coarse materials, and there is not expected to be any build-up of excess pore pressure in these layers.

The material parameters for the granular soils are chosen based on empirical values. The strength and index parameters for the granular soils are based on values obtained from Figure 2.39 in

the Norwegian Public Road Administration's (NPRA) Manual V220 (2018), which contains recommended soil parameters for the design of retaining walls. The stiffness parameters of the granular soils are based on Table 16.2 from Brinkgreve (2019), which proposes parameters for sand with low and high relative density.

The material parameters for the cohesive soils are chosen based on a combination of empirical values and site investigations. In general, empirical values have been used to determine stiffness parameters, while site investigations have been used for strength and index parameters.

Below are descriptions of how certain key parameters were selected in the HSS models. The full list of selected parameters can be seen in tables 5.2 and 5.1. For the field sample laboratory analyses used in the choice of some of the parameters, see appendix A.

### Unit weight

The unit weight for the granular soils is chosen from empirical values in the NPRA Manual V220 (2018). The unit weights for the cohesive soils are based on laboratory tests on field samples. All unit weights fall in the range of  $\gamma = 18 - 19 \text{ kN/m}^3$ .

### Stiffness

The stiffness of the granular soils are chosen to represent the different coarse materials. The fill materials are expected to have a high stiffness, and a value of  $E_{50}^{ref} = 40000 \text{ kPa}$  is selected. The stiffness of the sand is chosen to represent a sand moderately stiffer than loose sand at  $E_{50}^{ref} = 25000 \text{ kPa}$ , and the silty, clayey sand layer slightly less stiff than a loose sand,  $E_{50}^{ref} = 15000 \text{ kPa}$ . The unloading-reloading stiffness  $E_{ur}^{ref}$  for the granular soils is selected following the rough estimate (Nordal, 2019):

$$E_{unloading} \approx 3 \cdot E_{loading} \quad (5.1)$$

The stiffness for the clays are calibrated through oedometer tests performed on field samples from the relevant zones. The tests indicate medium stiff clays for zone 1 at  $E_{oed}^{ref} = 2000 \text{ kPa}$  and slightly stiffer clays in zone 3 at  $E_{oed}^{ref} = 3000 \text{ kPa}$ . The clays display notably higher unloading-reloading stiffnesses, with  $E_{ur}^{ref} = 14000 \text{ kPa}$  in zone 1 and  $E_{ur}^{ref} = 15000 \text{ kPa}$  in zone 3.

### Friction angle

The friction angle for the granular soils are all above  $30^\circ$ , determined through the NPRA Manual V220 (2018). The friction angle for the cohesive soils are determined through laboratory testing, especially the use of triaxial tests. The clays in zone 1 show a low friction angle at  $\phi = 23^\circ$ , whereas the friction is markedly higher in zone 3 with  $\phi = 30 - 31^\circ$ .

### Dilatancy angle

The dilatancy angle of the materials is found as described by Nordal (2019):

$$\psi = \phi - \phi_i \approx \phi - 30 \quad (5.2)$$

Where  $\phi_i$  is the internal friction of the soil, around  $30^\circ$  for quartz sand commonly found in Norway.

### Coefficient of earth pressure

Because the clays in the area are considered close to normally consolidated, the coefficient of earth pressure at rest  $K_0^{nc}$  is calculated through Jaky's formula as described by Brinkgreve (2019):

$$K_0^{nc} = 1 - \sin(\phi) \quad (5.3)$$

Where  $\phi$  is the friction angle of the soil.

### Overview of HSS parameters

The following section gives an overview of the parameters utilized in the Hardening Soil Small models, both for the granular soils as well as the cohesive materials. The granular soils are found in table 5.1 and the cohesive materials in table 5.2. Zone 1 is modelled with two clay zones in the HSS model in order to better represent the laboratory results.

**Table 5.1:** Soil parameters for the granular soils

	Unit	Fill materials	Sand	Sandy, silty, clayey soil
Material model	—	HS small	HS small	HS small
Drainage type	—	Drained	Drained	Drained
$\gamma_{unsat}$	$kN/m^3$	19.0	18.0	18.0
$\gamma_{sat}$	$kN/m^3$	19.0	18.0	18.0
$E_{50}^{ref}$	$kN/m^2$	40000	25000	15000
$E_{oed}^{ref}$	$kN/m^2$	40000	25000	15000
$E_{ur}^{ref}$	$kN/m^2$	120000	75000	45000
Power (m)	$kN/m^2$	0.5	0.5	0.5
c'ref	$kN/m^2$	1	1	1
$\phi$	$^\circ$	36	33	32
$\psi$	$^\circ$	6	3	2
$\gamma_{0.7}$	—	0.0001	0.0001	0.0001
$G_o^{ref}$	$kN/m^2$	100000	75000	19000
$\nu'_{ur}$	—	0.2	0.2	0.2
$p_{ref}$	$kN/m^2$	100	100	100
$K_0^{nc}$	-	0.41	0.46	0.47
$k_x$	m/day	0.6	0.2624	0.2624
$k_y$	m/day	0.6	0.2624	0.2624
$R_{inter}$	—	0.5	0.5	0.5
$K_0$ determ.	—	Automatic	Automatic	Automatic
OCR	—	1.2	1.2	1.2

**Table 5.2:** Soil parameters for the cohesive soils in the HSS model

	Unit	Clay 1 Zone 1	Clay 2 Zone 1	Clay 1 Zone 3	Clay 2 Zone 3
Material model	—	HS small	HS small	HS small	HS small
Drainage type	—	Undrained (A)	Undrained (A)	Undrained (A)	Undrained (A)
$\gamma_{unsat}$	$kN/m^3$	18.5	18.5	18.0	18.0
$\gamma_{sat}$	$kN/m^3$	18.5	18.5	18.0	18.0
$E_{50}^{ref}$	$kN/m^2$	3000	3000	5000	5000
$E_{oed}^{ref}$	$kN/m^2$	2000	2000	3000	3000
$E_{ur}^{ref}$	$kN/m^2$	14000	14000	15000	15000
Power (m)	$kN/m^2$	1.0	1.0	0.6	0.6
c <sub>ref</sub>	$kN/m^2$	6.0	0.0	9.0	1.0
$\phi$	$^\circ$	23	23	30	31
$\psi$	$^\circ$	0	0	0	0
$\gamma_{0.7}$	—	0.22E-3	0.22E-3	0.22E-3	0.22E-3
$G_o^{ref}$	$kN/m^2$	25000	25000	25000	25000
$\nu'_{ur}$	—	0.2	0.2	0.2	0.2
$p_{ref}$	$kN/m^2$	100	100	100	100
$K_0^{nc}$	—	0.6093	0.6093	0.50	0.4850
$k_x$	$m/day$	0.7510E-3	0.7510E-3	0.7510E-3	0.7510E-3
$k_y$	$m/day$	0.7510E-3	0.7510E-3	0.7510E-3	0.7510E-3
$R_{inter}$	—	0.5	0.5	0.5	0.5
$K_0$ determ.	—	Automatic	Automatic	Automatic	Automatic
OCR	—	1.2	1.2	1.2	1.2

### 5.2.2 NGI-ADP

As mentioned in section 5.2.1, the NGI-ADP is only used for cohesive soils, and therefore only the clays are modelled using this soil model. The parameters for the NGI-ADP models are chosen, as for the HSS models, using empirical values to determine stiffness parameters, while site investigations have been used for strength and index parameters. Some of the relevant site investigations and laboratory results can be found in appendix A.

#### Unit weight

The unit weights for the cohesive soils are based on laboratory tests on field samples. All unit weights fall in the range of  $\gamma = 18 - 18.5 kN/m^3$ .

#### Shear stiffness to shear strength ratio

The value for the shear stiffness to shear strength ratio is based on empirical values. For situations where serviceability criteria and deformations are of high importance, these values should be determined through laboratory testing. However, seeing as stiffness factors have little effect on safety in the NGI-ADP model (Brinkgreve, 2019), empirical parameter values are deemed sufficient for use. A value of  $G_{ur}/S_u^A = 700$  is selected for the clays in both zones.

### Shear strain at failure

The shear strains at failure for active, direct and passive shear are also selected based on empirical values for the same reasons mentioned above. The values are selected at the lower end of the range suggested by Brinkgreve (2019), who recommends values of  $\gamma_f^E = 3 - 8\%$ ,  $\gamma_f^{DSS} = 2 - 8\%$  and  $\gamma_f^C = 0.5 - 4\%$ .

### Undrained shear strength

The active shear strength of the clays are determined through site investigations, using CPTU results in combination with triaxial testing to define a strength profile. Using the results, the strength profiles for zone 1 and 3 are defined as:

$$s_{u \text{ zone } 1}^A = \begin{cases} 43 \text{ kPa}, & z > -13.5 \text{ m} \\ 43 + 2.83(13.5 - z) \text{ kPa}, & z \leq -13.5 \text{ m} \end{cases} \quad (5.4)$$

and

$$s_{u \text{ zone } 3}^A = \begin{cases} 22 + 3.125(1 - z) \text{ kPa}, & -1 \text{ m} < z < -8.5 \text{ m} \\ 37 + 3.3(8.5 - z) \text{ kPa}, & z \leq -8.5 \text{ m} \end{cases} \quad (5.5)$$

Where  $z$  is depth below ground, defined negative downwards.

### Factors of anisotropy

The factors of anisotropy  $s_u^P/s_u^A$  and  $s_u^{DSS}/s_u^A$  are calculated using the plasticity indices obtained from laboratory testing. The calculations are based on correlations from NIFS (2014):

$$s_u^{DSS}/s_u^A = \begin{cases} 0.63, & I_p < 10\% \\ 0.63 + 0.00425 \cdot (I_p - 10), & I_p \geq 10\% \end{cases} \quad (5.6)$$

and

$$s_u^{DSS}/s_u^A = \begin{cases} 0.35, & I_p < 10\% \\ 0.35 + 0.00375 \cdot (I_p - 10), & I_p \geq 10\% \end{cases} \quad (5.7)$$

Where  $I_p$  is the plasticity index, entered as % in the formula.

### Initial mobilization

Initial mobilization can be calculated as (Brinkgreve, 2019):

$$\frac{\tau_o}{s_u^A} = \frac{0.5 \cdot (1 - K'_o) \cdot \sigma'_y}{s_u^A} \quad (5.8)$$

For normally consolidated clays, a value of  $\frac{\tau_o}{s_u^A} = 0.7$  is typically used. Seeing as the clays in the area are all considered normally consolidated, the parameter is kept at its standard value.

### Overview of NGI-ADP parameters

The following section gives an overview of most of the parameters utilized in the NGI-ADP models. The parameters can be found in table 5.3.

**Table 5.3:** Soil parameters for the granular soils in the HSS model

	Unit	Clay 1 Zone 1	Clay 1 Zone 3	Clay 2 Zone 3
Material model	—	NGI-ADP	NGI-ADP	NGI-ADP
Drainage type	—	Undrained (C)	Undrained (C)	Undrained (C)
$\gamma_{unsat}$	$kN/m^3$	18.5	18.0	18.0
$\gamma_{sat}$	$kN/m^3$	18.5	18.0	18.0
$G_{ur}/S_u^A$	$kN/m^2$	700	700	700
$\gamma_f^C$	%	1.0	1.0	1.0
$\gamma_f^E$	%	3.0	3.0	3.0
$\gamma_f^{DSS}$	%	2.0	2.0	2.0
$S_{u\ ref}^A$	$kN/m^2$	43	22	37
$\gamma_{ref}$	$m$	-13.5	-1.0	-8.5
$S_{u\ inc}^A$	$kN/m^2/m$	2.83	3.125	3.3
$S_u^P/S_u^A$	—	0.37	0.36	0.38
$\tau/S_u^A$	—	0.7	0.7	0.7
$S_u^{DSS}/S_u^A$	—	0.65	0.64	0.66
$\nu'(nu)$	—	0.495	0.495	0.495
$R_{inter}$	—	0.5	0.5	0.5
$K_0$ determ.	—	Automatic	Automatic	Automatic

### 5.2.3 Sheet piles

Both zones are to use AZ17-700 for the main sheet piles and AZ12-770 for the anchoring sheet piles. An overview of some of the input parameters can be seen in table 5.4 below.

**Table 5.4:** Input parameters for the sheet piles used in the excavation

Parameter	Unit	AZ 17-700	AZ 12-770
$EA$	$kN/m$	2.793E6	2.646E6
$EI$	$kNm^2/m$	76.08E3	38.09E3
$w$	$kN/m/m$	1.045	0.9904
$\nu(nu)$	—	0.25	0.25
$M_p$	$kNm/m$	585	405.7
$N_{p,1}$	$kN/m$	4497	4260

### 5.3 Modelling the zones in Plaxis 2D

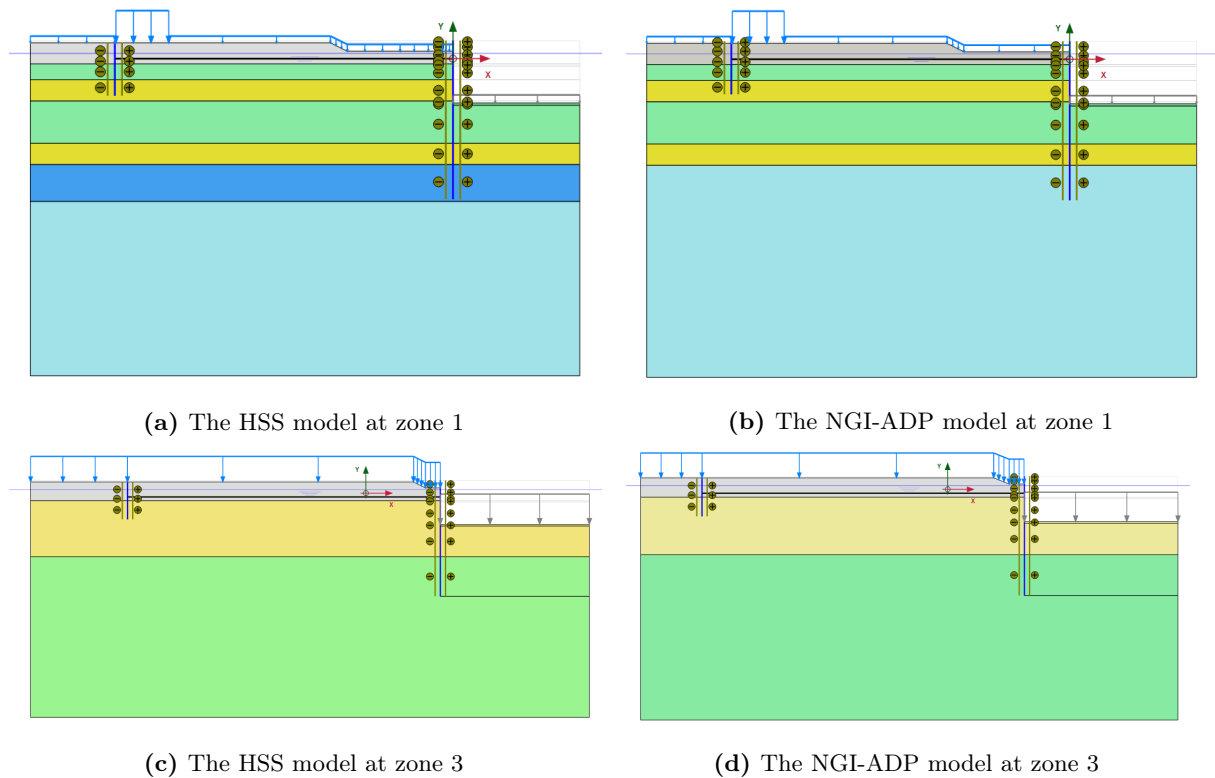
#### 5.3.1 Rock contour

As mentioned in Chapter 4, the depth to rock in the area can vary greatly. However, as the analyses for the excavation itself only focus on safety criteria, outlining the rock contour is not considered to be of any importance. This is because all failure planes will be significantly higher than any point where the rock would have influenced the result.

#### 5.3.2 Loads

The main loads around the excavation are considered to be snow and traffic loads, and the sheet pile wall at zone 1 is exposed to a distributed snow/traffic load of  $4.2 \text{ kN/m}^2$ . In order to avoid failure at the anchoring sheet pile wall, a stabilizing load of  $20 \text{ kN/m}^2$  is applied in the front of the wall.

The sheet pile wall at zone 3 is also exposed to a distributed snow/traffic load of  $4.2 \text{ kN/m}^2$ , but with no stabilizing load needed in front of the anchoring sheet pile wall.



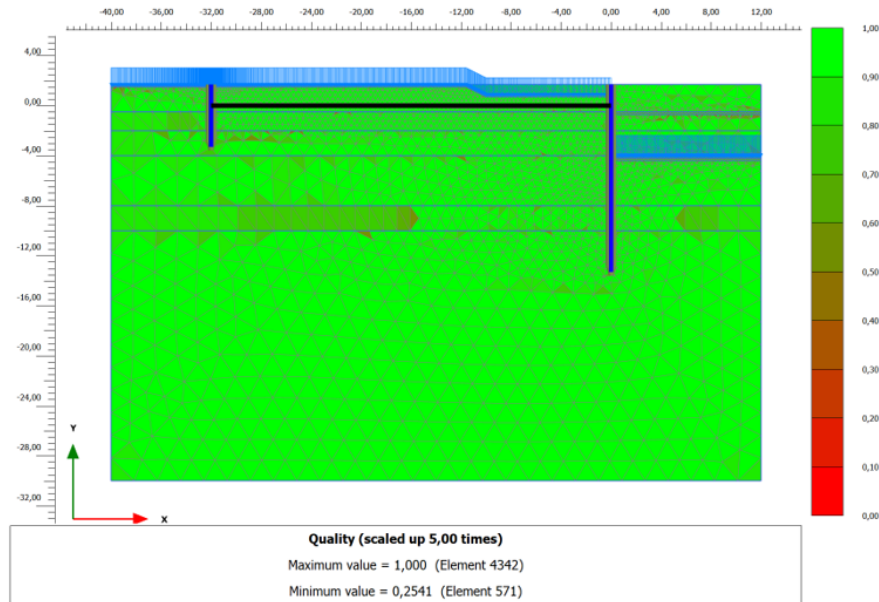
**Figure 5.1:** PLAXIS Models of zones 1 and 3 of the excavation

#### 5.3.3 Meshing

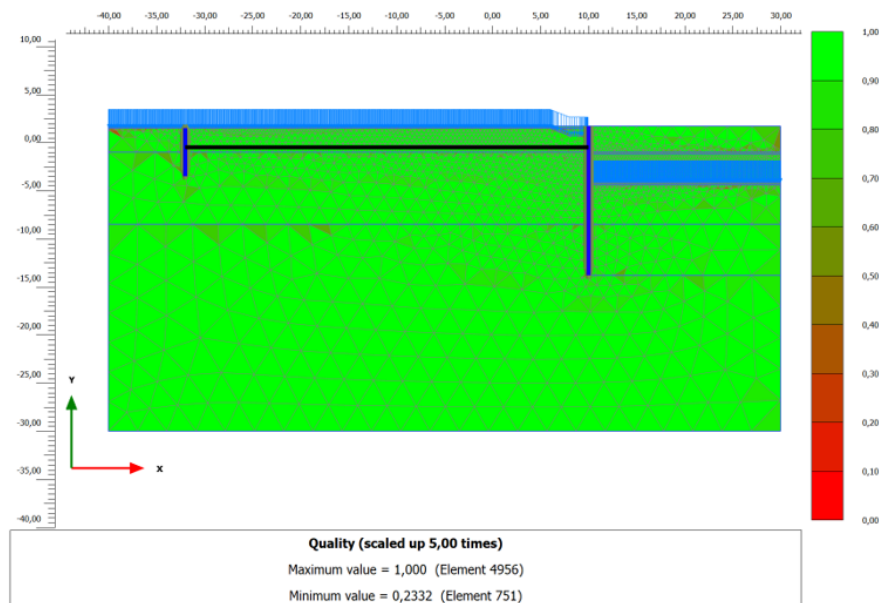
The mesh quality will affect the precision of the result, and a higher mesh quality will give better results. However, a higher mesh quality also means an increase in computational time. Due to the large amount of analyses that will have to be run during an optimization process, finding an appropriate trade off between quality and reduced computational time is key.



Because of more defined soil layers in zone 1, the meshing for this profile is more difficult than for zone 3. For this reason, zone 1 is meshed using a *very fine* mesh. Zone 3 is meshed using a *fine* mesh, as this is deemed sufficient even with a slightly coarser mesh than in zone 1. The overall quality of the meshes be seen in Figure 5.2a and 5.2b. Seeing as the NGI-ADP models will be the the ones in use, only the meshes for this soil model are presented.



(a) Mesh quality for the NGI-ADP model at zone 1



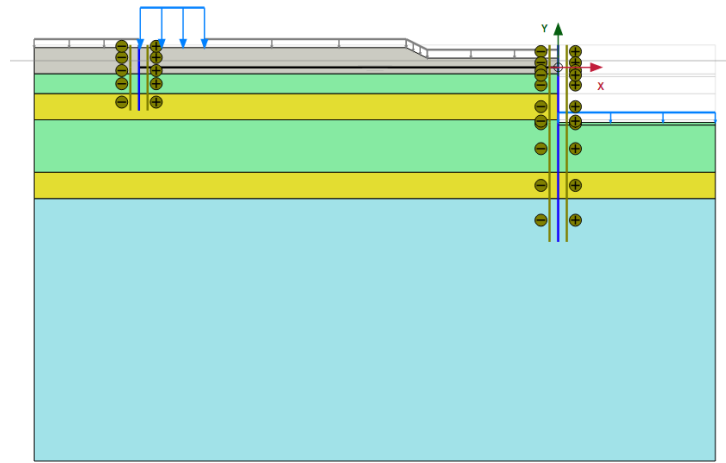
(b) Mesh quality for the NGI-ADP model zone 3

**Figure 5.2:** Mesh quality for the two zones

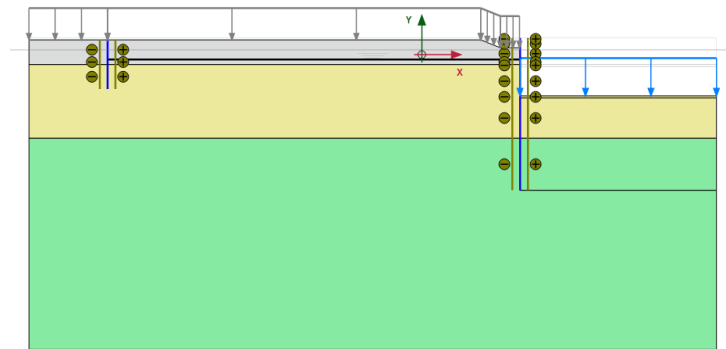
### 5.3.4 Ultimate limit state

The sheet piles have to keep the excavation pit dry for a 25-year flood, in which the water level rises from +0.5 meters to +1.7 meters above sea level (top of the sheet piles). This presents a large additional force on the support system in the form of considerable water pressure on the sheet pile walls. The additional load will present the most critical state, and consequently the ultimate limit state for the system.

For construction work in the excavation pit, a concrete slab is to be constructed at the excavation floor. This will provide a stabilizing load of approximately  $5 \text{ kN/m}^2$ , which is included in the models for the flood state. Additionally, snow loads are removed in the flood phase. This is because as such an event is expected to come during spring as a result of snow melting, or heavy rains during summer, and will in any case replace the present snow load. Seeing as a flood can be somewhat anticipated, vehicles will also be expected to be moved away from the excavations when the water level is high. For the flood phases in the two zones, see Figure 5.3. Even with the aforementioned stabilizing measures, the flood state still presents the situation closest to failure.



(a) Ultimate limit state for the NGI-ADP model at zone 1. Water level set at +1.7m



(b) Ultimate limit state for the NGI-ADP model at zone 3. Water level set at +1.7m

**Figure 5.3:** Ultimate limit states

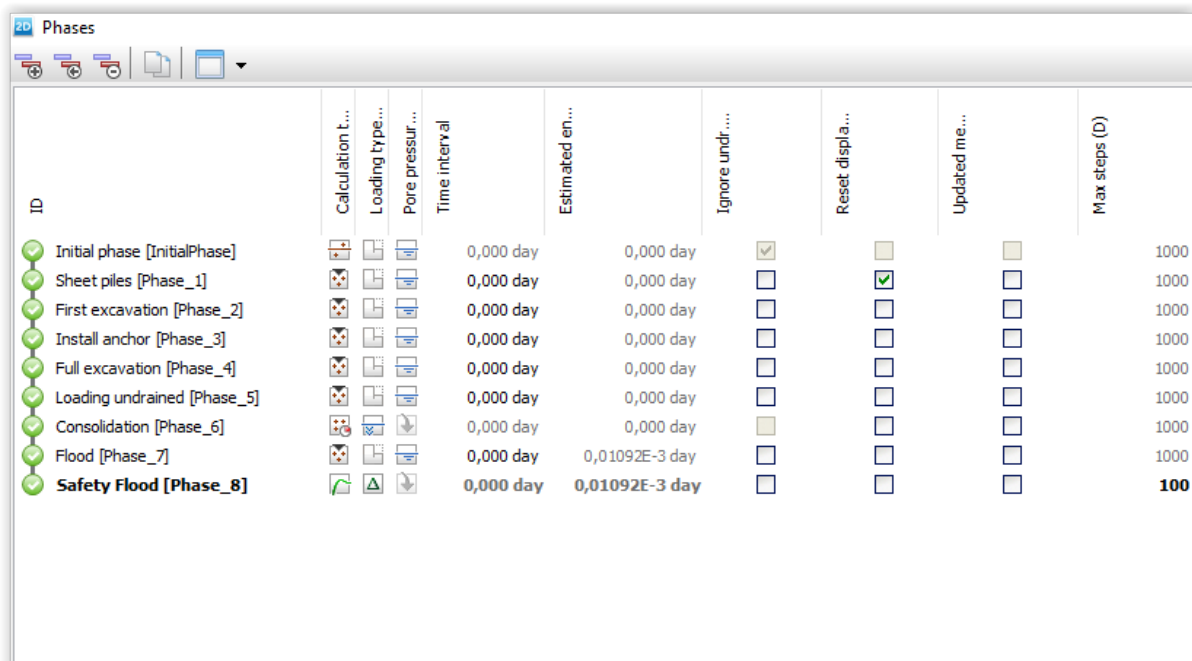
### 5.3.5 Phases

Both models consist of the same amount of construction phases. They can be summarized as:

1. Installment of sheet piles and anchoring sheet piles
2. Excavating to 0.5m below planned depth of anchor
3. Installment and prestressing of anchor
4. Excavating to full depth
5. Loading of sheet piles
6. Consolidation of excess pore pressure
7. Flood phase
8. Flood safety phase

Initial testing of the profiles was done using safety analyses for a flood on an undrained excavation along safety of the consolidated phase without flooding, but as mentioned in section 5.3.4, the flood phase on a drained excavation was found to be the most critical.

The phases as set up in PLAXIS 2D for the optimization processes can be seen in Figure 5.4.



ID	Calculation t...	Loading type...	Pore pressur...	Time interval	Estimated en...	Ignore undr....	Reset displa...	Updated me...	Max steps (D)
Initial phase [InitialPhase]				0,000 day	0,000 day	<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1000
Sheet piles [Phase_1]				0,000 day	0,000 day	<input type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	1000
First excavation [Phase_2]				0,000 day	0,000 day	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1000
Install anchor [Phase_3]				0,000 day	0,000 day	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1000
Full excavation [Phase_4]				0,000 day	0,000 day	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1000
Loading undrained [Phase_5]				0,000 day	0,000 day	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1000
Consolidation [Phase_6]				0,000 day	0,000 day	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1000
Flood [Phase_7]				0,000 day	0,01092E-3 day	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	1000
<b>Safety Flood [Phase_8]</b>				0,000 day	<b>0,01092E-3 day</b>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<b>100</b>

**Figure 5.4:** The phases used in the optimization process for both zones

## 5.4 Cost estimate of the anchored sheet pile system

To be able to analyze the cost-efficiency of the different designs, a framework for cost estimates must be established. The parameters being varied are the depth of the sheet piles and the length of the anchor. The depth of the smaller anchor sheet piles can also be varied, but as the length is more limited in range, this is not looked at. This will give a total cost of:

$$C_{tot} = C_{sheet\ pile} + C_{anchor} \quad (5.9)$$

According to representatives from Kynningsrud Fundamentering AS, the price of the sheet piles is more or less directly linked to the amount of steel used, at an approximate price of 15  $kr/kg$  steel. At zones 1 and 3, AZ17-770 sheet piles are expected to be sufficient for use. With a weight of 104.4  $kg/m^2$ , the approximated cost of the sheet piles can be expressed as:

$$C_{sheet\ pile} = 15 \frac{kr}{kg} \cdot 104.4 \frac{kg}{m^2} = 1566 kr/m^2 \quad (5.10)$$

Additional details, such as having to weld sheets together at a certain depth are not considered at this point.

The price of a 7 wire anchor is approximated at around 800  $kr/m$ . With a center distance of 4.2 meters, this gives a cost of:

$$C_{anchor} = 800 \frac{kr}{m} \cdot \frac{1}{4.2m} = 190.5 kr/m^2 \quad (5.11)$$

Finally, this gives the approximated cost function:

$$C_{tot} = 1566 \cdot l_{sp} + 190.5 \cdot l_a \quad (5.12)$$

Where  $C_{tot}$  is the total cost per meter in the plane and  $l_{sp}$  and  $l_a$  represent the total length of the sheet pile wall and the anchor, respectively. It should be noted that equation (5.12) is a very rough approximation of the change in design costs, and is only used to give an idea of what the costs may look like. There are also set, one-time costs that come with setting up for the installation itself. However, as these costs will be identical no matter the choice of parameter lengths, they are not included in the cost function.



# Chapter 6

## Analysis and Results

This chapter takes a look at the results from the optimization analyses. First, the formulation of the optimization is presented before the problems encountered with the SLSQP algorithm are briefly explained. The results from each zone are then presented separately, looking at initial mapping of the zone before presenting the optimized solutions.

### 6.1 Optimization formulation

The optimization for the project at Drammen Hospital is looking to minimize cost while maintaining a safety factor  $M_{sf} = 1.40$ . Due to the large amounts of open space around the excavation site, there are no limiting deformation criteria. Hence, the optimization at Drammen Hospital takes the form:

$$\text{SO} \begin{cases} \text{minimize cost function } C(x) \\ \text{subject to } \begin{cases} M_{sf} \geq 1.40 \\ \text{design constraints on } x \\ \text{equilibrium constraint} \end{cases} \end{cases}$$

### 6.2 Problems with the SLSQP algorithm

The optimization algorithms from **scipy.optimize**, as described in Chapter 3, did in theory have a good possibility of optimizing cost as a function of chosen design parameters and with certain constraints. However, after numerous testings, as well as dialogue with programmers with a greater knowledge of the **scipy.optimize** library in general, this was found to be a much more challenging task than expected. The main problem stems from the algorithms using the same input for the optimization and the constraints. This means that having the safety factor as an independent parameter in a constraint becomes impossible, which in turn makes the optimization process not work.

A possible workaround to this problem could have been to express the safety factor as a function of sheet pile length and anchor length. However, this thesis is looking to evaluate the viability

of Python in PLAXIS 2D for regular practice in geotechnical engineering, and such a solution would require too much for it to be realistic to implement. For instance, it would require a lot of information about how the safety factor develops with change in the design parameters, which if available, can just be used directly in the design process. For this reason, such a workaround was not considered to be of much interest, and the analyses were carried out using solely the optimization script and the automatized brute force method.

## 6.3 Zone 1

### 6.3.1 Initial testing

Zone 1 overall shows a few numerical instabilities, especially for solutions with shorter sheet piles or for short anchor lengths. As the solutions reach higher safety factors, however, the change in safety factor stabilizes and is fairly distinct.

Figure 6.1 and Figure 6.2 show the change in safety factor with sheet pile length and anchor length, respectively. For increasing sheet pile lengths, the safety factor displays an initial rapid increase, with an abrupt stop at a length of around 11 meters. From this point on, an increase in sheet pile length seems to have almost no effect on the global safety of the excavation. An explanation for this can be seen in Figure 6.3 further down. Numerical curiosities can be observed for the shorter anchor lengths, with the safety factor actually slightly decreasing with longer sheet piles.

An increase in anchor length appears to give a strictly linear increase in safety factor for the longer sheet piles, but the model again shows some numerical instabilities as the sheet piles become shorter.

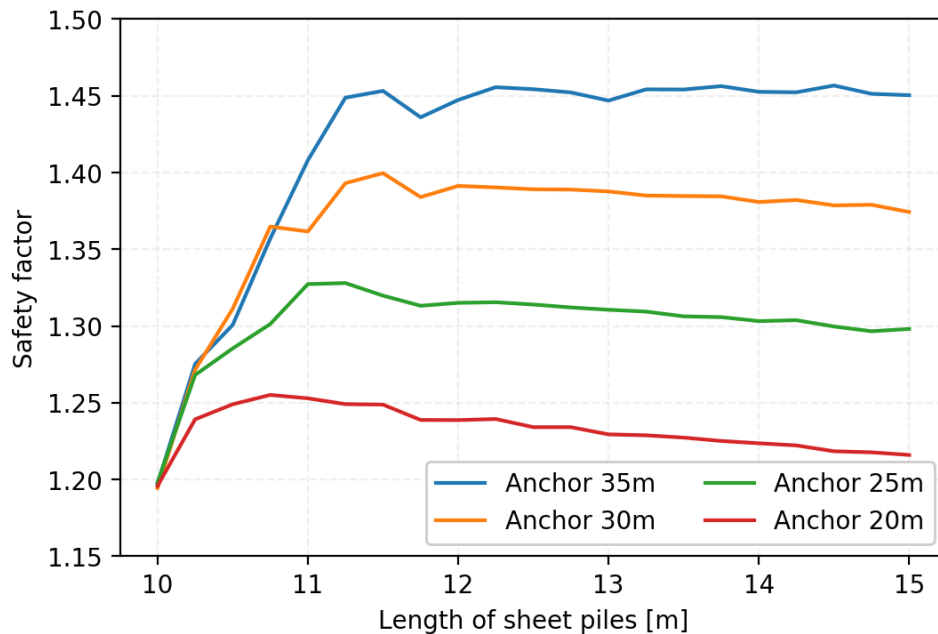
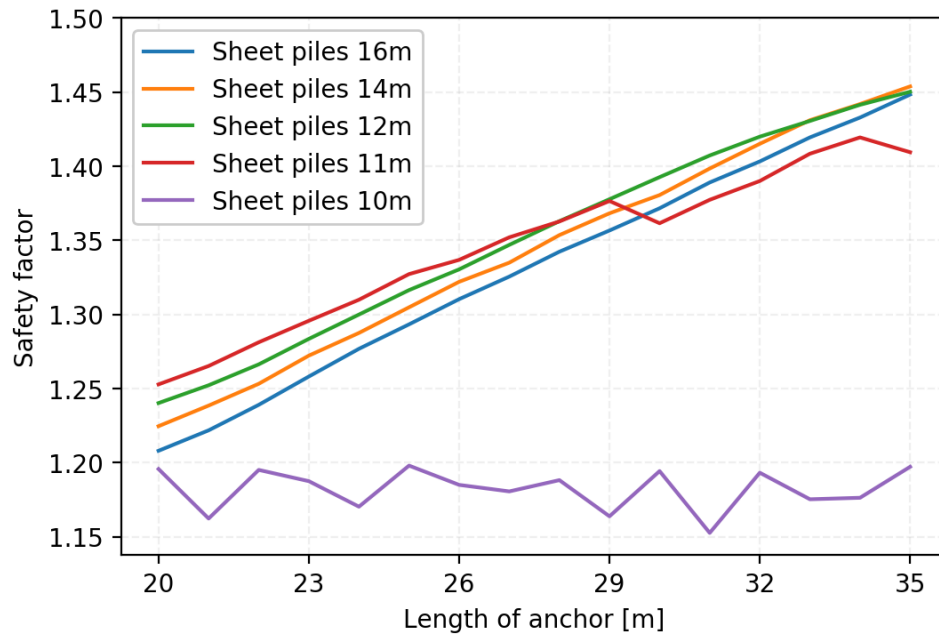
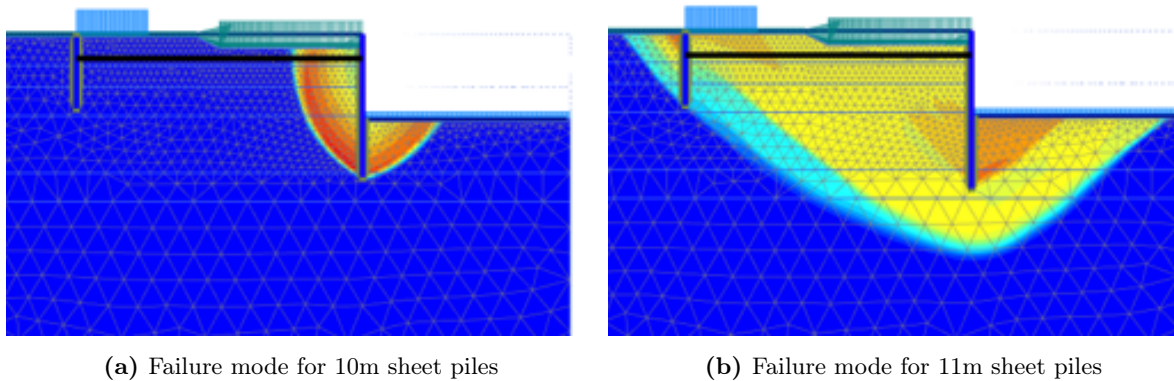


Figure 6.1: Change in safety factor with sheet pile length in zone 1



**Figure 6.2:** Change in safety factor with anchor length in zone 1

As is observed in both figures, the safety factor drops significantly from a sheet pile length of 11 meters to 10 meters. In fact, as seen in Figure 6.2, the safety factor for 10 meter sheet piles is completely unaffected by anchor length. When studying the case in more detail, this is found to be caused by a change in global failure mode as the sheet pile length reaches a critical length. A demonstration of the change in failure mode can be seen in Figure 6.3.



**Figure 6.3:** Change in failure mode for a 20m anchor as the sheet pile length decreases

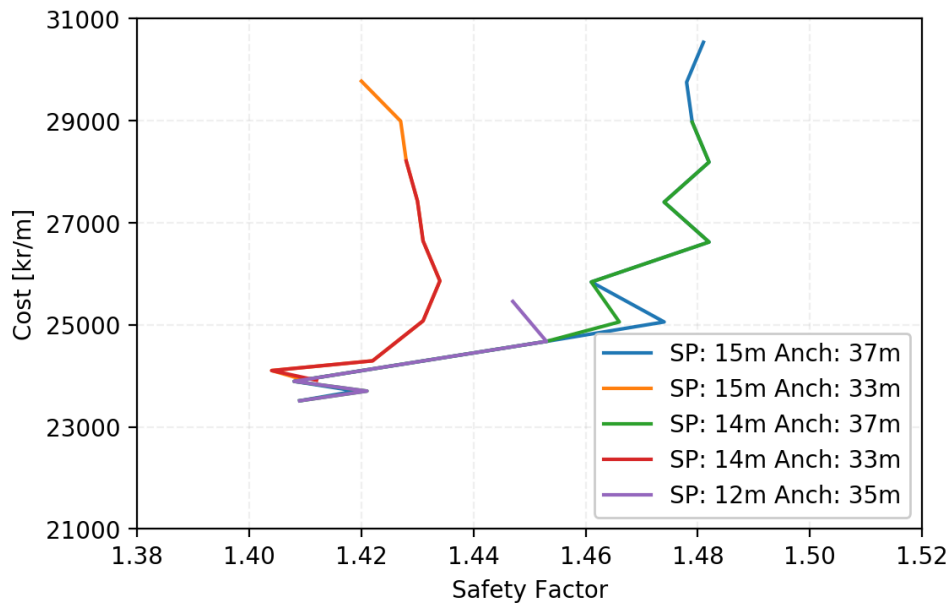


### 6.3.2 Optimization

Both optimization methods are run over the span of a night, in order to simulate a period in between workdays. A bound of  $M_{sf} \geq 1.395$  is set as the lowest constraint for both methods, as this value will satisfy the safety criteria when rounded to two decimal places.

#### Optimization script

The optimization script is run several times from different starting points. The paths taken from each starting point can be seen in Figure 6.4.



**Figure 6.4:** Path of the optimization for different starting points for zone 1

Neglecting the final steps, which overshoot the safety criteria, the optimization paths all end up at nearly identical solutions, with a maximum difference of 400 kroner per meter. As one could expect based on the initial testing, all the solutions end up with sheet piles of 11-11.5 meters, having found this to be the critical sheet pile length for zone 1. The final solutions for the different paths can be seen in table 6.1.

**Table 6.1:** Final solutions for different starting points in zone 1

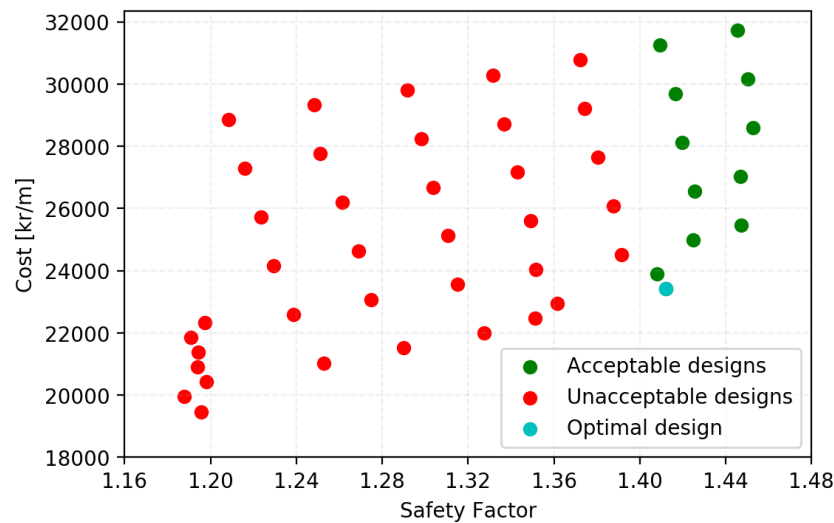
Starting position		Final solution			
Sheet piles [m]	Anchor [m]	Sheet piles [m]	Anchor [m]	$M_{sf}$	Cost [kr/m]
15	37	11	33	1.409	23513
15	33	11.5	31	1.410	23915
14	37	11	33	1.409	23513
14	33	11.5	31	1.412	23915
12	35	11	33	1.409	23513

### Automatized Brute Force

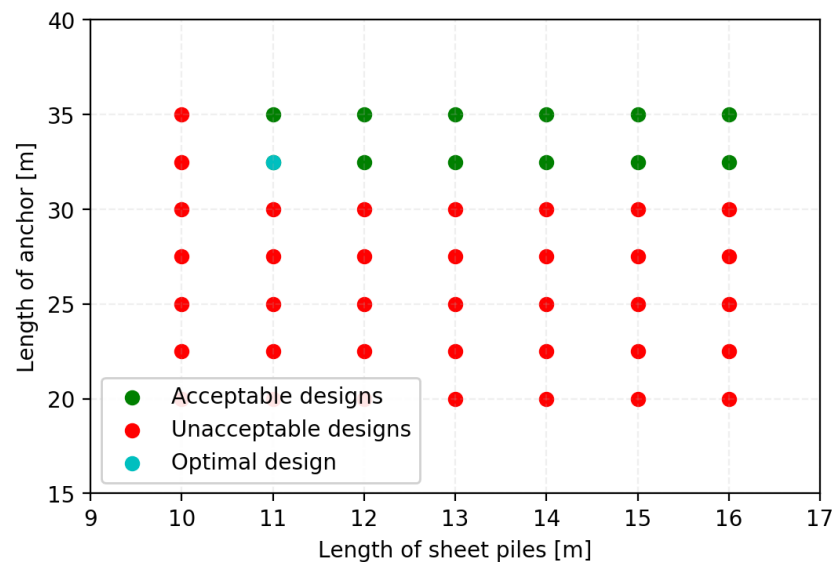
With an approximate run-time of 8-8.5 minutes per analysis and an assumed minimum time between workdays of 14 hours, a total of roughly 100 analyses could be run per night. However, as doing initial testing of safety factor behavior has given an unnatural amount of information about the relevant parameter ranges, it is chosen to run less analyses than the 100 available.

The automatized brute force optimization is run looking at solutions with anchor lengths from 20–35 meters at an increment of 2.5 meters per step and sheet pile lengths from 10–16 with an increment of 1 meter per step.

The checked solutions are plotted in Figure 6.5 and Figure 6.6. Note that the solutions previously found through the optimization script are not included in these plots.



**Figure 6.5:** Cost versus safety factor in the automatized brute force optimization for zone 1



**Figure 6.6:** Visualization of the lengths tested in the automatized brute force optimization for zone 1

Like the optimization script, the solution found by the automatized brute force optimization also has 11m sheet piles, but finds a slightly more cost-efficient solution due to a shorter anchor. The automatized brute force optimization is summarized in Table 6.2.

**Table 6.2:** Final solutions for the automatized brute force optimization in zone 1

Parameter range		Final solution			
Sheet piles [m]	Anchor [m]	Sheet piles [m]	Anchor [m]	$M_{sf}$	Cost [kr/m]
16–10	35–20	11	32.5	1.398	23418

As discussed in chapter 3 and confirmed by Figure 6.5, several solutions satisfy – and are close to – the required safety factor, but at quite different costs. For instance, the difference between the most expensive design and the optimal design is almost 8000 kroner per meter into the plane. For larger excavations with sheet piles stretching over 100-200 meters, this can mean a difference of 800,000-1,600,000 kroner. The differences will certainly not always be of this magnitude, but it is clear that there is potential of saving significant expenses through optimized design.

### Final solutions

The two optimization methods finish with an almost identical final solution for zone 1, with only 100 kroner per meter separating them. The final solutions are summarized in table 6.3.

**Table 6.3:** Final solutions for the two optimization methods in zone 1

Method	Sheet piles [m]	Anchor [m]	$M_{sf}$	Cost [kr/m]
Optimization script	11.0	33.0	1.409	23513
Automatized brute force	11.0	32.5	1.398	23418

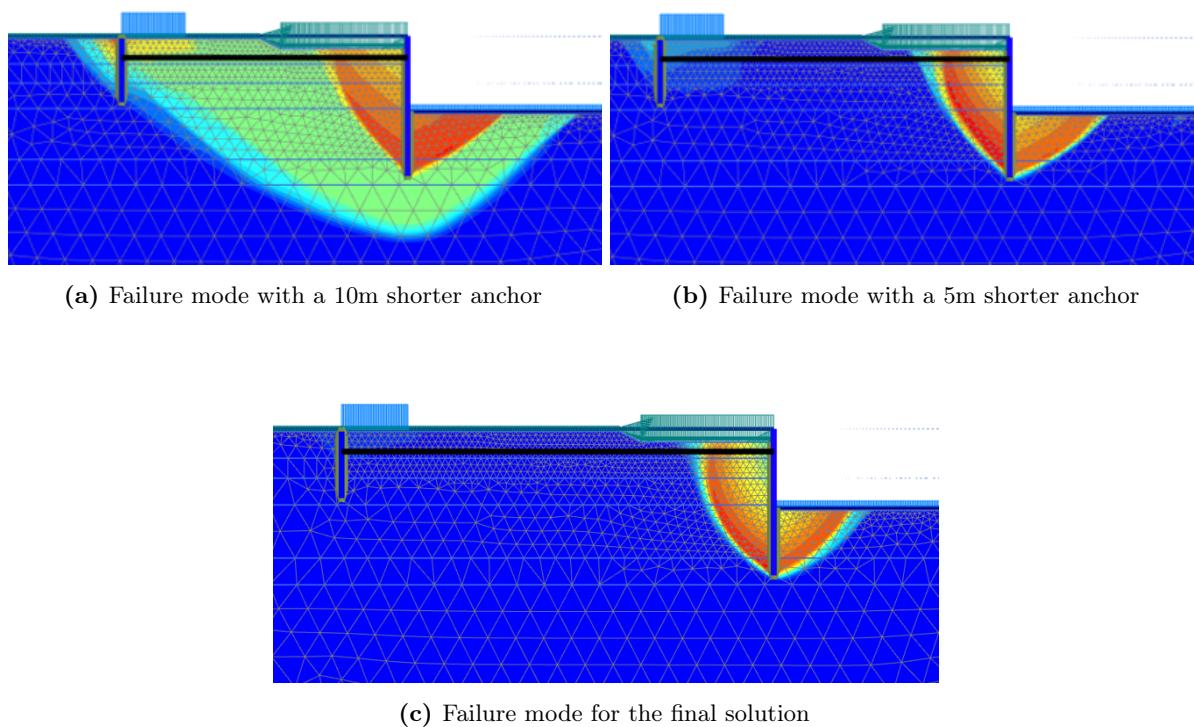
Having found the final solution, the profile is recalculated using the HSS model to extracted the forces and set them up against the resistance of the components. The forces in the different components are summarized in table 6.4.

**Table 6.4:** Final solutions for the two optimization methods in zone 1

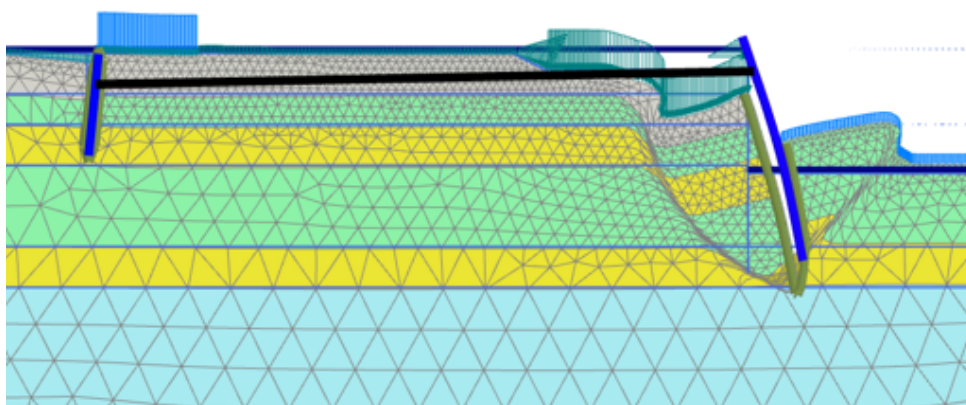
Component	$M_{max}$	$M_{Rd}$	$N_{max}$	$N_{Rd}$
Sheet piles	253.0 kNm/m	585 kNm/m	50.1 kN/m	4497 kN/m
Anchoring sheet piles	68.0 kNm/m	406 kNm/m	18.5 kN/m	4260 kN/m
Anchor	–	–	721.8 kN	1185 kN

Even with load factors, the components should have no problem dealing with the loads.

Comparing the failure mechanism for the final solution to solutions with a shorter anchor length, one can observe that for shorter anchors the system still gets notable resistance from the anchor sheet piles. This is not the case as the optimization reaches its final solution, see Figure 6.7. The sheet pile movement for the failure mechanism of the final solution is displayed in Figure 6.8.



**Figure 6.7:** Development of the failure mechanism as the anchor length increases



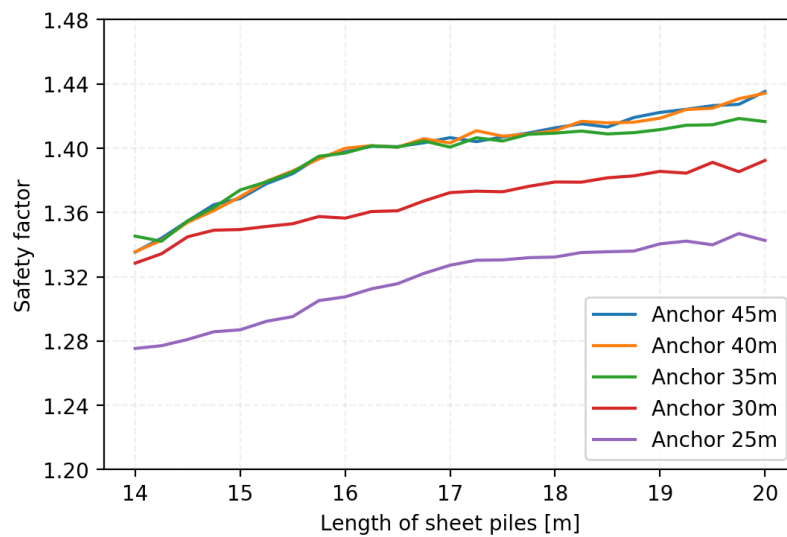
**Figure 6.8:** Sheet pile movement in the failure mechanism of the final solution in zone 1

## 6.4 Zone 3

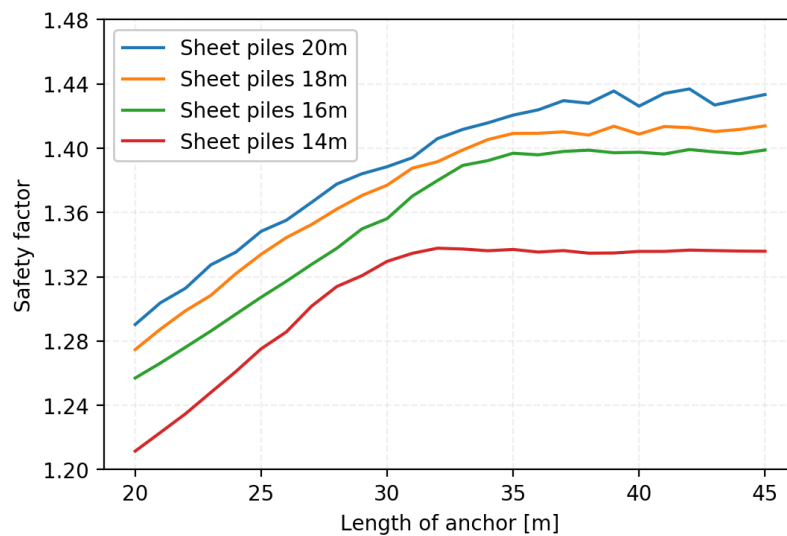
### 6.4.1 Initial testing

Zone 3 overall shows less numerical instability than zone 1, but the change in safety factor is quite different, see Figure 6.9 and Figure 6.10. Figure 6.9 shows that the change in safety factor with sheet piles length is fairly linear, but for designs with longer anchor lengths, there seems to be slightly less effect when sheet pile length goes below 16 meters. Increasing anchor length beyond 35 meters seems to have no added benefit.

Figure 6.10 shows a very distinct change in safety factor with anchor length, with an almost linear increase in safety factor with increasing anchor length, up until around 30-35 meters depending on sheet pile length. From this point on, an increase in anchor length has almost no effect on the safety factor. An explanation for this can be found in Figure 6.11 further down.

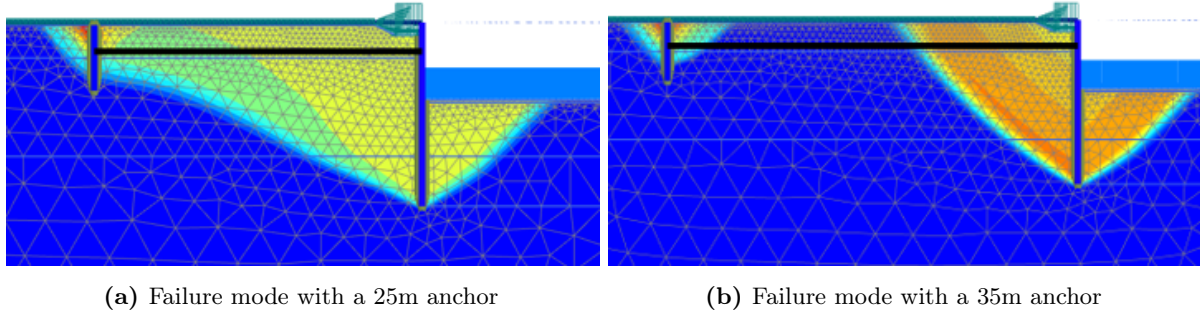


**Figure 6.9:** Change in safety factor with sheet pile length in zone 3



**Figure 6.10:** Change in safety factor with anchor length

As observed by the Figure 6.10, increasing the anchor length beyond 30-35m eventually stops having a positive impact on safety. Studying this in more detail, it is again observed that the failure mechanism changes as the anchor length reaches a critical length, see Figure 6.11. As the failure mechanism goes from connected to split, further increasing anchor has no impact on safety.



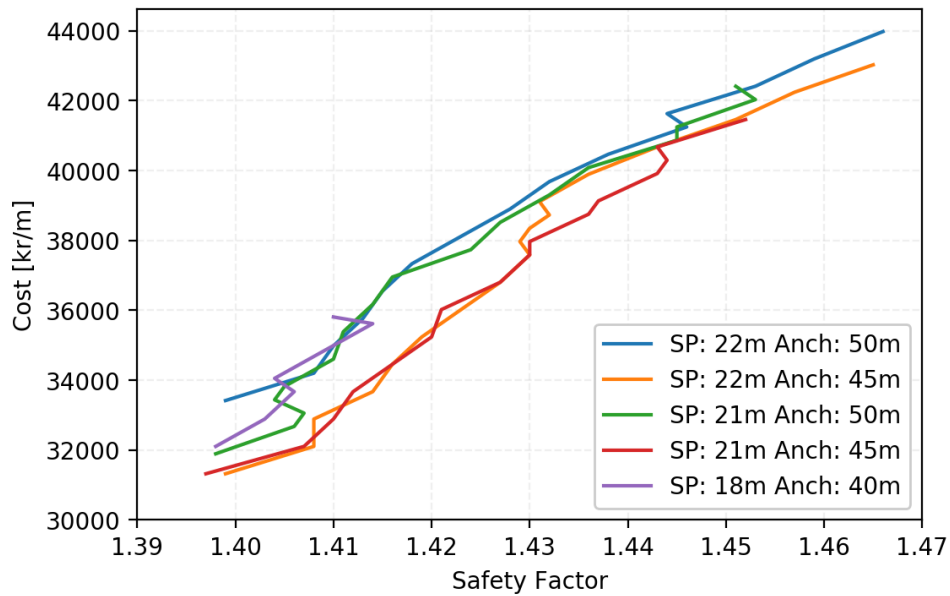
**Figure 6.11:** Development of the failure mechanism for 14m sheet piles as the anchor length increases

### 6.4.2 Optimization

Both optimization methods were again run over the span of a night, in order to simulate a period in between workdays, with a bound of  $M_{sf} \geq 1.395$  is set as the lowest constraint.

#### Optimization script

The optimization script is run several times from different starting points. The paths taken from each starting point can be seen in Figure 6.12.



**Figure 6.12:** Path of the optimization for different starting points for zone 3

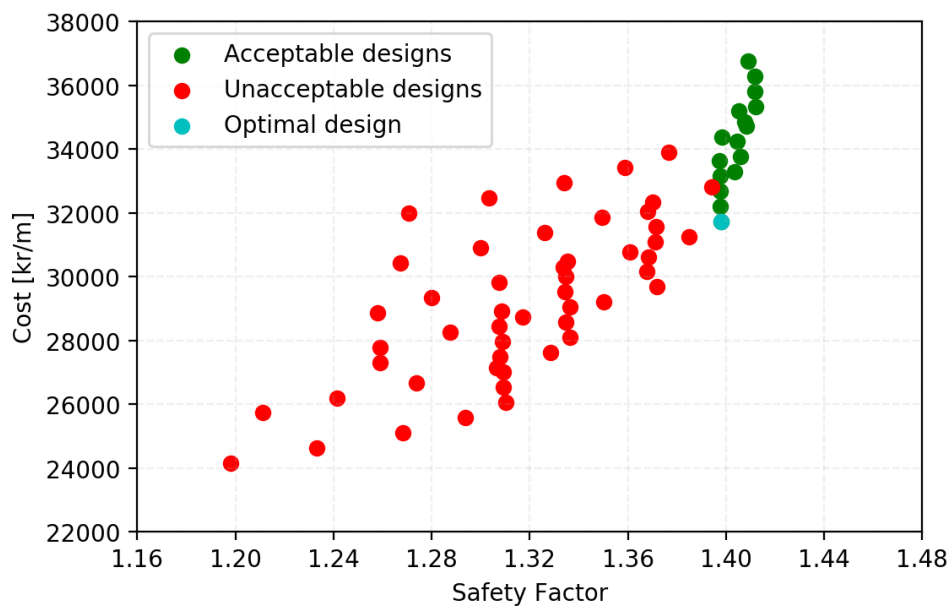
With the exception of one of the runs, the analysis yields fairly similar final solutions. As expected based on initial testing, the solutions have sheet piles of around 16, but somewhat surprisingly, there is a large variability in anchor length, indicating that the optimization script might struggle to find the correct optimization path if the changes in safety factor with each increment is small, as is the case for zone 3. The final solutions for the different starting points are summarized in Table 6.5.

**Table 6.5:** Final solutions for different starting points in zone 3

Starting position		Final solution			
Sheet piles [m]	Anchor [m]	Sheet piles [m]	Anchor [m]	$M_{sf}$	Cost [kr/m]
22	50	15.5	48	1.399	33417
22	45	15.5	37	1.399	31322
21	50	15.5	40	1.398	31893
21	45	15.5	37	1.397	31322
18	40	16	37	1.398	32105

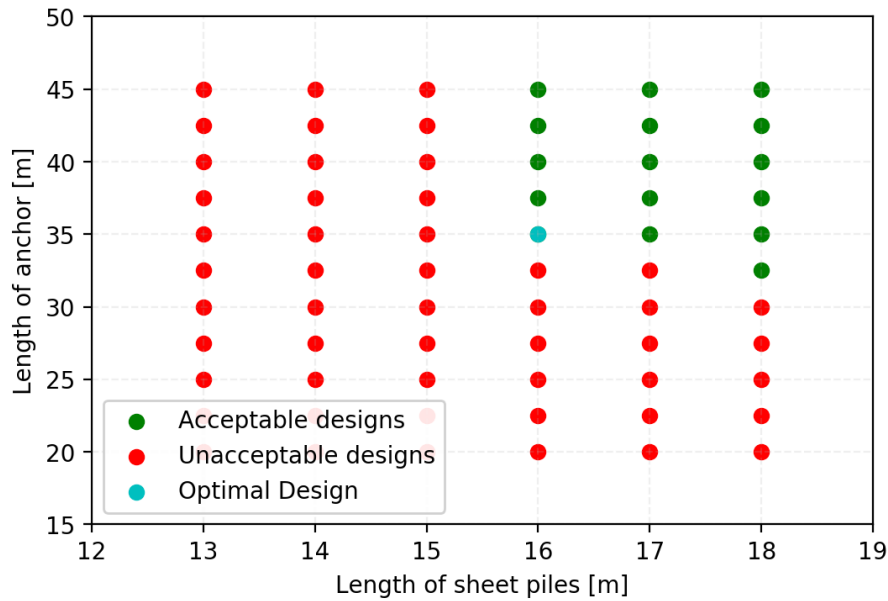
### Automatized Brute Force

The optimization done through automatized brute force is run looking at solutions with anchor lengths between 20 and 45 meters and sheet pile lengths between 13 and 18, with the same increments as in zone 1. The different solutions are plotted in Figure 6.13 and Figure 6.14. Note again that the solutions found by the optimization script are not included in the plots.



**Figure 6.13:** Cost versus safety factor in the automatized brute force optimization for zone 3





**Figure 6.14:** Visualization of the lengths tested in the automatized brute force optimization for zone 3

Like for zone 1, the automatized brute force method finds a solution close to the solutions found by the optimization script. But while the automatized brute force method found the most cost-efficient solution in zone 1, it is the optimization script that does this in zone 3. This is due to the optimization script finding a solution located in between the checked solutions in the automatized brute force optimization. The automatized brute force optimization is summarized in Table 6.6.

**Table 6.6:** Final solutions for the automatized brute force optimization in zone 3

Parameter range		Final solution			
Sheet piles [m]	Anchor [m]	Sheet piles [m]	Anchor [m]	$M_{sf}$	Cost [kr/m]
18–13	45–20	16	35	1.398	31724

It is again observed that several designs satisfy the required safety factor, showing the potential for cost optimization in zone 3 as well.

### Final solutions

The two optimization methods finish with an almost identical final solution for zone 1, with only 100 kroner per meter separating them. The final solutions are summarized in Table 6.7.

**Table 6.7:** Final solutions found by the two optimization methods

Method	Sheet piles [m]	Anchor [m]	$M_{sf}$	Cost [kr/m]
Optimization script	15.5	37.0	1.399	31322
Automatized brute force	16.0	35.0	1.398	31724

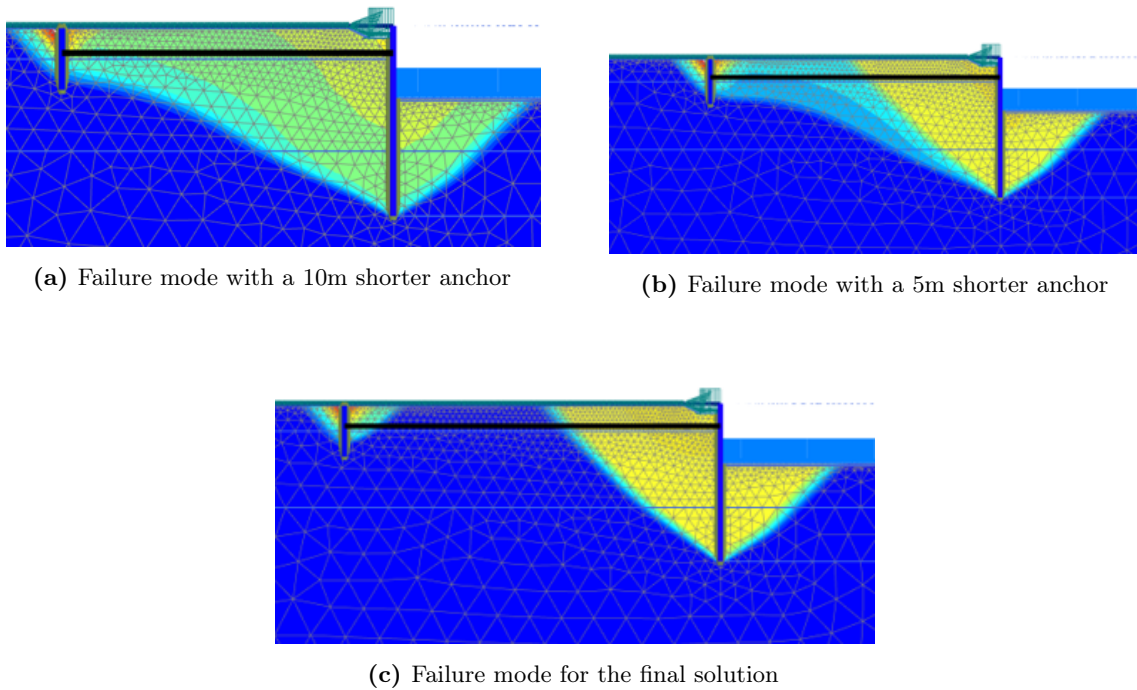
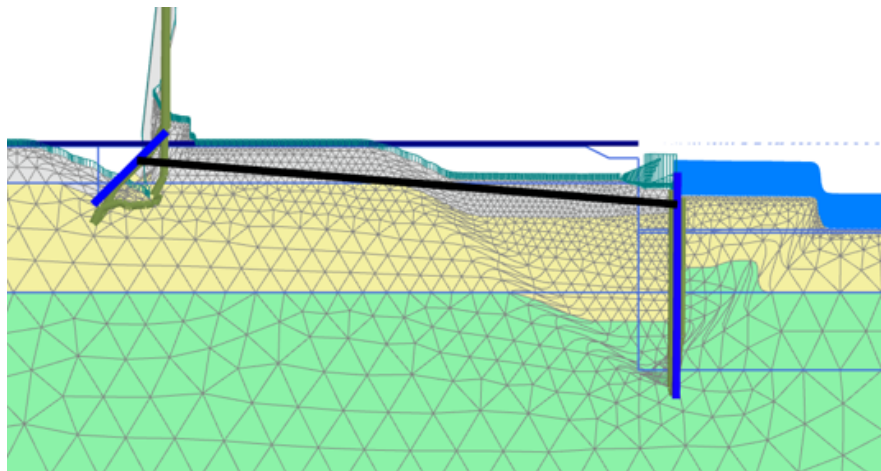
The profile is again recalculated using the HSS model to extract the maximum forces experienced for each component. The forces in the different components in zone 3 are summarized in Table 6.8. As for zone 1, the components should have no problem dealing with the loads in zone 3.



**Table 6.8:** Maximum forces in the different components in zone 3

Component	$M_{max}$	$M_{Rd}$	$N_{max}$	$N_{Rd}$
Sheet piles	192.5 kNm/m	585.0 kNm/m	60.0 kN/m	4497 kN/m
Anchor	–	–	664.1 kN	1185 kN
Anchoring sheet piles	79.4 kNm/m	406.0 kNm/m	9.5 kN/m	4260 kN/m

Looking at the development of the failure mechanism towards the final solution, one can again observe how increasing anchor length beyond a certain value is ineffective, see Figure 6.15. The sheet pile movement for the failure mechanism of the final solution is displayed in Figure 6.16.

**Figure 6.15:** Development of the failure mechanism as the system reaches the final solution**Figure 6.16:** Sheet pile movement in the failure mechanism of the final solution in zone 3



# Chapter 7

## Discussion

This chapter will be a more in-depth discussion on the results in Chapter 7 as well as the overall potential of modelling and analyses using Python in PLAXIS 2D. First, thoughts on the found solutions for the two zones are presented, and the observed advantages and disadvantages of the two different optimization methods are discussed. The general viability of scripted modelling in PLAXIS 2D is also examined. Finally, the prospects of analysis with Python in PLAXIS 2D are discussed.

### 7.1 Found solutions

The analyses of the two single anchor sheet piles at Drammen Hospital has resulted in two solutions that attempt to maximize safety per cost. Studying the development of the failure mechanisms for the system, it seems that solutions found in a transition between failure mechanisms give the most cost-efficient solutions. This could be because the system is able to get the most support out of each system component in these transitional areas.

The solution in zone 1 ends up with notably shorter sheet piles than zone 3. This could be explained by stronger materials in the upper layers in zone 1 or perhaps some numerical inaccuracies due to more difficult meshing. It should in any case be noted that the solution in zone 1 is of a more theoretical nature for the purpose of the thesis, as the sheet piles in most cases will be penetrated into the clay layer further down in order to prevent water flow into the excavation.

Interestingly, where in zone 1 it is the sheet piles that eventually stop giving added benefit, it is the anchors that do this in zone 3. However, it is also noted that the maximum anchor length is greater for zone 3 than zone 1 (45m to 35m) due to more available space. As it is around roughly 30–35 meters that increasing anchor length stops having an effect on safety in zone 3, it might be the case that the trend is not observed in zone 1 due to space constraints. Running another analysis to see the effect on safety from 30–45m anchors in zone 3 would also have been interesting.

While the originally planned coupling of SLSQP-algorithm to PLAXIS 2D proved to be tough to make work, the written optimization script has still allowed for a comparative analysis between an algorithm and automatized brute force. And the optimization methods, although simplistic in their theory, do serve their purpose. As long as the final solutions are checked properly, they can

be readily used in a geotechnical design process.

Certain numerical curiosities were seen through the initial mapping of the zones, but this seems somewhat reasonable as the models have complicated failure modes and the parameter increments are small. The curiosities were especially visible in zone 1 for unstable solutions, but overall the trends for the safety factors were fairly well defined. Clear change in safety factor development seems to be connected to a change in failure mechanism, and it appears that finding a solution at the border of the two most prominent failure mechanisms gives the most cost efficient solutions.

It was chosen to only look at anchor and sheet pile length, as these were perceived to have the greatest potential for cost optimization. It would also have been interesting to add a third parameter in the form of the length of the anchoring sheet piles, but this was unfortunately not possible to implement due to time constraints.

In hindsight, slightly more emphasis should have been put on using curve points during the optimization process. Due to the project analyzed not having maximum deformation criteria, this was not implemented, but it would have allowed for greater control of output. For instance, it would have made it possible to run scripts checking node deformations to maximum global deformations in order to spot local, unrealistic failure modes. Thankfully, it appears that, at least for the found solutions, this was not a major issue.

When examining the final solutions for the excavation, it is also important to discuss the actual utility of these types of optimizations. While the results from this thesis indicate that expenses can be saved through optimized design, this could vary greatly from project to project. Material costs are not always readily available and are also subject to change over time, meaning it could be difficult to know with certainty which solution are the most cost efficient.

With this in mind, it could be argued that looking at optimizing design with regards to deformation is the better choice in terms of general utility. Although this project operates without maximum deformation criteria, it is a very common design constraint, and unlike the construction costs, deformations are readily available through the PLAXIS output. This would also require more focus on the use of curve points in the form of nodes and stress points, which as mentioned is advantageous as it allows for more output control.

The solutions found in this thesis show there is potential in optimized design, whether looking at costs, deformations or other objectives. In a hectic working environment, an engineer could easily find a solution fairly close to a certain constraint and be satisfied, not knowing that a certain parameter is overly large and barely adds any benefit. Using optimized design can help this issue. That said, if no objective function in the form of costs or deformations is available, no "real optimization" can be made, as many solutions will fit the design constraints.

### **7.1.1 Optimization scripts versus Automatized Brute Force**

After having run multiple analyses and making modifications and tweaks, it becomes clear that the two optimization methods each have their strengths and weaknesses. This section will attempt to summarize some of the experiences made during the optimization processes.

The strongest feature of an optimization script is that it will always provide a solution very close to the optimum, especially if two or three runs can be completed from different starting points. This differs from the automatized brute force method, where the optimum could be located

somewhere in between the tested solutions. On the other hand, the optimization script suffers from a high dependence on correct choice of step sizes. An optimization algorithm far away from its constraints can operate with large steps, but when nearing the final solution, the step sizes have to decrease in order to avoid overshooting the optimal solution. Therefore, making these step sizes too coarse will likely lead to a suboptimal solution, but making them too fine will make the convergence of the optimization very slow. For instance, seeing as the safety factors in zone 1 and zone 3 have different sensitivities to parameter change, they should ideally have different step sizes, but this is a tough task to know intuitively.

Choice of step size is also relevant for optimization done using automatized brute force, but a big difference is that, knowing roughly the time needed for a single analysis, one may calculate the approximate amount of analyses that can be run over the span of a night. Thus, the increments for the relevant parameters can be adjusted to help find a more optimal solution.

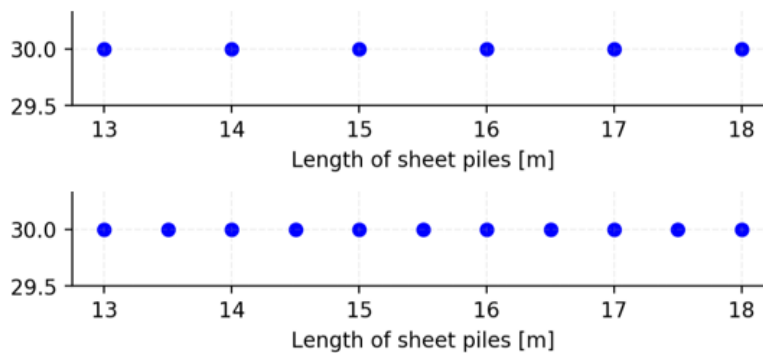
An advantage of using automatized brute force is that the model is mapped out to a greater extent than using an optimization script. This means that even if the most optimal solution is located in between the tested dimensions, the user will gain some knowledge of how the output changes with change in input.

A problem both methods face is the determination of starting points or parameter ranges. With limited knowledge of how a model behaves, it can be difficult to know where to start the optimization script from. Naturally, the starting point should be far enough away to not miss the optimum, but close enough that the optimization will converge sufficiently quickly. Similarly, optimizations done using automatized brute force have to select a relevant range for each parameter. While there is no convergence issue for this type of method, making the range too large might risk having the optimal solution between two points, but too small a range could miss the optimal solution entirely. Having good step sizes can help mitigate the convergence issue for the optimization script, whereas the automatized brute force optimization will have to rely on intuitive understanding of where the optimal solution will lie.

Both methods will also face problems when scaled to take in more parameters. The optimization script used in this thesis would arguably have the more difficult time implementing the new parameters, due to its structure, but if more parameters are wanted, this could definitely be fixed with a more elegant solution. That said, adding more parameters will inevitably lead to an increase in time needed per optimization, as each step in the process will need an extra PLAXIS analysis per added parameter. This is especially true if the convergence is slow. If wanting to do multiple runs from different starting points, as has been observed to be useful, this can quickly add up to making the optimization process overly time-consuming.

Similarly, the automatized brute force method will also suffer from more added parameters, but slightly differently than the optimization script. Where the optimization script suffers in the form of increased time per optimization, the brute force method will suffer from decreased accuracy due to larger gaps between each tested solution. This is because the amount of analyses available is independent of the amount of parameters that are changed. To illustrate, if 100 analyses are available and only two parameters are changed, one could for instance test 10 different lengths for both parameters. If three parameters are selected, however, this would mean testing only 4 or 5 different lengths per parameter, if they are each to be tested equally. As the parameter range in both cases will be equal, this will mean larger increments between each length, reducing the

accuracy of the solution. For an illustration, see Figure 7.1.



**Figure 7.1:** Equal parameter range but with different amounts of available analyses

Overall, the automatized brute force method is quite a bit easier to implement and set up properly, but has the disadvantage of potentially missing the optimal solution. With a better, more generalized optimization script that scales well with added parameters, this might seem more viable than automatized brute force. This is especially true as the amount of parameters included in the optimization increases.

## 7.2 Modelling with Python

As found in the prestudy and experienced again in this thesis, modelling with Python in PLAXIS 2D is definitely a case of both benefits and drawbacks. Certain aspects have some promise, but some negatives could potentially ruin the prospects.

One of the strongest attributes of scripted modelling is definitely when there are multiple, similar profiles that are to be modelled. In these cases, once the script for the first model is created, tweaking this seems easy and efficient, compared to using an original model created manually and using *move*-commands and changing project properties through the user interface. One could also argue scripting will have the advantage of creating separate, new models, and that these models are less exposed by numerical "noise" compared to models created manually, but based on a parent model.

However, the notion of new, clean models for each profile also introduces one of the biggest problems with scripted modelling: object names. Each object in a PLAXIS 2D model, such as lines and polygons, are assigned a name that has to be called in Python in order to make parameter changes. The problem is that these object names change if the geometry of the profile, such as number of soil layers, or the anchor length or sheet pile length, changes. This is particularly problematic when creating phases. Because the object names are used directly when creating the phases of the excavation, this means a separate script for phase creation must be made for each profile. In that case, scripted modelling will be just as tedious as doing it manually.

Creating phases through Python can also be a little tedious in itself, as each boundary condition and soil polygon must be called on specifically in order to make changes. Unfortunately, there seems to be no way of changing a group of polygons within the same command, as can be done

manually in the user interface. For instance, changing the water level for a single soil polygon is done using:

```
g_i.gotoflow()
g_i.WaterConditions_1_2.Conditions[g_i.Phase_2] = "Custom level"
g_i.WaterConditions_1_2.Level[g_i.Phase_2] = g_i.UserWaterLevel_1
```

If there are several soil layers or excavation layers, this will automatically produce a lot of soil polygons. Combined with needing to change water conditions for several phases, it will mean a lot of lines needed in the script.

Another big part of the modelling process in PLAXIS is creating materials, and doing this through Python has generally also proven a somewhat challenging experience. Most soil materials are created without problems, but some have had to be created using workarounds or tweaking after its original creation. There have especially been difficulties setting stiffness parameters and parameters that are affected by others, as these have to be in agreement in order to avoid error messages. Plate materials are particularly challenging on this part, where if a  $G_{ref}$  value is not specified (normally created automatically based on initial stiffness parameters), PLAXIS will display an error message.

While having the materials created through Python can be practical in some cases, such as slightly quicker tweaking of parameters, most actions and modifications concerning the materials are easier done manually in the user interface. Available options in PLAXIS, such as the ability to copy materials, already serve as sufficient tools if needing to create several similar soils. Add the use of project specific material libraries, and a user practically has all the tools needed for efficient material creation.

In general, modelling in Python could offer some practical advantages, but whether this is worth it is a case that must be evaluated from one project to another. It could be possible to work up a sort of modelling library for certain standard cases, such as a single anchor sheet pile excavation or soldier pile excavations, which would help in making scripted Python more useful. However, this still runs into the problem with line numbers and polygon numbers changing with different geometries, making a completely generalized modelling library difficult to create.

Finally, it is worth noting is that all the arguments in this section, while fairly unenthusiastic, are made with focus on practicality in geotechnical engineering. For more research based PLAXIS modelling, using Python can be a highly useful tool. This is demonstrated in a master thesis by Grecu (2018), looking at suction buckets for offshore wind turbines. In the thesis, 100 suction bucket models of differing dimensions were created using scripted modelling, saving enormous amounts of time and effort. Without an automatized modelling process, a thesis like this would have been much less feasible.

### 7.3 Analysis with Python

Analysis using Python in Plaxis has overall proven to have a lot of potential. Being able to run optimization scripts or automatized modelling overnight opens up previously unexplored territory



when it comes to efficient analysis. At the same time, there are a few negative aspects that must be accounted for.

Firstly, whether using it for running optimization scripts, doing automatized brute force optimization or simply running different profiles overnight, there is no doubt that analysis using Python scripting in PLAXIS has great prospects for geotechnical engineering in the years to come. Although running multiple calculations through batch files already is an available tool in PLAXIS, the ability to do modifications to a model within a batch analysis is an enormous change. This thesis has only tested a small part of what is accessible in the realm of automatized analysis, but the possibilities seem endless.

That said, analysis done through Python scripting also has some highly relevant issues that must be acknowledged. One of the most blatant differences to manual analysis is the lack of output control, and this is definitely an issue if not taken into much consideration. An important part of using FEM-software is the engineer's ability to control an obtained solution, and this is not possible in the same way when multiple analyses are run directly after one another. An optimization on an incorrect model will still give incorrect results. For this reason, making sure a certain model is working properly before starting any form of optimization is critical.

For instance, small, local, unrealistic failure mechanisms that are quite common in PLAXIS 2D are more difficult to pick up when the output is not available for control after every analysis. This issue could be mitigated through certain measures, such as scripts comparing deformations in a critical point to the overall biggest deformations. It could also be possible to write scripts that take a screenshot of the incremental displacements/strains for every step, at least allowing for control of the failure modes. But even with these type of measures, controlling soil pressures, pore pressures and other relevant values in each step is still difficult. A solution to this could be to save each step of the optimization process as a separate PLAXIS file, but this would quickly start requiring a lot of disk space. In addition, if every step has to be checked manually, many of the advantages in having an automatized analysis in the first place would be lost.

Another smaller inconvenience with analysis through scripting is that not all values are available for extraction. For instance, extracting groundwater flow in an excavation is challenging, as this requires the user to define a section through which the flow should be calculated. On the other hand, most designs will solve this issue either by piling down to a suspected clay layer if available, or using pumps after excavating to get rid of the water. In both cases, the sheet piles will be designed to meet safety or deformation criteria, not minimizing groundwater flow.

A final drawback with automatized analysis that a user must be conscious of is with an analysis not converging, as this will completely halt a run. This can happen with both overloaded loading phases and safety phases for complicated calculations. In order to avoid this issue, it is recommended to always start calculating the more stable solutions, before reducing lengths or in other ways lowering the safety factor. That way, even if a calculation should diverge, at least some results are obtained, and a point where the model starts having numerical problems is revealed.



## Chapter 8

# Conclusion and further work

The aim of this thesis has been to examine the possibilities of optimizing design of single anchor sheet piles using automatized modelling and analysis in PLAXIS 2D. The thesis has also attempted to give a general assessment of the potential in using Python in PLAXIS 2D for geotechnical engineering in practice. As a part of the thesis, several previously tested optimization methods have been presented and discussed, looking at advantages and drawbacks. Some general optimization theory has been presented, as well as the choice of optimization methods to be tested in the thesis. The choice of input parameters in the models, along with the model itself and the economic framework of the sheet piles have been explained to give good understanding of the situation at Drammen Hospital. The profiles have been analyzed and attempted optimized, with a focus on the viability of Python in PLAXIS 2D for geotechnical engineering in practice.

### 8.1 Conclusion

For anchored sheet piles, finding the solution at the transition of the two most prominent failure mechanisms seems to give the most cost efficient solutions. The results from the analyses indicate that the sheet piles must be of a certain length in order to fully utilize the extra support of the anchoring sheet piles, but after reaching a satisfactory length, further increase in sheet pile length has little to no added benefit on safety. This was especially prominent in zone 1, but similar trends were also found in zone 3.

For each zone, the two optimization methods found nearly identical optimal solutions, only separated by a few hundred kroner per meter. Overall, it is clear that for few parameters, not much separates the methods in terms of the final result. This speaks in favor of the automatized brute force method, as it gives good information on the profile and is easier to set up.

Although an automatized brute force optimization is easier to set up properly, a top down optimization script seems to have greater prospects for optimization as the number of parameters to be optimized increases. That said, an advantage of automatized brute force is that the model is mapped, giving the user a good image of how different outputs change with change in input. In both cases, choosing step sizes or parameter ranges can still be difficult. Both methods have difficulties with an increase in parameters analyzed, but in different ways. Where the optimization script will suffer from an increase in computational time, the automatized brute force suffers from decreasing accuracy. Ultimately, which type of method is used must be chosen depending on the

project and the needs of the engineer.

When it comes to modelling using Python in PLAXIS 2D, this is found to have some blatant drawbacks. Especially for material creation, the few small advantages it could give a user fade completely to the tedious creation process and numerous error messages. This perception is reinforced by the fact that the manual tools already available in PLAXIS 2D allow for very efficient material creation.

Creating the actual geometry of a model could however still have some potential. If a modelling library can be set up, it could offer some good efficiency in terms of modelling, but these cases would probably have to be very generalized. It would also still need some manual action from the user to set groundwater and boundary conditions properly. Overall, when it comes to modelling, it seems using Python has too many drawbacks to make it efficient for geotechnical engineering in practice.

Analysis using Python in PLAXIS 2D has been found to have serious potential. Being able to run detailed analyses overnight adds a new dimension to efficient analysis and seems to have great future prospects. Although some values are unavailable, such as groundwater flow, a user still has access to safety factors, deformations and different components forces, allowing for automatized optimization as well as automatized deformation and force extraction. At the same time, automatized analysis also has some drawbacks that must be handled, most notably the lack of output control.

The utility of optimizing with regards to cost will also scale with the size of the excavation. For large excavations, spending 20-30 hours extra optimizing design will most likely be a worthwhile economic investment, whereas for smaller projects, the additional hours spent might not make the overall costs lower for the contractor. For this reason, the potential benefits of optimization should always be weighted to the additional hours spent in design.

## 8.2 Further work

As mentioned in the introduction, this thesis has only taken a look at single anchor sheet piles in order to get a proper analysis of the system. Further work within the topic would be looking at other types of excavations and examining the potential for optimization there. For instance, excavations such as tieback anchor excavations would be interesting to look at, where the system could be optimized not only with regards to length or angle, but also the amount of anchors.

Further work could also include developing more robust optimization scripts that scale well with increasing amounts of parameters. This would properly show the potential of these types of optimizations for geotechnical engineering in practice. Within this improvement there should be focus on output control in the form of for instance scripts that check deformations in a curve point to maximum observed deformation. Another interesting option could be to integrate scripts that take a screenshot of incremental displacements for each step, in order to give an idea of failure mode development.

It could also be considered to look further into already existing optimization algorithms in more detail. While the coupling with the algorithms tested in this thesis proved difficult to make work, there could perhaps exist others that make the originally planned coupling possible.

# Bibliography

- Baecher, G. B. and Christian, J. T. (2003), *Reliability and statistics in geotechnical engineering*, Wiley, Chichester.
- Bendsøe, M. P. and Kikuchi, N. (1988), ‘Generating optimal topologies in structural design using a homogenization method’, *Computer Methods in Applied Mechanics and Engineering* **71**(2), 197–224.
- Brinkgreve, R. (2019), ‘Plaxis 2d connect edition v20 – material models manual’, *Plaxis* .
- Christensen, P. W., Klarbring, A. and Gladwell, G. M. L. (2008), *An Introduction to Structural Optimization*, Vol. 153 of *Solid Mechanics and Its Applications*, Dordrecht: Springer Netherlands, Dordrecht.
- Ding-Zhu Du, Panos M. Pardalos, W. W. (2009), *History of Optimization*, Springer, Boston, MA.
- Doltsinis, I. and Kang, Z. (2004), ‘Robust design of structures using optimization methods’, *Computer Methods in Applied Mechanics and Engineering* **193**(23-26), 2221–2237.
- Gong, W., Huang, H., Juang, C. H. and Wang, L. (2017), ‘Simplified-robust geotechnical design of soldier pile-anchor tieback shoring system for deep excavation’, *Marine Georesources and Geotechnology* **35**(2), 157–169.
- Gong, W., Wang, L., Juang, C. H., Zhang, J. and Huang, H. (2014), ‘Robust geotechnical design of shield-driven tunnels’, *Computers and Geotechnics* **56**, 191–201.
- Grecu, S. (2018), ‘Behaviour of axially loaded bucket foundations in sand’, *Aalborg University* .
- Hult, A. (2019), ‘Datarapport supplerende grunnundersøkelser’, *Norconsult* .
- Juang, C. H., Wang, L., Hsieh, H.-S. and Atamturktur, S. (2014), ‘Robust geotechnical design of braced excavations in clays’, *Structural Safety* **49**, 37.
- Lindgård, J. (2019), ‘Parameter optimization of sheet piles at drammen hospital - prestudy’, *NTNU* .
- M.J.D., P. (1994), ‘A direct search optimization method that models the objective and constraint functions by linear interpolation’, *Advances in Optimization and Numerical Analysis. Mathematics and Its Applications*. **vol 275**.
- NIFS (2014), *En omforent anbefaling for bruk av anisotropifaktorer i prosjektering i norske leirer*, Norges vassdrags- og energidirektorat.
- Nordal, S. (2019), *Geotechnical Engineering Advanced Course*, Norwegian University of Science and Technology.

- Paiva, R. M., Crawford, C. and Suleman, A. (2014), ‘Robust and reliability-based design optimization framework for wing design’, *AIAA Journal* **52**(4), 711–724.
- Prästings, A. (2019), *Managing uncertainties in geotechnical parameters: From the perspective of Eurocode 7*, KTH Royal Institute of Technology.
- Pucker, T. and Grabe, J. (2011), ‘Structural optimization in geotechnical engineering: basics and application’, *Acta Geotechnica* **6**(1), 41–49.
- Schittkowski, K. (1980), *Nonlinear programming codes : information, tests, performance*, Vol. 183 of *Lecture notes in economics and mathematical systems*, Springer, Berlin.
- StatensVegvesen (2018), ‘Geoteknikk i vegbygging’, *Vegdirektoratet* .
- Taguchi, G. (1986), *Introduction to quality engineering : designing quality into products and processes*, Sekkeisha no tame no hinshitsu kanri, Asian Productivity Organization, Tokyo.
- VestreViken (2019), *Nytt Sykehus i Drammen*, hentet fra: <https://vestreviken.no/om-oss/nytt-sykehus-i-drammen>.

## Appendix A

### Site investigations

This appendix contains the site investigations and laboratory results received from Norconsult that were used for determination of soil layers and input parameters in Chapter 5.

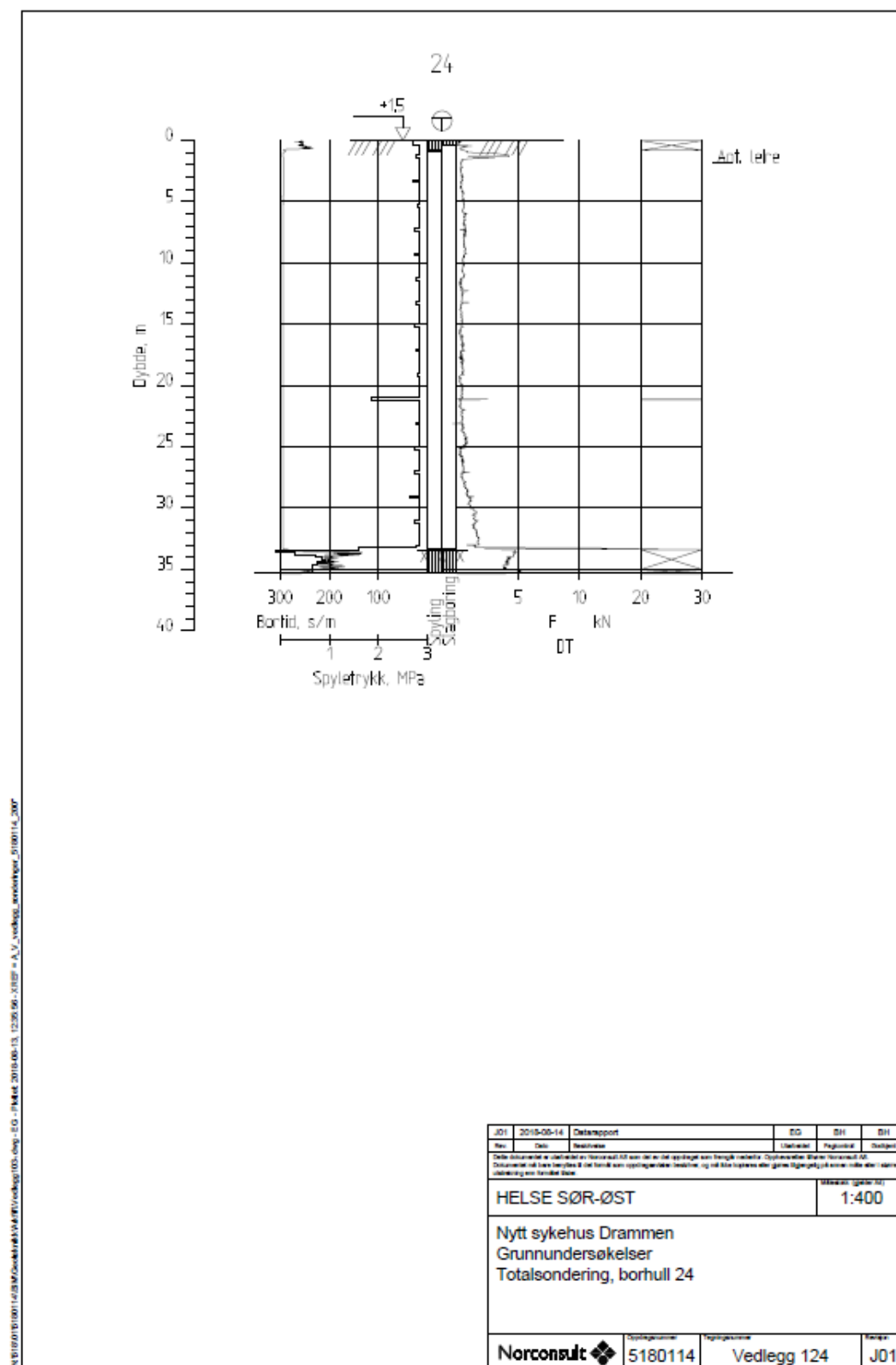


Figure A.1: Total sounding from borehole 24 in zone 3

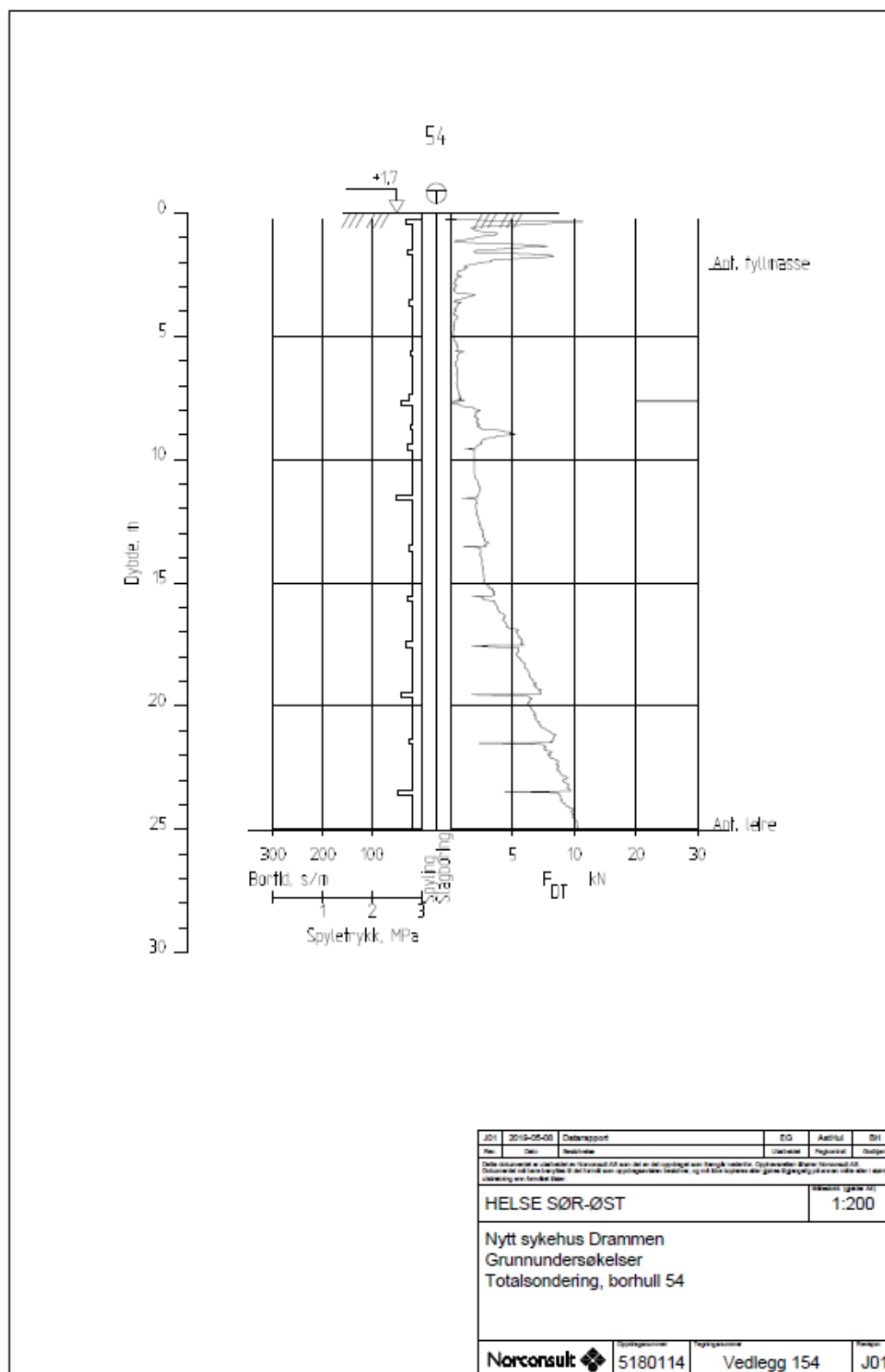


Figure A.2: Total sounding from borehole 54 in zone 3

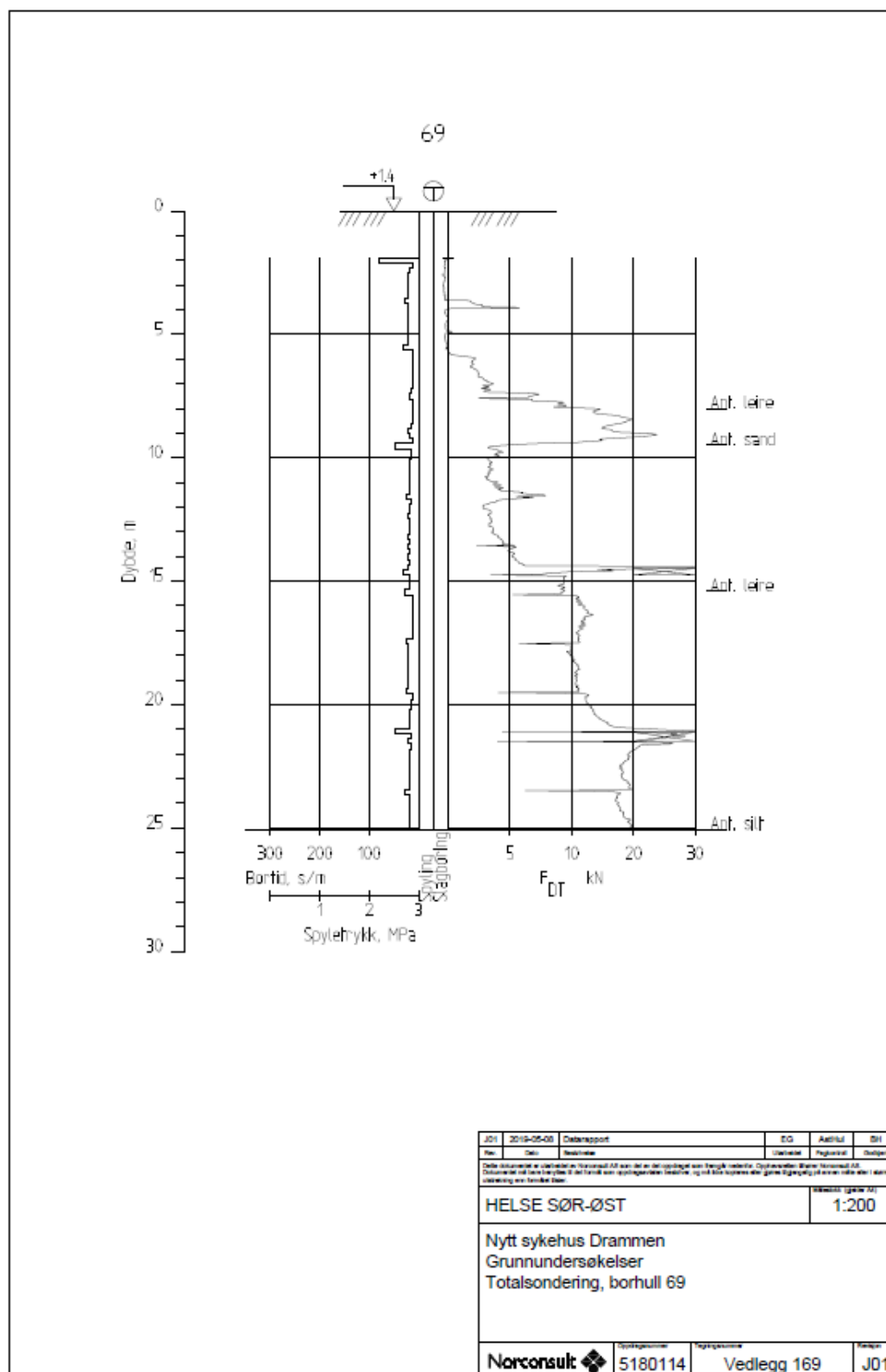
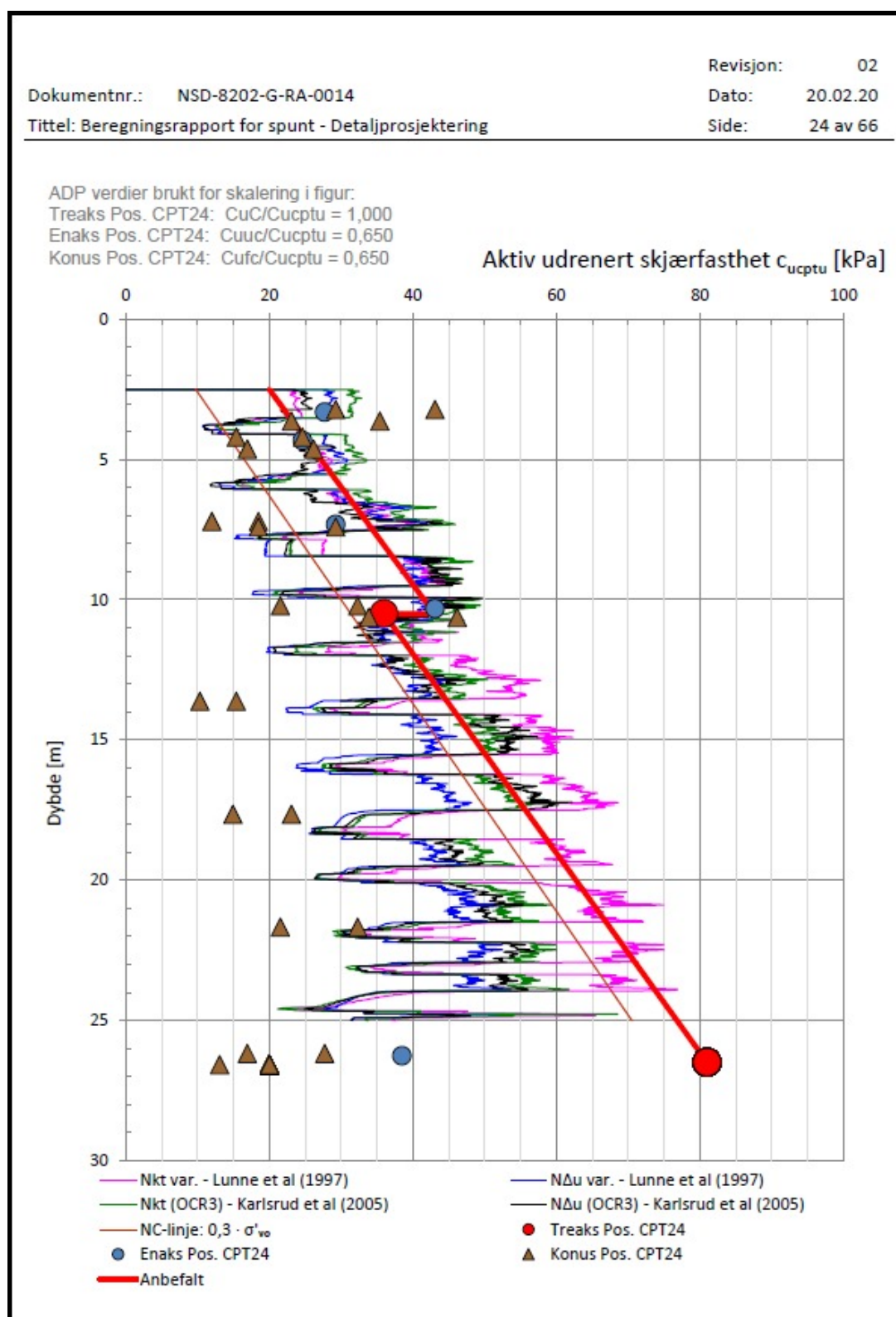
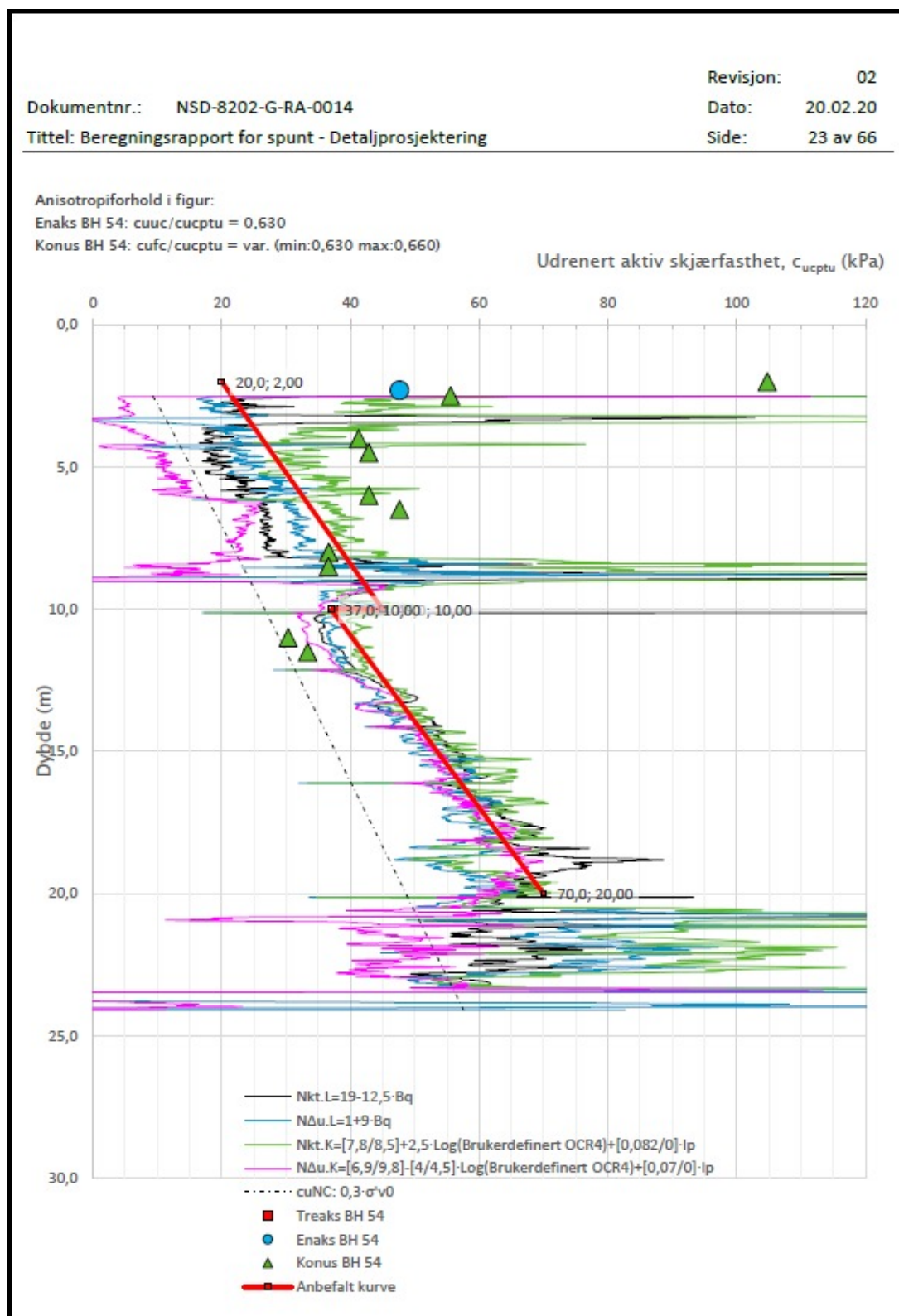


Figure A.3: Total sounding from borehole 69 in zone 1





**Figure A.4:** Shear profile based on a CPTU test from borehole 24 in zone 3. Profile interpreted by Norconsult



**Figure A.5:** Shear profile based on CPTU tests from borehole 54 in zone 3. Profile interpreted by Norconsult

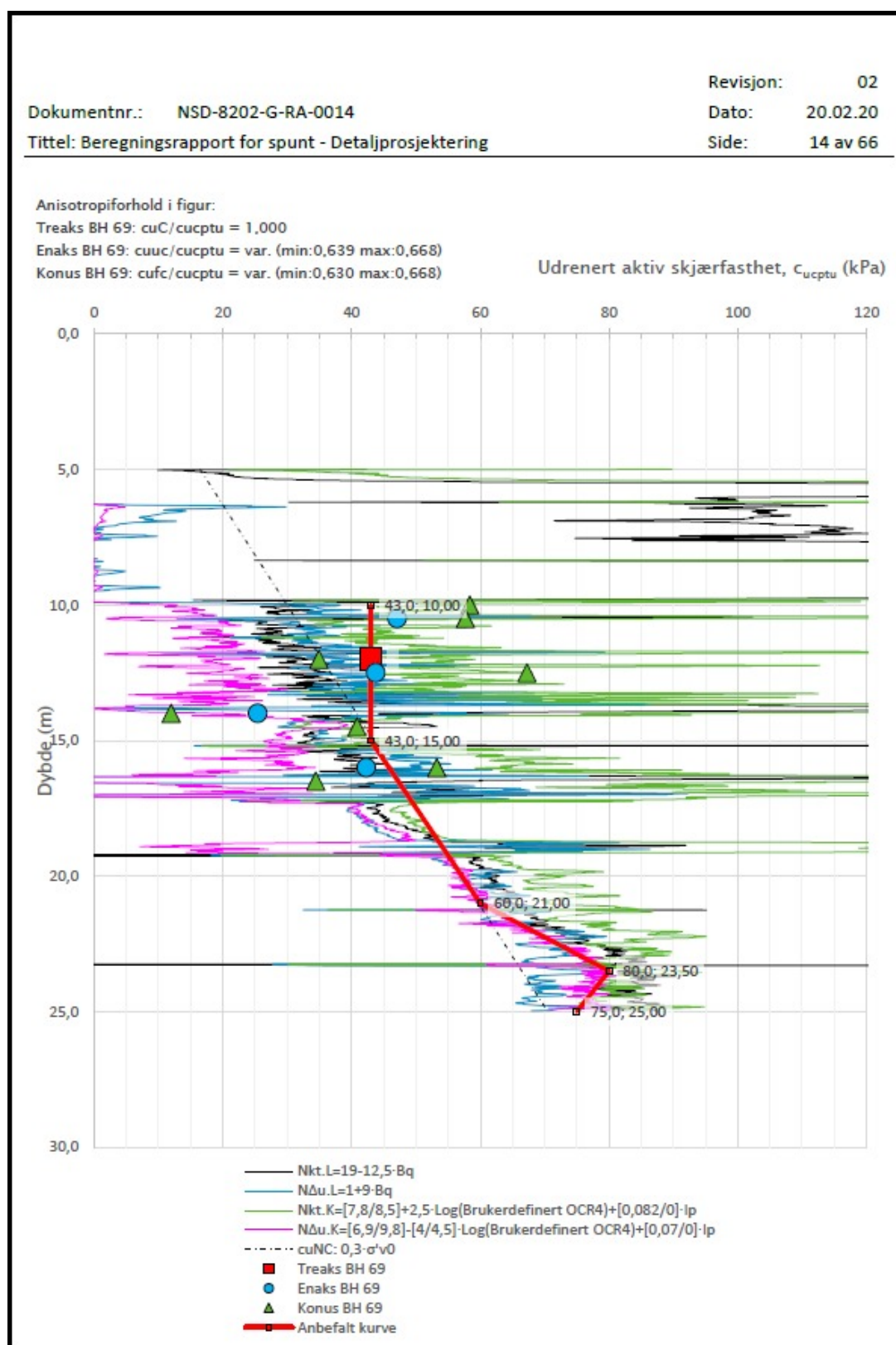


Figure A.6: Shear profile based on a CPTU in from borehole 69 zone 1. Profile interpreted by Norconsult

 NTNU

 NTNU

 NTNU

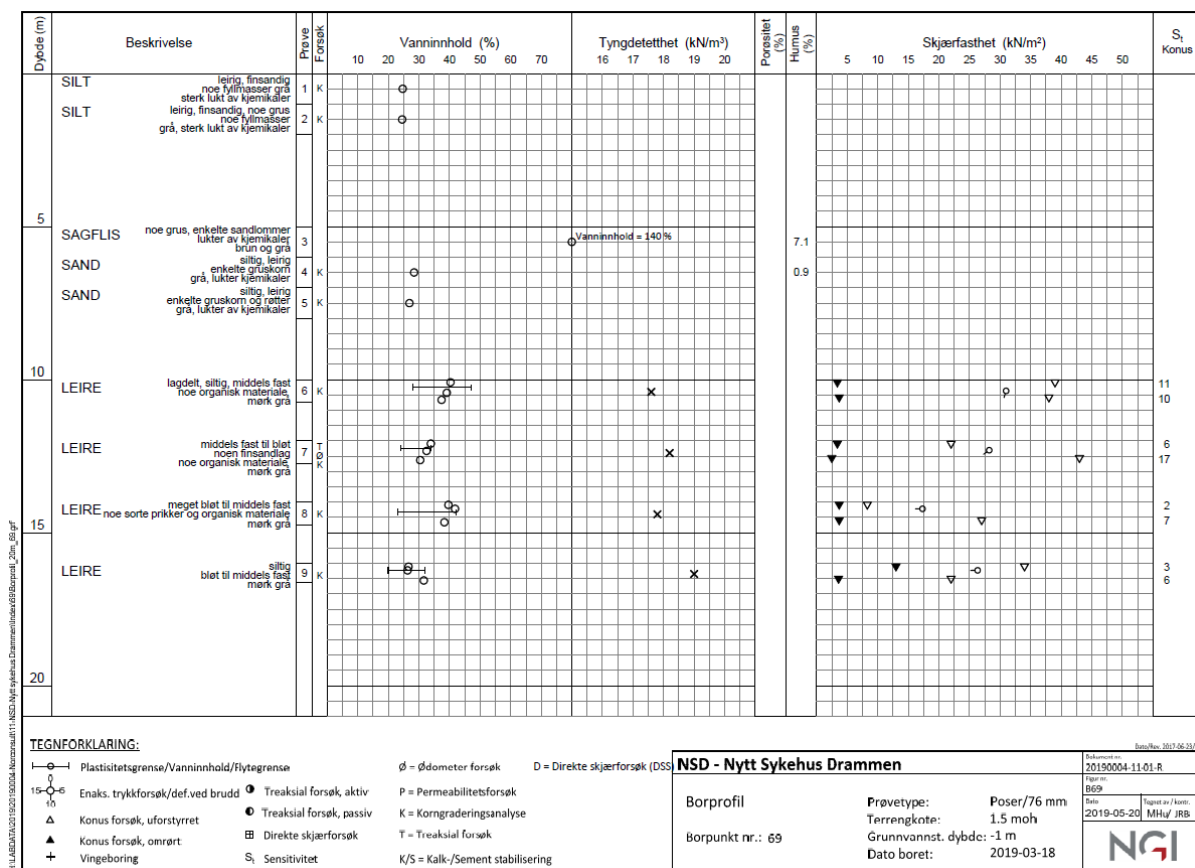


Figure A.10: Borehole profile for borehole 69 in zone 1

# Appendix B

## Optimization script

This appendix contains the optimization script used in the thesis.

```
# -*- coding: utf-8 -*-
"""
Created on Mon Apr 20 13:23:28 2020

@author: Jonas
"""
import imp
import os

plaxis_path=r'C:\Program Files\Plaxis\PLAXIS 2D'
plaxis_python_path=r'python\Lib\site-packages'

plx_modules = ['plxscripting','encryption']

for plx_module in plx_modules:
    print('importing module %s'%plx_module)
    found_module = imp.find_module(plx_module,[os.path.join(plaxis_path,plaxis_python_path)])
    plxscripting = imp.load_module(plx_module,*found_module)

from plxscripting.easy import *
s_i, g_i = new_server('localhost', 10000, password='Jonas123')
s_o, g_o = new_server('localhost', 10001, password='Jonas123')

import numpy as np
import pandas as pd

import modellering_sone3_NGIADP #SJEKK AT RIKTIG MODELL!
import calculations_sone3

meshing = {"Very coarse":0.12, "Coarse":0.08, "Medium":0.06, "Fine":0.04,"Very fine":0.03}
mesh = "Fine" #Specify wanted mesh refinement
```



```

sps = [18] #Sheet pile lengths to test
anchs = [40,45] #Anchor lengths to test

for s in sps:
    for a in anchs:
        modellering_sone3_NGIADP.modell() #Create the model
        calculations_sone3.calculations() #Set up phases

        c_sp = 1566 #cost of sheet pile wall per meter
        c_anch = 190.5 #cost of anchor per meter
        d_start = 0.5 #Step for making sensitivity
        lmin_anch = 12 #Boundary, anchor
        lmin_sp = 12.7 #Boundary, sheet pile wall

        g_i.gotostructures()
        sp_l_old = g_i.Point_1.y-g_i.Point_2.y
        anchor_l_old = g_i.Point_5.x-g_i.Point_6.x
        g_i.move(g_i.Line_8, 0, sp_l_old - s)# Set Sheet pile
        g_i.move(g_i.Line_2, anchor_l_old - a, 0) #Set Anch
        g_i.gotomesh()
        g_i.mesh(meshing[mesh])
        g_i.gotostages()
        g_i.calculate()

        sheet_p_lengths = []
        anchor_lengths = []
        vmsf_total = []
        costs = []

        b=1
        msf_new = g_i.Phase_8.Reached.SumMsf.value
        while msf_new > 1.4:
            print('\nRound ' + str(b))
            b+=1

            msf_old = g_i.Phase_8.Reached.SumMsf.value
            sp_l_old = g_i.Point_1.y-g_i.Point_2.y
            anchor_l_old = g_i.Point_5.x-g_i.Point_6.x
            cost = c_sp*sp_l_old + c_anch*anchor_l_old

            sheet_p_lengths.append(sp_l_old)
            anchor_lengths.append(anchor_l_old)
            vmsf_total.append(msf_old)
            costs.append(cost)
            print(anchor_l_old)
            print(sp_l_old)

```

```
#Calculate "sensitivites" of sheet pile and anchor
#sheet pile
if sp_l_old > lmin_sp:
    print('sheet pile sensitivity')
    g_i.gotostructures()
    g_i.move(g_i.Line_8, 0, d_start)
    g_i.gotomesh()
    g_i.mesh(meshing[mesh])
    g_i.gotostages()
    g_i.calculate()

    msf_sp_new = g_i.Phase_8.Reached.SumMsf.value #Extract new safety factor
    dmsf_sp = msf_old - msf_sp_new #Calculate change in safety factor

    g_i.gotostructures() #Reset sheet pile
    g_i.move(g_i.Line_8, 0, -d_start)

#anchor
if anchor_l_old > lmin_anch: #As long as length is above boundary
    print('anchor sensitivity')
    g_i.gotostructures()
    g_i.move(g_i.Line_2, d_start, 0)
    g_i.gotomesh()
    g_i.mesh(meshing[mesh])
    g_i.gotostages()
    g_i.calculate()

    msf_anch_new = g_i.Phase_8.Reached.SumMsf.value
    dmsf_anch = msf_old - msf_anch_new

    g_i.gotostructures()
    g_i.move(g_i.Line_2, -d_start, 0) #reset anchor

#Change step size depending on how close the safety factor is
if msf_new > 1.6:
    dl_sp = 1
    dl_anch = 4
elif msf_new > 1.5:
    dl_sp = 0.5
    dl_anch = 3
elif msf_new > 1.45:
    dl_sp = 0.5
    dl_anch = 2
else:
    dl_sp = 0.5
    dl_anch = 1
```

```

#Change dimension of relevant parameter depending on which gives the best "value"
if c_sp/c_anch*dmsf_anch < dmsf_sp and anchor_l_old >= lmin_anch:
    print('\nMoving anchor ' + str(dl_anch) + 'm')
    g_i.move(g_i.Line_2, dl_anch, 0) #More cost saved in shortening anchor
elif c_sp/c_anch*dmsf_anch >= dmsf_sp and sp_l_old >= lmin_sp:
    print('\nMoving sheet pile ' + str(dl_sp) + 'm')
    g_i.move(g_i.Line_8, 0, dl_sp) #More cost-efficient to shorten sheet piles
elif anchor_l_old >= lmin_anch:
    g_i.move(g_i.Line_2, dl_anch, 0) #if sheet pile wall boundary is reached
elif sp_l_old >= lmin_sp:
    g_i.move(g_i.Line_8, 0, dl_sp) #if anchor boundary is reached
else:
    print('Boundaries reached.')
    break

print('final step')
g_i.gotomesh()
g_i.mesh(meshing[mesh])
g_i.gotostages()
g_i.calculate()

msf_new = g_i.Phase_8.Reached.SumMsf.value #Set new safety factor

opt_path = pd.DataFrame(
    {'Sheet': sheet_p_lengths,
     'Anchor': anchor_lengths,
     'Msf': vmsf_total,
     'Cost': costs
    })

msf_old = g_i.Phase_8.Reached.SumMsf.value
sp_l_old = g_i.Point_1.y-g_i.Point_2.y
anchor_l_old = g_i.Point_5.x-g_i.Point_6.x
cost = c_sp*sp_l_old + c_anch*anchor_l_old

sheet_p_lengths.append(sp_l_old)
anchor_lengths.append(anchor_l_old)
vmsf_total.append(msf_old)
costs.append(cost)

opt_path = pd.DataFrame(
    {'Sheet': sheet_p_lengths,
     'Anchor': anchor_lengths,
     'Msf': vmsf_total,
     'Cost': costs
    })

print(opt_path)

```

## Appendix C

# Instructions for Python in PLAXIS 2D

This appendix will give some insight in the how to use Python in PLAXIS 2D. Further tips can be found in the master thesis by Grecu (2018).

TIPS AND TRICKS Not regenerate the model each time, instead moving the already existing points/lines

In order to allow Plaxis to communicate with Python, a boilerplate code must be set up as follows in the script.

```
from plxscripting.easy import *
s_i, g_i = new_server('localhost', 10000, password='password')
s_o, g_o = new_server('localhost', 10001, password='password')
```

Note that the password of the boilerplate must match the password used to initiate the SciTE editor in order for the editor to work.

If wanting to use another development environment, such as Spyder, the following code must be added at the very beginning:

```
import os
import imp

plaxis_path=r'C:\Program Files\Plaxis\PLAXIS 2D'
plaxis_python_path=r'python\Lib\site-packages'

plx_modules = ['plxscripting','encryption']

for plx_module in plx_modules:
    print('importing module %s'%plx_module)
    found_module = imp.find_module(plx_module,[os.path.join(plaxis_path,plaxis_python_path)])
    plxscripting = imp.load_module(plx_module,*found_module)
```

Once the boilerplate is in order, the project properties can be set.

```
# Start a new project
s_i.new()
# Set model and elements properties
g_i.setproperties("ModelType", "Planestrain",
"ElementType", "15-Noded")
g_i.SoilContour.initializerectangular(x1, y1, x2, y2)
```

Creating soils is done through first establishing a soil material and then setting certain properties. To set  $K_o$ , math has been imported in order to use mathematical functions and calculate the proper value.

```
# Create soil material sets
leire1 = g_i.soilmat()

leire1.setproperties("MaterialName", "NSD sone 1 leire_HS nedre",
"Colour", farger['Lysebla'],
"SoilModel", mat_modell['HS Small'],
"DrainageType", "Undrained (A)",
"gammaUnsat", 18.5,
"gammaSat", 18.5,
"DilatancyCutOff", True,
"einit", 0.6,
"emax", 999,
"E50ref", 3000,
"EoedRef", 2000,
"EurRef", 14000,
"powerm", 1,
"cref", 0,
"phi", 23,
"psi", 0,
"G0ref", 25000,
"gamma07", 0.00022,
"K0nc", 1-math.sin(23*math.pi/180),
"InterfaceStrength", "Manual",
"Rinter", 0.5,
"perm_primary_horizontal_axis", 0.751E-3,
"perm_vertical_axis", 0.751E-3,
"UseDefaultsFlow", "none")
```

Note that certain parameters can be slightly difficult to change, for instance setting an unload-reload stiffness in an HS-small model. A workaround to this problem can be editing the material after its creation and overriding the stiffness.

```
g_i.sps(leire2, "EurRef", 14000)
```

In order to create soil layers, a borehole must first be set. Then the soil layers can be set to different depths. The `soillayer`-function will create a layer starting at height 0 and down to the specified depth. If used again, the function will create a new layer starting from where the previous one ended. Note that the input to the function will be the width of the layer, not the coordinate of the bottom.

Finally, in order to create the desired height above 0 for the first layer, the `setsoillayer`-function can be utilized. When all the layers are created, a material can be assigned as seen below.

```
borehole1 = g_i.borehole(X) # X = X-coordinate of the borehole
borehole1.setproperties("Head", Y) # Y = Y-coordinate of the water head

kote_losmasse=1.7

g_i.soillayer(Y1) # Define depth of first layer
g_i.soillayer(Y2) # Define thickness of second layer
g_i.setsoillayerlevel(borehole1,0,kote_losmasse) #set height for first soil layer

#assign soils to the soil layers
g_i.Soils[0].Material = leire1
g_i.Soils[1].Material = leire2
```

## C.1 Structures

Lines are made using the *line-function*, and have to be created first before a variable name can be assigned. A variable name does not have to be assigned, but can be advantageous when handling the specific line at a later time.

Plates and anchors can be created either through creating them directly, or creating lines and assigning the plate and anchor property to them. Which method is chosen is not of much importance, but since interfaces have to be applied to a line and not a plate, creating the lines first could arguably make it slightly easier to keep order.

```
g_i.gotostructures() # Move to STRUCTURES tab

# Create lines
g_i.line(x1, y1, x2, y2) #Create Line 1
g_i.line(x1, y1, x2, y2) #Create Line 2
sheet_wall = g_i.Line_1 #Assign variable name to Line 1
anchor = g_i.Line_2

g_i.plate(sheet_wall) #Assign plate attributes to Line 1
g_i.n2nanchor(anchor) #Assign anchor attributes to Line 2

g_i.neginterface(spunt) # Negative skirt interface
g_i.posinterface(spunt) # Positive skirt interface
```

Plate materials can be slightly problematic to create, as there seems to be some issues with certain values overriding others. A way to create both an isotropic and a non-isotropic plate material is shown below.

```
steel1 = g_i.platemat() #Create plate materials
steel2 = g_i.platemat()

steel1.setproperties("MaterialName", "AZ 17-700 355 forspunt ",
"Colour", 16711680,
"MaterialNumber", 1, #0=elastisk 1=elastopl 2=elastopl w residual strength
"Elasticity", 0,
"IsIsotropic", False,
"IsEndBearing", False,
"EA", 2.793E6,
"EA2", 139.7E3,
"EI", 76.08E3,
"nu", 0,
"w", 1.045,
"d", 0.5717,
"Mp", 585,
"Np", 4497,
"Np2", 150,
"Gref", 2442541.60767924)

steel2.setproperties("MaterialName", "AZ 17-700 355 bakspunt ",
"Colour", 16711680,
"MaterialNumber", 1, #0=elastisk 1=elastopl 2=elastopl w residual strength
"Elasticity", 0,
"IsIsotropic", True,
"IsEndBearing", False,
"EA", 2.646E6,
"EA2", 2.646E6,
"EI", 38.09E3,
"nu", 0,
"w", 0.9904,
"d", 0.4156,
"Mp", 405.7,
"Np", 4260,
"Np2", 4260,
"Gref", 3182994.23740057)

g_i.Plate_1.Material = steel1 #Assign material properties to materials
g_i.Plate_2.Material = steel2
```

Note that the Gref-value must be specified in order for the material to be created. If omitted, an error message will appear, stating that a plate stiffness must be specified.

The apparent easiest way to find the proper values at the time of writing is to create the plate material in Plaxis and using the *echo-command* in the command line in order to find the correct values.

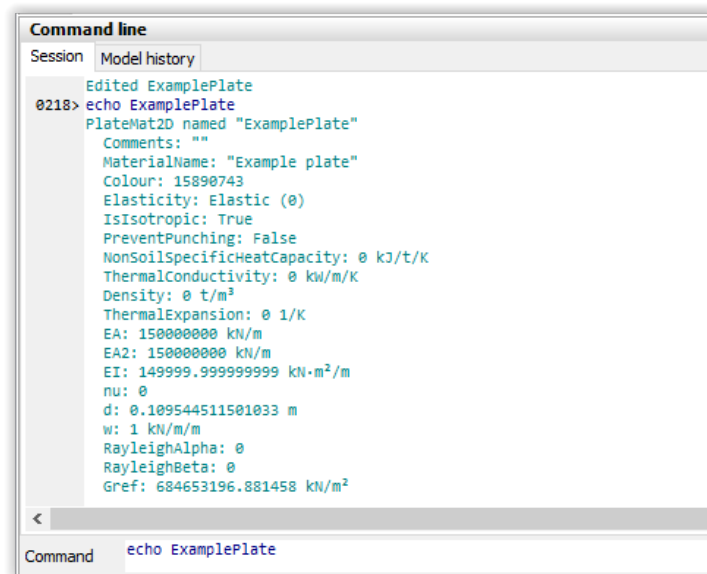


Figure C.1: Using the echo command to find plate input parameters

Creating anchor materials is done in the same way as for plate materials.

```
losmassestag = g_i.anchor_mat() #Create anchor material

losmassestag.setproperties("MaterialName", "Lissestag 7 lisser",
    "Colour", 0,
    "MaterialNumber", 1,
    "Elasticity", 1, #0=elastic 1=elastopl 2=elastopl w residual strength
    "Lspacing", 4.2,
    "EA", 2058E5,
    "FMaxTens", 1183,
    "FMaxComp", 1)

g_i.NodeToNodeAnchor_1.Material = losmassestag #Assign properties to anchor material
```

In order to create a lineload, the following command can be used.

```
g_i.lineload(X1_last, Y1_last, X2_last, Y2_last, "qy_start", -1)
```



## C.2 Mesh

Meshing is done with the *mesh*-function. Creating a dictionary is not necessary, but can help to not have to learn all the mesh sizes.

```
g_i.gotomesh()

meshing = {"Very coarse":0.12,
"Coarse":0.08,
"Medium":0.06,
"Fine":0.04,
"Very fine":0.03}

g_i.mesh(meshing["Very fine"]) #Set mesh
```

## C.3 Flow conditions and staged construction

Creation of phases is done in the flow or staged construction sections. Some examples of different actions is shown below:

```
g_i.gotoflow() #Go to flow section

g_i.InitialPhase.PorePresCalcType = "Steady state groundwater flow"

#Phase 1 (Sheet pile)
g_i.phase(g_i.Initialphase) #Add Phase_1
g_i.setcurrentphase(g_i.Phase_1) #Make Phase_1 current
g_i.Phase_1.Identification = "Sheet pile" # Name the phase

g_i.Plates.activate(g_i.Phase_1) # Activate plates
g_i.Interfaces.activate(g_i.Phase_1) # Activate interfaces
g_i.GroundwaterFlowBCs.activate(g_i.Phase_1) # Activate groundwater flow BCs

# Phase_2 (First excavation)
g_i.phase(g_i.Phase_1) # Add Phase_2
g_i.setcurrentphase(g_i.Phase_2) # Make Phase_2 current
g_i.Phase_2.Identification = 'First excavation' # Name the phase

g_i.Phase_2.Deform.ResetDisplacementsToZero = True #Reset deformations
g_i.BoreholePolygon_1_1.deactivate(g_i.Phase_2) #Deactivate wanted polygon
g_i.LineLoad_4_1.activate(g_i.Phase_2) #Activate line loads

g_i.waterlevel(x1, y1, x2, y2) #Set a new groundwater level
g_i.GWFlowBaseBC_12.Behaviour[g_i.Phase_2] = "Closed" #Create closed flow conditions

#Set the water level to a custom level
g_i.WaterConditions_1_2.Conditions[g_i.Phase_2] = "Custom level"
g_i.WaterConditions_1_2.Level[g_i.Phase_2] = g_i.UserWaterLevel_1
```

Activating the anchor and setting a prestress force can be done as follows:

```
g_i.phase(g_i.Phase_2) # Add Phase_3
g_i.setcurrentphase(g_i.Phase_3) # Make Phase_3 current
g_i.Phase_3.Identification = "Establish anchor" # Name the phase

g_i.NodeToNodeAnchors.activate(g_i.Phase_3)

g_i.NodeToNodeAnchor_1_1.AdjustPrestress[g_i.Phase_3] = True
g_i.NodeToNodeAnchor_1_1.PrestressForce[g_i.Phase_3] = 595
```

If needed, examples to set up a safety analysis and a consolidation analysis can be found below:

```
g_i.setcurrentphase(g_i.Phase_5)
g_i.Phase_5.Identification = "Safety"

g_i.Phase_5.DeformCalcType = "Safety" # Define the calculation type
g_i.Phase_5.Deform.UseDefaultIterationParams = False
g_i.Phase_5.Deform.MaxSteps = 200

#Phase 6
g_i.setcurrentphase(g_i.Phase_6)
g_i.Phase_6.Identification = "Consolidation"

g_i.Phase_6.DeformCalcType = "Consolidation"
g_i.Phase_6.Deform.LoadingType = "Minimum excess pore pressure"
```

Finally, once all phases are set, the calculations can be activated using the *calculate*-command.

```
g_i.gotostages()
g_i.calculate()
```

