Numerical Framework for Modeling Cyclic Degradation of Soils

Johannes Osorio Mydland

Civil and Environmental Engineering
Submission date: June 2017
Supervisor: Gustav Grimstad, IBM

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

NORWEGIAN UNIVERSITY OF SCIENCE AND TECHNOLOGY
DEPARTMENT OF CIVIL AND ENVIRONMENTAL ENGINEERING

Report Title: Numerical Framework for Modeling Cyclic Degradation of Soils	Date: 08.06.2017 Number of pages: 99			
	Master Thesis	x	Project Work	
Name: Johannes Mydland				
Professor in charge/supervisor: Gustav Grimstad				
Other external professional contacts/supervisors:				

Abstract:
For many years fatigue in soils has been a major design challenge. Due to cyclic loading the material behavior changes and permanent deformations are observed. Although explicit cyclic calculation methods exists, none are ideal for cyclic design computation. The available methods are currently either to simplified, labor-intensive or inaccurate.

A framework based on Python, Matlab and Fortran has been developed for creating soil models and doing cyclic simulations. The script Destructor is an interface for PLAXIS 2D and automates the whole calculation process. An average and a cyclic material model based on elasto-plasticity and non-linear elasticity has been created and coupled together. A state variable used to model material softening is used to couple the materials. The material equations were defined in Matlab and generated as object-oriented Fortran code ready to be compiled together with a Fortran wrapper.

Plane strain simulations has been done for monopiles, represented by a wall, and gravity based structures. The results are encouraging. The material models and Python framework is able to simulate loss of ultimate strength capacity, higher long-term settlements, permanent deformations and a change in pore pressure after cyclic loading. The verification was done qualitative.

Keywords:

1. Soil models with destructuration
2. Framework for cyclic degradation calculation
3. FEM using PLAXIS 2D with Python
4. Object-Oriented coded soil materials

Jomy

MASTER DEGREE THESIS
Spring 2017
for

Student: Johannes Mydland

Numerical Framework for Modeling Cyclic Degradation of Soils

BACKGROUND

For any given structure, the exterior loads must be transferred to the ground. The environmental loads, which are irregular and random in nature, cause degradation in the soil. This result in an increase in pore pressure and may also lead to permanent deformations, lower capacity and higher long-term settlements.

In recent years, the interest for cyclic soil degradation has increased, especially with regards to offshore structures. Although several methods are available for designing foundation exposed to cyclic loading, an easy and reliable method is yet to be determined.

TASK

A framework for calculating degradation on soil is to be developed. The framework needs to be able to calculate boundary problems using non-linear finite element method and soil materials based on continuum mechanics. The framework should be intuitive to use and the end user should not be required to know programming to utilize the program. Soil materials used for calculation of cyclic degradation needs also to be determined and developed. The proposed framework should be verified based on observed cyclic behavior in tests.

Task description

A framework based on Python should be created to be used with PLAXIS 2D. With a graphical interface for user inputs. Material models with state variable(s) used to model softening/degradation will be used. The soil models for average and cyclic part of the behavior should be coupled together via transformation of relevant state variables.

Objective and purpose

The overall goal is to create a framework for simulating cyclic degradation of soils. For this a script that automates the process. Material models capable of simulating degradation must also be developed.

Subtasks and research questions

For the modeling cyclic degradation, a theoretical background is required. The focus should be on what causes cyclic degradation and how to model it.

Professor in charge:

Gustav Grimstad

Department of Civil and Transport Engineering, NTNU
Date: 09.06.2017

Gustav Grimstad
Professor in charge (signature)

Preface

The Master Thesis has been written in relation to Civil and Environmental Engineering degree with major in geotechnics. The work was carried out during the spring semester of 2017 at Norwegian University of Science and Technology (Trondheim). The topic was proposed and supervised by Professor Gustav Grimstad.

Trondheim, 2017-06-07

Jomy

Johannes Mydland

Acknowledgment

Many thanks goes to Gustav Grimstad. He proposed an assignment that suited me perfectly and given me a nudge in the right direction whenever I was stuck. We've also have had some fruitful discussions which in turn led to a greater understanding of cyclic soil behavior. I couldn't have asked for a better supervisor.

Steinar Nordal has been my second go-to guy. His outstanding knowledge of finite element and soil modeling is inspirational. Whenever asked a question he either has the answer or a paper with the solution. His help has been deeply appreciated.

This thesis would not have been feasible in the amount of time without the help of Jon A. Rønningen. His object-oriented soil material code has served as inspiration for much of the framework presented here. He has been ever so helpful with discussing and explaining his material models. Most importantly he has always taken the time to be readily available when asked.

Thanks is also due to chef Håkon Eggebø. His cooking enthusiasm has served me many meals over the years and allowed me to focus more on work. Moreover he has also been a great sport and proof-read the whole thesis.

Lastly huge thanks to Eirin Haugen for keeping me sane the last years. She has reminded me more than once to take a break and to focus on the important things in life: wine, sushi & movienights.

J.M.

Johannes Mydland

Abstract

For many years fatigue in soils has been a major design challenge. Due to cyclic loading the material behavior changes and permanent deformations are observed. Although explicit cyclic calculation methods exists, none are ideal for cyclic design computation. The available methods are currently either to simplified, labor-intensive or inaccurate.

A framework based on Python, Matlab and Fortran has been developed for creating soil models and doing cyclic simulations. The script Destructor is an interface for PLAXIS 2D and automates the whole calculation process. An average and a cyclic material model based on elasto-plasticity and non-linear elasticity has been created and coupled together. A state variable used to model material softening is used to couple the materials. The material equations were defined in Matlab and generated as object-oriented Fortran code ready to be compiled together with a Fortran wrapper.

Plane strain simulations has been done for monopiles, represented by a wall, and gravity based structures. The results are encouraging. The material models and Python framework is able to simulate loss of ultimate strength capacity, higher long-term settlements, permanent deformations and a change in pore pressure after cyclic loading. The verification was done qualitative.

Abstrakt

I mange år har utmatting av jord vært en stor utfordring. På grunn av syklisk belastning vil materialets oppførsel forandre seg og permanente deformasjoner er observert. Selv om eksplisitte sykliske beregningsmetoeder eksisterer, finnes det ingen idelle sykliske prosjekteringsmetoder. De tilgjengelige metodene er for øyeblikket enten for forenklet, arbeidskrevende eller unøyaktige.

Et rammeverk basert på Python, Matlab og Fortran har blitt utviklet for å lage jordmodeller og gjøre sykliske simuleringer. Skriptet Destructor er et grensesnitt for PLAXIS 2D og automatiserer hele beregningsprosessen. Et gjennomsnittlig og en syklisk materialmodell basert på elasto-plastisk og ikke-lineær elastisitet har blitt laget og koblet sammen. En tilstands variabel brukt til å modellere mykning har blitt brukt til å koble sammen materialene. Materialligningene er definert i Matlab og er generert som objekt-orientert Fortran-kode som er klar til å kompileres sammen med et Fortran rammeverk.

Plan-tøyningssimuleringer har blitt utført for monopeler, representert som en vegg, og en gravitasjonsbasert konstruksjon. Resultatene er lovende. Material modellene og Python rammeverket klarer å simulere tap av maksimal styrkekapasitet, høyere langtidssetninger, permanent deformasjon og en forandring i poretrykket etter syklisk belastning. Verifisering er gjort kvalitativt.

Contents

Chapter 1 Introduction

1.1 Background

Since the discovery of oil in the early 70's, the study of cyclic degradation on soil has been given a special attention. Although the environmental loads like wind, ocean waves and earthquake are irregular in nature, they may for certain situations be regarded as constant static load. In the case of offshore structures the ever changing load situation is of great importance. The constant pounding from the ocean waves causes the soil material to wear and tear. Even worse it may be in the case of a storm.

Today there is big interest in renewable energy sources. While there are many options, not all are economically feasible. There's high hopes that the Offshore Wind Turbine (OWT) can be used extensively. In contrast to the oil industry, wind turbines has a low earning and therefor a greater focus on expenses (cyl). One of the more costly areas in designing an OWT is the foundation.

When designing an offshore structure there are many uncertainties. The wind and waves are both random irregular loads. The natural frequency of the structure may also be a source of added load. And as always there is a high uncertainty in the mechanical behavior of the soil.

Even though methods exists for designing foundation exposed to cyclic loading, none of them are ideal. Certain methods are to unreliable while other methods requires extensive laboratory testing. And even then the data must be organized and interpreted, which may be fallible.

It's the hope that a simpler calculation method can reduce expenses and increase the accuracy of soil behavior. For this a framework and new soil models are needed. Ideally the soil models will also enlighten the designing geotechnical engineer to understand the phenomena of cyclic degradation in soil.

Problem Formulation

The overall goal is to create a reliable calculation method for foundation design exposed to cyclic load. Due to time limitations the focus will be on creating the framework for cyclic calculations. The new method should be easy to use and automates a lot of the processes where applicable. The method should also be of the explicit type, meaning that only the major permanent effects are accounted for. An implicit method that tries to follow each cycle is considered too unreliable and time-consuming.

It's here assumed that the cyclic problem can be divided into two soil models. The first model accounts for the permanent effects, denoted the average model. This model should be able to be used in standalone calculations. The other soil model, here called the cyclic model, has the purpose of causing a degradation for a cyclic load. Together they may be used to calculate fatigue in the soil.

The created framework and soil models should also be tested that they actually work together and gives reasonable results. The verification here will be qualitative based on known phenomena that occurs during cyclic loading. For example when an undrained material is cycled there should be permanent deformation, excess pore pressure and lower yield capacity.

Literature Survey

The framework and the degradation formulation is mostly based on the discoveries presented by Knut H. Andersen and NGI. Most of the cyclic results and figures here is gathered from the McClelland Lecture by Knut H. Andersen (Andersen, 2015). A few selected articles has also been covered regarding calculation methods. The UDCAM (Jostad et al., 2014; Grimstad et al., 2012), PDCAM (Jostad et al., 2015), Master's thesis by Aleksander S. Gundersen & Jon-Michael Josefsen (Gundersen and Josefsen, 2016) and PhD thesis by Fang Cai (Cai, 2015) has been studied for this purpose.

To account for degradation the soil models with a structure has been studied. Lectures and papers presented by Burland (Burland, 1990), Yin (Yin and Karstunen, 2008) and Jon A. Rønningen (Rønningen et al., 2014) has also been reviewed for coupling parameters to be used with the cyclic degradation.

What Remains to be Done?

Although cyclic degradation on soil has been given a lot of attention, there's still intense undergoing study. There are currently no exceedingly good ways of carrying out a cyclic calculation, and the ones available are labor-intensive and require a lot of test samples (Gundersen and Josefsen, 2016). Also the understanding of what really causes cyclic degradation is still not fully understood. Implicit and explicit methods are still being developed but they have their different usage. There's also the issue of knowing what happens in a general stress state when exposed to cyclic loading. Currently mostly cyclic DSS and Triaxial tests are available, meaning what happens for general stress states is yet unknown.

1.2 Objectives

The Master thesis has been divided into three major objectives. Each objective in itself is vital for the overall success of accurately modeling the cyclic degradation.

- Firstly a solid understanding of fundamental material modeling and cyclic behavior in soil is needed. Ideally the literature will give some indications of the governing soil mechanics during cyclic loading. Some existing methods for design calculation should also be explored.
- Soil models that are designed for cyclic degradation needs to be created. By themselves the models should be able to work for testing and modeling the response individually. The average and cyclic model should also be easily coupled if needed. It would also be a great advantage if the materials could be generated directly from some mathematical equations, instead of defining everything explicitly in computer code.
- Lastly a framework which automates the cyclic calculation should be developed. The input ought to primarily be given in the program or script developed and should be simple & intuitive to use. Due to time limitations a framework will only be developed for PLAXIS 2D.

1.3 Limitations

Regarding soil modeling, small strains and explicit cyclic analysis will only be considered. There will not be a thorough examination of visco-plastic materials, although they will indirectly be mentioned. The models presented are isotropic elastic and it's assumed that the cyclic material is undrained. This implies that the model is total stress based and that only shape distortion can cause degradation in the cyclic model. Although partial drainage could be included, these effects are here ignored. The focus will be on undrained cyclic soil behavior.

The framework will only be created to work for a PLAXIS 2D plane strain problem, but it should easily be possible to extend the framework to do 3D calculations. Little consideration has been given to the choice of computer algorithms and numerical schemes, although some methods will be mentioned in conjunction with material modeling.

One major assumption is that the cyclic degradation can be expressed by an average and a cyclic soil material. This implies that the load somehow must be divided into an average and a cyclic part. The same is true for stresses and strains. Whether or not this is realistic might be controversial.

Lastly it's emphasized that the verification shall be done qualitatively. A full-worthy analysis should be carried out with some good soil models on an actual problem. The verification will also not thoroughly test the materials for all types of load combinations and soil parameters.

1.4 Methods

The response expected in an cyclic boundary problem is to be analyzed by numerical simulations. Two material models are coupled together with a framework to solve a generic cyclic problem. It's here assumed that the cyclic loading is the primary cause of degradation in soil and can be modeled by using material models with a structure variable. The calculation is done in three steps. Firstly the static response due to a load change is calculated. Secondly a cyclic phase with a new material model is carried out and causes degradation. Finally equilibrium and compatibility must be reacquired due to changes in the structure. Verification will be based on the basis of qualitative reasoning.

1.5 Structure of the Report

The Master thesis is divided into nine chapter as shown in figure 1.1. Some chapters can be read individually, but it's recommended to read from the start to get the whole picture. In chapter 2 & 3 some selected theory is shown. Chapter 2 focuses on the basics of elasto-plastic material modeling. Some important notation and concepts will also be introduced. In chapter 3 the cyclic behavior of undrained soil will be studied. The development of cyclic contour diagrams and the use of these diagrams will also be shown.

Chapter 4 & 5 is about the requirements of the soil models. A state parameter called structure is introduced in the average model and some selected soil models are reviewed with regards to cyclic degradation. The cyclic material model is highly based on the discovery done in chapter 3. A degradation formulation based on a cyclic contour is developed, and the background for creating more advanced models is shown.

Chapter 6 & 7 is more independent from the other chapters since it focuses more on actual programming. Some of the sections are written more in a textbook style and is meant to explain the central aspects that the framework is based on. In chapter 6 Python is introduced and shown how it can be used together with PLAXIS to do finite element calculations. A short introduction of graphical user interface programming is shown for the purpose of code maintenance and readability. Lastly the developed framework in Python is explained. Chapter 7 is based on the work done by Jon A. Rønningen. The idea is that the mathematical equations are written in Matlab and from there gradients, symbolics and equations are solved. The Matlab symbolics is afterwards converted to Fortran code and can from there easily be compiled to a material model without having to modify a lot of Fortran code. Some verification of this framework is shown in the end of the chapter.

In chapter 8 some simulations are carried out. The coupling between the material models and the framework is explained and some results are shown.

In the last chapter the important discoveries are summarized and discussed. There will also be recommendation for what else can now be done with the developed framework and suggestion for future soil models.

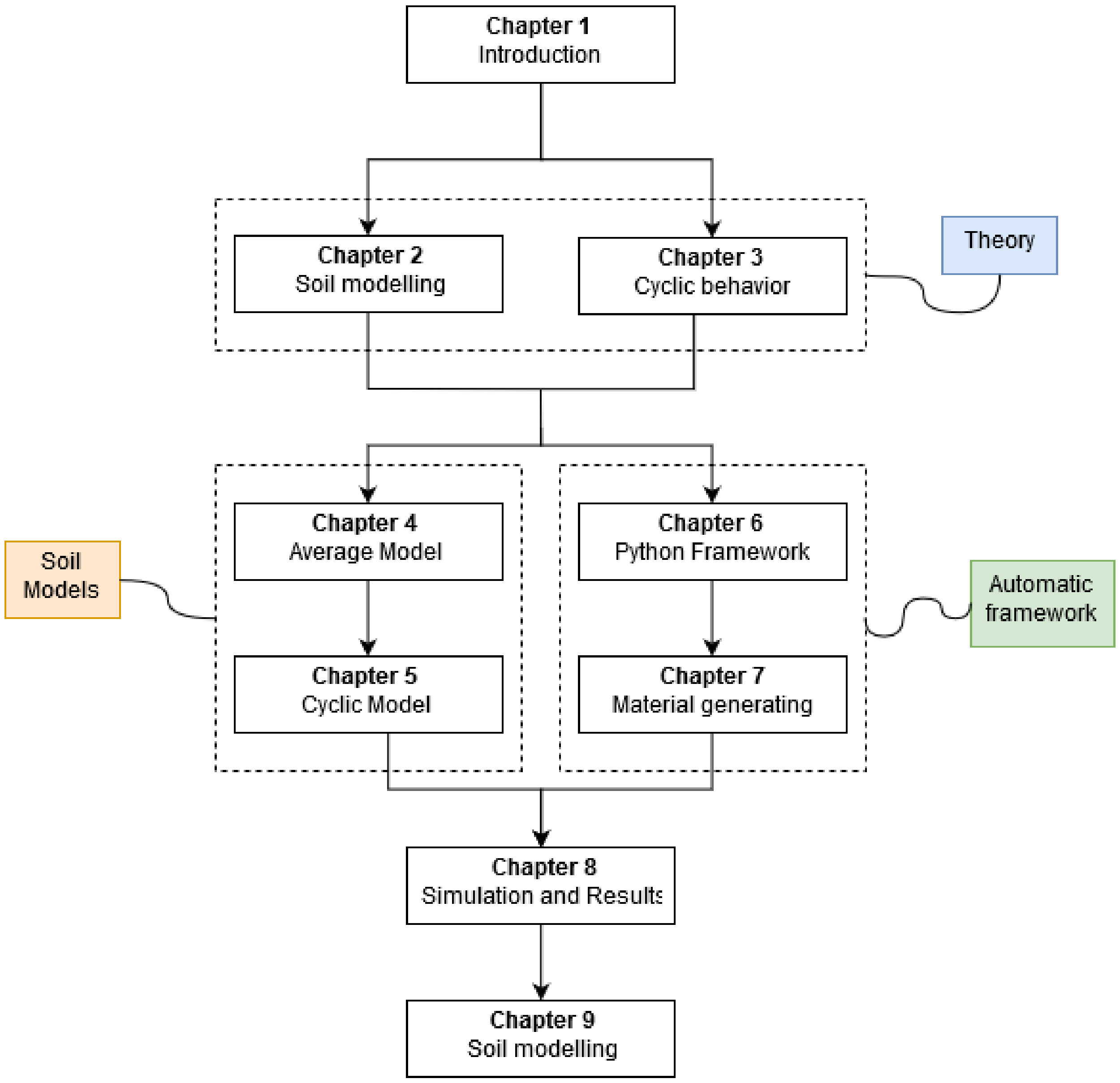


Figure 1.1: Structure of the Report.

Chapter 2 Soil Modeling

2.1 Short Introduction to Soil Modeling and Finite Element Analysis

Even though soil is a multiphase material composed of many particles, it is now due to the extensive use of Finite Element (FE) simulations common practice to model the soil as a continuum. The goal is to capture the overall response of the soil when subjected to loads and displacements. Mathematical relationship between forces and displacements (macro-modeling), or more commonly between stresses and strains defines the constitutive material models.

Originally metal, and later soil, has been modeled using the elasto-plastic framework with great success. A uniaxial tensile test of a fictive steel-like material is shown in figure 2.1. It illustrates some of the key aspects regarding elasto-plasticity. The response is purely elastic up to a certain stress state denoted as the yield strength. Afterwards plastic flow occurs where the energy is dissipated as heat. Due to rearrangement in the grains, the material will harden up to necking. Upon unloading and reloading the response will be purely elastic, but now with a new yield capacity. Note that necking is a geometric instability and marks the end of small deformation theory. Necking is not a problem in soil mechanics since soils cannot sustain high tensile stresses. Nonetheless it is common to limit the Finite Element Analysis (FEA) to small deformation theory since it simplifies the problem and one avoids geometric nonlinearities.

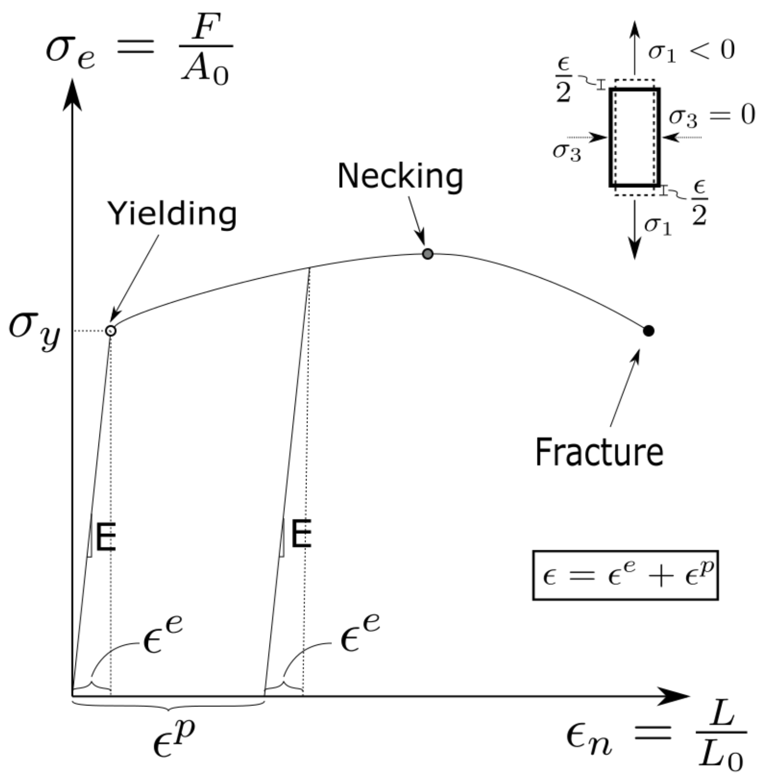


Figure 2.1: Illustration of a uniaxial tensile test on a steel-like material.

In a FEA a real physical problem is modeled in a FE program environment. The model is afterwards chopped up into smaller elements, a process called meshing. The elements normally consists of nodes and stress points as indicated in figure 2.2. The constitutive material model gives the elements its stiffness. The deformation of the element is interpolated from a linear combination of the node displacements and shape functions, classically Lagrangian polynomials. The stiffness is normally calculated by numerical integration, also known as a quadrature. In the case of polynomial shape function one can ensure exact integration by using the Gaussian quadrature. In that particular case it's also normal to position the stress points according to the Gaussian quadrature. These particular position is also known as Gauss Points (GP). The elements are afterwards put together and used to solve a Partial Differential Equation (PDE).

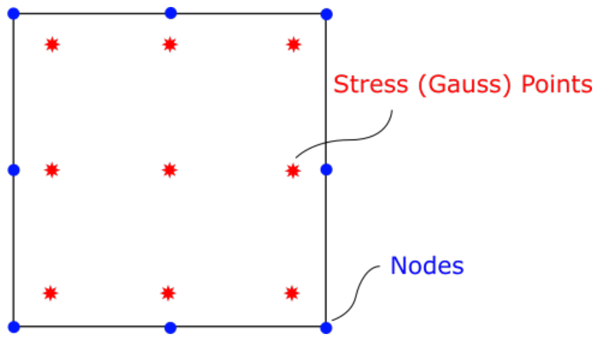


Figure 2.2: Q8 element with corresponding nodes and stress points.

2.2 Stress, Strain, Invariants & Tensors

In classical continuum mechanics a stress state can be represented as a matrix in each GP by 9 components. This may be visualized as an infinitesimal cubic element as shown in figure 2.3. Due to conservation of angular momentum the stress matrix can be shown to be symmetric and thereby reduced to 6 independent components. Note that in soil modeling it's common convention to define compression as positive.

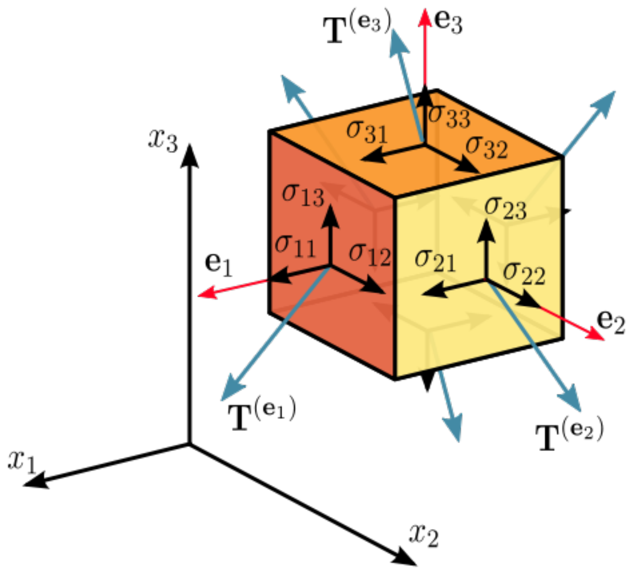


Figure 2.3: Infinitesimal stress cube.
From: https://en.wikipedia.org/wiki/Cauchy_stress_tensor/

Since a stress state must be the same independent of any chosen coordinate system (Cartesian, Spherical, etc.), this implies that the stress must in fact be a tensor. Tensors has a special type of linear mapping that ensures that the same value is represented in any set of chosen unit vectors, also known as bases. From a tensor one can calculate some values that stays the same for any chosen set of bases. These values are known as invariants. Examples of invariants in an Euclidean space is length, area and volume. Depending on the tensor there can be one or more linear independent invariants. Since it's hard to imagine a 6-dimensional space, the tensor is often rewritten in terms of invariants and the corresponding set of unit vectors. This is also known as solving for the eigenvalues and eigenvectors.

In material mechanics σ_1, σ_2 and σ_3 are invariants and more commonly called principal stresses. In the principal space the off-diagonal entries in the stress matrix are all zero. This reduces the situation down to a 3-dimensional space. The invariants can also be rewritten as linear combination of each other. Some common invariants are the hydrostatic pressure p, the deviatoric stress q an the Lode angle θ. A figure of the principal space is shown in figure 2.4.

$$\begin{aligned}
p &= \frac{\sigma_1 + \sigma_2 + \sigma_3}{3} \\
q &= \sqrt{3J_2} = \sqrt{\frac{1}{2} \cdot [(\sigma_1 - \sigma_2)^2 + (\sigma_2 - \sigma_3)^2 + (\sigma_3 - \sigma_1)^2]} \\
\theta &= \frac{1}{3} \arcsin\left(\frac{-3\sqrt{3}J_3}{2\sqrt{(J_2)^3}}\right), \qquad J_3 = (\sigma_1 - p)(\sigma_2 - p)(\sigma_3 - p)
\end{aligned} \tag{2.1}$$

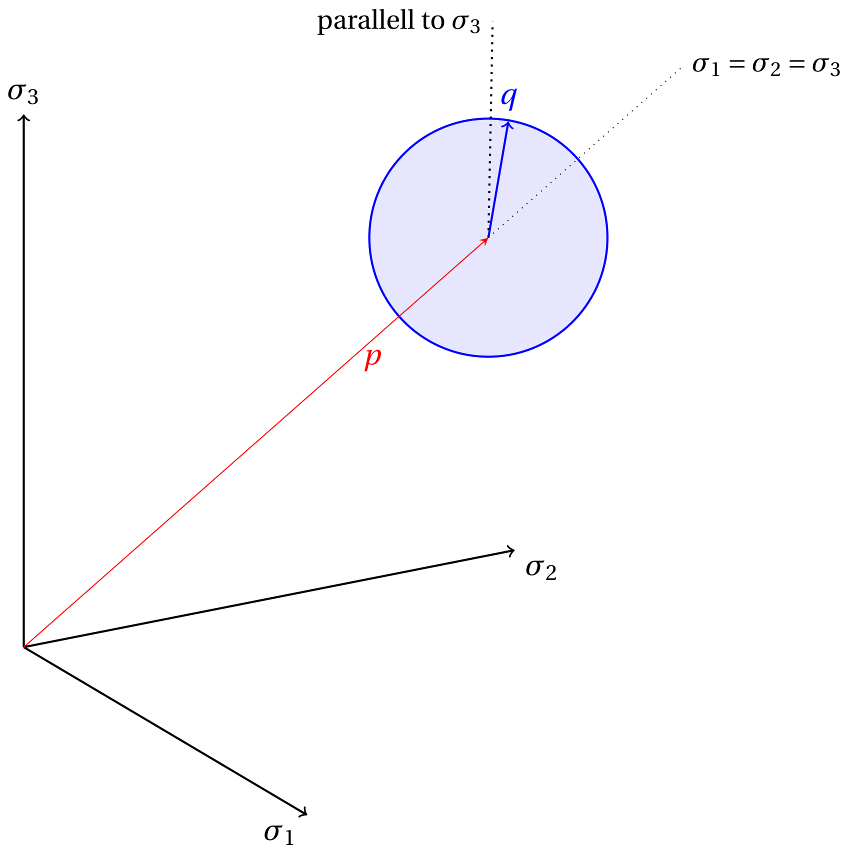


Figure 2.4: Principal space

The hydrostatic axis is parallel to the p-vector and defines an isotropic stress state. Along this axis there are no shear stresses in the material. Normal to this line is the deviatoric plane also referred to as the π-plane. In this plane the shear stresses increases radially from the hydrostatic axis. The Lode angle can be thought of as an angle between the q-vector and one of the principal stresses on the π-plane. Depending on the formulation of the Lode angle the geometry can vary, but a function formulated on the Lode angle will always be symmetrical on this plane. Note that under triaxial conditions ($\sigma_2 = \sigma_3$) the deviatoric stress is $q = \sigma_{axial} - \sigma_{radiell} = \sigma_1 - \sigma_3$.

A computer in general only can handle vector based computations. It's therefore advantageous to describe the stress state as a vector. Using Mandel or Voigt (Hopperstand and Børvik, 2015) notation a stress vector can conveniently be used in material calculations depending on the problem at hand.

$$\begin{aligned}\tilde{\boldsymbol{\sigma}}^m &= \begin{bmatrix}\sigma_{11} & \sigma_{22} & \sigma_{33} & \sqrt{2}\sigma_{21} & \sqrt{2}\sigma_{13} & \sqrt{2}\sigma_{12}\end{bmatrix}^T \\ \tilde{\boldsymbol{\sigma}}^v &= \begin{bmatrix}\sigma_{11} & \sigma_{22} & \sigma_{33} & \sigma_{21} & \sigma_{13} & \sigma_{12}\end{bmatrix}^T\end{aligned} \tag{2.2}$$

The same arguments can be made for strains. Limiting us to small strain theory the strains can be linearized. In that way energy compatibility is avoided and the problems are easier to solve since the coordinate system must not be updated. As for the stresses, invariants for the strain tensor can also be calculated. Some notations for strain and calculation methods (Brinkgreve et al., 2017) is listed below where $\boldsymbol{u} = [u_1, u_2, u_3]$ is the vector displacement in a node.

$$\begin{aligned}\tilde{\boldsymbol{\epsilon}}^v &= \begin{bmatrix}\epsilon_{11} & \epsilon_{22} & \epsilon_{33} & \gamma_{21} & \gamma_{13} & \gamma_{12}\end{bmatrix}^T \\ \epsilon_{ii} &= \frac{\partial u_i}{\partial x_i} \quad \text{where} \quad i, j \subset [1,2,3] \\ \gamma_{ij} &= \frac{1}{2}\left(\frac{\partial u_i}{\partial x_j} + \frac{\partial u_j}{\partial x_i}\right) \\ \epsilon_{vol} &= \epsilon_1 + \epsilon_2 + \epsilon_3 = \epsilon_{11} + \epsilon_{22} + \epsilon_{33} \\ \epsilon_q &= \sqrt{\frac{2}{9}[(\epsilon_{11} - \epsilon_{22})^2 + (\epsilon_{22} - \epsilon_{33})^2 + (\epsilon_{33} - \epsilon_{11})^2 + \frac{1}{3}(\gamma_{12}^2 + \gamma_{13}^2 + \gamma_{21}^2)]} \\ &= \frac{2}{3}(\epsilon_1 - \epsilon_3) \quad \text{when} \quad \epsilon_2 = \epsilon_3 \\ q_\epsilon &= \sqrt{3J_2^\epsilon} = \sqrt{\frac{1}{2}[(\epsilon_1 - \epsilon_2)^2 + (\epsilon_2 - \epsilon_3)^2 + (\epsilon_3 - \epsilon_1)^2]}\end{aligned} \tag{2.3}$$

2.3 Linear and Non-Linear Elasticity

All materials can for sufficiently small deformations be modeled as linear elastic due to first-order approximation. This requires that the energy is stored elastically and can be released upon unloading. It also requires that there are no permanent deformations, i.e. no change in the particle structure.

Since both the strain and the stress tensor is a rank two tensor, this implies that the stiffness tensor (elasticity tensor in this case) must be a rank four tensor. By imposing the major and minor symmetry in the elastic tensor one can reduce the amount of independent entries down to 21. Further reduction requires imposing symmetry planes. A material where all planes are symmetry planes are said to be isotropic elastic and only needs to 2 parameters to be uniquely defined (Nordal, 2014).

Some materials like rubber and soil exhibits nonlinear elastic response for a larger range of deformation. This means that the stress-strain graph is curved and stress or strain dependent. One notable model is the Duncan-Chang (Brinkgreve et al., 2017). The stress-strain curve is described as a hyperbolic function which implies larger deformations as the stress goes towards a predefined value. Another easy way to obtain nonlinear elasticity is by simply multiplying the stiffness tensor with hydrostatic stress p. Examples of linear and nonlinear curves are shown in figure 2.5

Care must be taken when using nonlinear elastic formulations. If the elastic strain energy is not properly defined, one might risk a material which generates energy on a specific load path. Example of this is given by Nordal (2014).

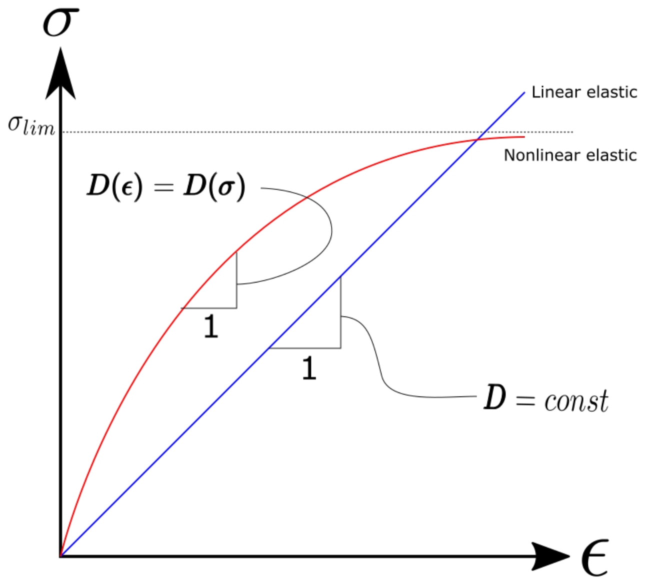


Figure 2.5: Linear and nonlinear elastic response.

2.4 Plasticity

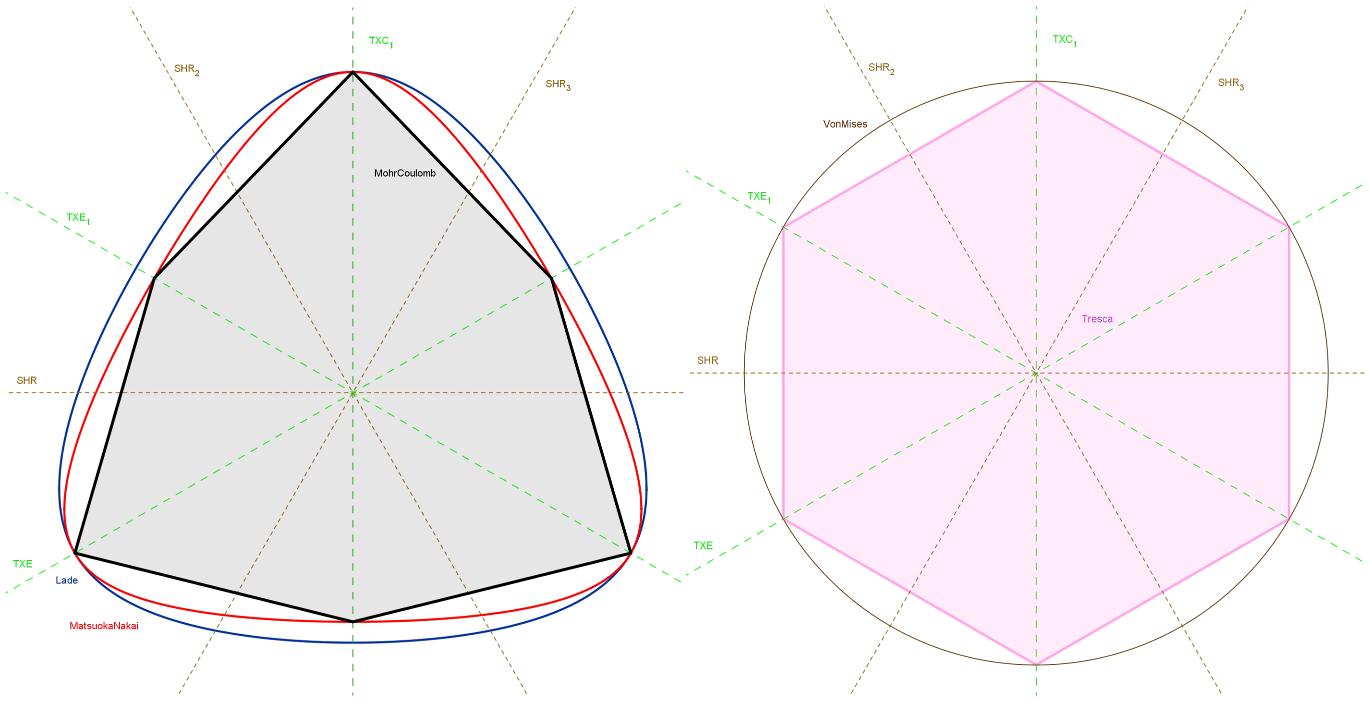
Yield surface

The yield surface marks the end of the elastic domain and the start of the plastic domain. At this particular point the material starts to plasticly deform and harden or soften according to the hardening rules. The yield surface has traditionally been given in terms of stresses since yielding seems to be stress dependent. The yield surface must encapsulate all admissible stress states and can mathematically be expressed as

$$F(\boldsymbol{\sigma}, \boldsymbol{\nu}) \leq 0 \tag{2.4}$$

where $\boldsymbol{\nu}$ are state variables influencing the yield surface. In the case of no hardening or softening there are no additional state variables $\boldsymbol{\nu}$.

Some common yield surfaces in soil modeling is shown in figure 2.6. The yield surfaces are all fitted to compression and are continuous and symmetric in the π-plane. The yield surfaces in (a) varies with the hydrostatic axis, while the surfaces in (b) are mean stress independent. The most well known yield functions in soil modeling are the Mohr-Coulomb and the Tresca criteria.



(a) Mohr-Coulomb (Inner), Matsuoka & Nakai (Middle), and Lade (Outer) criteria

(b) Von Mises (Outer) and Tresca (Inner) criteria

Figure 2.6: Yield surfaces fitted in compression shown in the π-plane.

During plastic flow the stress state must either stay on the yield surface or end up inside. The condition staying on the yield surface during plastic flow is called the consistency condition and can be written as

$$dF = \left\{\frac{\partial F(\boldsymbol{\sigma},\boldsymbol{\nu})}{\partial \boldsymbol{\sigma}}\right\} \cdot d\boldsymbol{\sigma} + \left\{\frac{\partial F(\boldsymbol{\sigma},\boldsymbol{\nu})}{\partial \boldsymbol{\nu}}\right\} \cdot d\boldsymbol{\nu}. \tag{2.5}$$

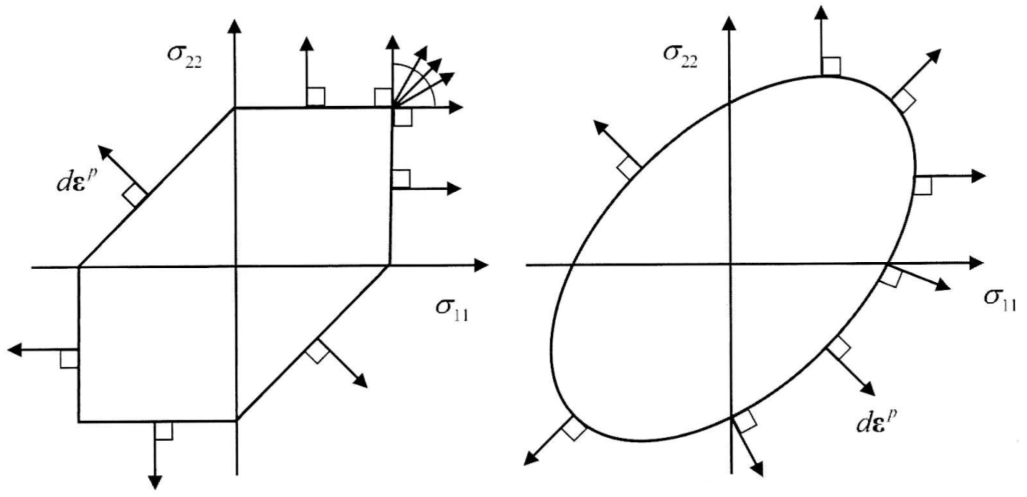
Plastic potential

The plastic potential gives the mathematical development of plastic strains during plastic flow. The equation is now formally known as the flow rule and can be written as

$$d\boldsymbol{\epsilon}^p = d\lambda \left\{ \frac{\partial Q}{\partial \boldsymbol{\sigma}} \right\} \tag{2.6}$$

where $d\lambda$ is a plastic multiplier factor and Q being the plastic potential. In the case where $Q = F$ the flow rule is said to be associated. It's generally accepted that granular materials are in fact non-associated

It's advantageous for both the yield surface and the plastic potential that the chosen function is sufficiently smooth such that the gradient is well defined. In the case where there are corner points the stress and especially the plastic increment can be singular if extra precaution is not taken. Illustration of the issue is shown in the figure 2.7.



(a) Tresca (b) Von Mises

Figure 2.7: Gradient on Tresca and Von-Mises viewed in σ_{11} - σ_{33} space. From Hopperstand and Børvik (2015)

Hardening

In the case where state variables are present, hardening rules must be defined to make the plastic development unique. Although the word hardening implies that a material gets stronger with increasing plasticity, a material may also soften as in the case of highly sensitive clays. The hardening rules combined together with the consistency condition makes the set of solutions unique, and a stress or strain increment can be solved numerically if a solution exists.

The easiest way to model hardening is to assume isotropic change in the yield surface. In the π-plane the yield surface will be scaled equal in all directions seen from the hydrostatic axis. Some materials exhibits higher strength for one stress path, and lower in an other after plastic flow. This is called kinematic hardening or sometimes the Bauschinger effect. This effect is achieved by translating the yield surface according to the hardening rules.

The hardening rules are commonly formulated based on dissipated energy, plastic strains or as mixed combination. In the case of dissipated energy both the stress and plastic strains are used to calculate the hardening evolution. Materials with limited volumetric deformations are often assumed to harden due to plastic distortion. In these cases the deviatoric strain can be used to determine the hardening. An alternative is to use volumetric hardening. It's also possible to create material where some state variables are coupled to plastic distortion, and other state variables are coupled with volume change.

2.5 Undrained Anisotropic Elastic Effect

Often materials are simplified as being isotropic, which in turn can have a huge impact on the stress path and consequently the ultimate failure capacity. Janbu proposed the dilatancy equation 2.7 shown below.

$$\Delta u = \Delta P + D \cdot \Delta q \tag{2.7}$$

The idea is based on that water cannot carry shear stresses. The change in pore pressure, either suction or pressure, is caused by a change in the isotropic mean stress and the deviatoric stress. For an undrained isotropic elasto-plastic material the volume and shear deformation is decoupled (Nordal, 2014). This implies a change in the total mean stress will not influence the soil particles due to constant volume. Shearing can however induce a pore pressure change, implying the soil particles have been rearranged.

In a p-q plot as shown in figure 2.8 the dilatancy is given directly by the inclination. Going straight up is an elastic response for an isotropic material. Any deviation means that plastic deformation has occurred. Meanwhile for an anisotropic elastic formulation the volume and shear deformations is not necessarily decoupled, meaning that a nonzero dilatancy may give no plastic deformation. The implications of the dilatancy is therefor very depended on the assumption of isotropic or anisotropic elasticity.

In the case of cyclic loading the material will be loaded, unloaded and reloaded. If the material degradation is thought of as an accumulation of plastic deformation, then the formulation of the elastic stiffness can have a large influence on the modeling of degradation.

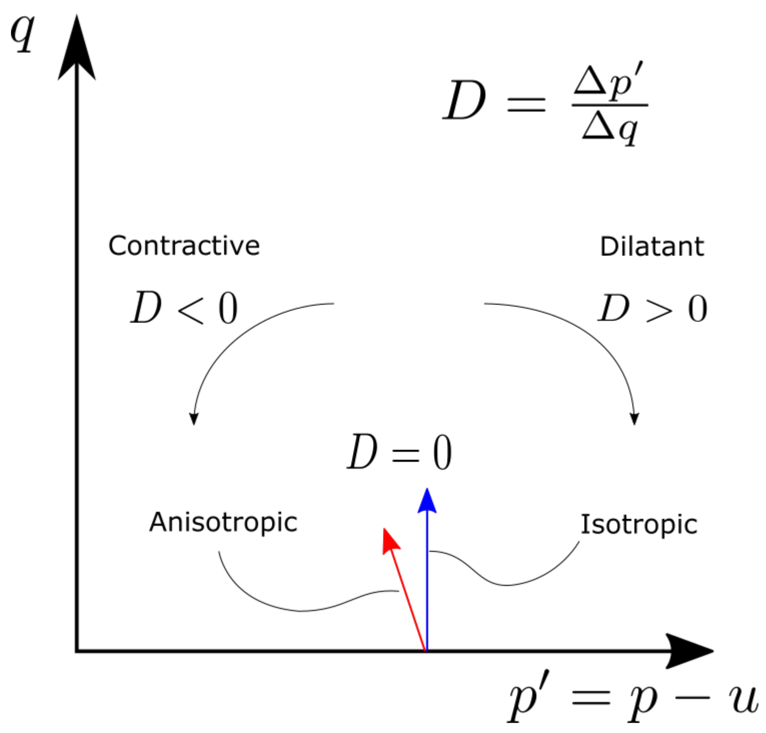


Figure 2.8: Initial dilatation shown for isotropic and anisotropic elastic material.

2.6 Stress Update & Numerical Methods

For any stress or strain increment a new state must be calculated. This is normally done by numerically solving differential equations in each GP. In the case of Ordinary Differential Equations (ODE) there are several numerical methods that can be utilized. Depending on the ODE one may use explicit or implicit schemes. The effectiveness of each numerical approach is highly dependent on the stiffness (numerical stability) of the ODE. While both approaches can be used to solve an ODE, the precision and calculation time can be very different. In general the explicit methods requires small increments such that the answers does not drift away from the real solutions. While the implicit methods will always follow the real solutions, the precision may be way off depending on the increment size.

The simplest explicit method is the Forward Euler, while more advanced family of methods like the Runge-Kutta are also commonly used. A simple analogy is that the Forward Euler runs towards the ball while the latter methods predicts where the ball will be. Even though the step increment is given by the FE program, it's possible to subdivide the step and increase the precision. Some methods can even be used to estimate step error like the Runge-Kutta-Fehlberg (Kreyszig, 2011).

The most well known implicit method is the Backward Euler. Instead of using the known values to calculate for the next position, an equation is set up with the unknown on both sides. This implies that a set of equations must be solved which further increases the computing time. In the case of multiple variables a Newton-Raphson (NR) methods is typically used. The NR methods requires calculating the gradient (Grimstad and Benz, 2016), also called the Jacobian matrix, which can be straight forward. The methods also require inverting the matrix, which can be difficult if not impossible for ill-conditioned matrices.

More numerical methods for calculating elastic and plastic increments can be found in *Concepts and Application of Finite Element Analysis* by Cook et al. (2002).

Chapter 3 Cyclic Behavior & Degradation

3.1 Load Characteristics

For any real physical construction the exterior loads must be transferred to the ground. A skyscraper, a bridge and even a tree must distribute the forces throughout the soil. In many cases the load from wind and self-weight can be viewed as static due to a long load period. In other cases the cyclic load phenomena cannot be neglected, especially regarding offshore constructions. The irregular loading from cars driving on a road and waves hitting a construction will momentarily cause a change in the pore pressure and plastic deformations may also occur.

The cyclic forces on the soil will wear and tear the soil fabric. If the water is not allowed to dissipate then a pore pressure will build-up and the soil particles will lose their friction against one another. This again causes the soil response to be less stiff and reduces the ultimate strength capacity. On the other-hand if water is allowed dissipate, the soil can become stiffer and stronger due to compaction and consolidation.

The framework presented here has been developed under the assumptions of offshore conditions, meaning relatively long load periods. Ocean waves has a duration around 10 to 20 seconds, while a storm can last for many hours (Andersen, 2015). Under this assumption the inertia of the soil can be neglected, meaning the problem can be solved as a quasi-static equilibrium.

3.2 Undrained Cyclic Behavior

The forces around foundations and piles can be quite complex. Depending on the load and soil-structure interaction the stress state can be hard to determine. Although one ideally would want to test a material for all sorts of stress combinations, there are physical limitations with doing so. As previously stated in chapter 2, a stress state can be represented by 6 independent entries. The commonly used triaxial test only has three degrees of freedom, and cannot show a rotation of principal axes. Meanwhile a Direct Simples Shear (DSS) test can induce rotation of principal axes, but the stress state is not explicitly defined and there are some question regarding the stress uniformity (Wood, 1991). A figure of the different stress states beneath a Gravity Based Structure (GBS) is shown in 3.1.

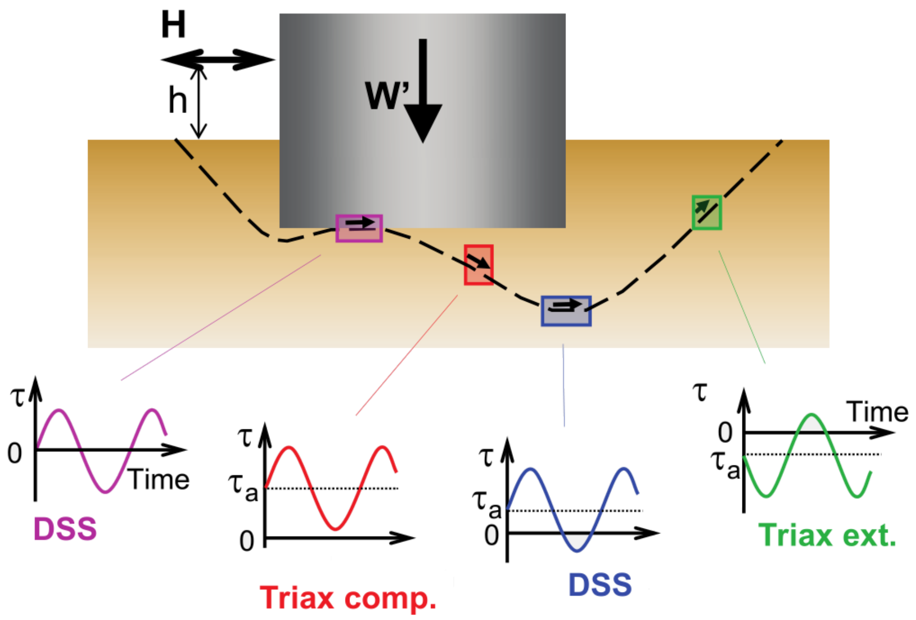


Figure 3.1: Simplified illustration of stress situation beneath a gravity bases structure. From Andersen (2015)

Although one cannot test a material for any conceivable load situation, many cyclic tests has been performed with triaxial and DSS apparatus. These tests lays the foundation for cyclic soil analysis which has been used in design calculations. Soil samples has been subjected to different conditions. Examples are changes in loading, cyclic frequency and also deformation. While metal is said to be mean shear stress independent (EC3), soil is the opposite. This means the traditional S-N curves used in fatigue calculations cannot be used for soil.

In both apparatus strains, stresses and pore pressure are logged over time. Since soil is mean shear stress dependent NGI has proposed a method to organize the cyclic data from the laboratory tests. A half cycle is determined when the shear stress (or strain for strain controlled tests) returns to the same initial value. In each cycle the shear stress is divded into an average and a cyclic part. The same is done for strains. Note that the permanent strains does not necessarily need to be the same as the average strain due to hysteresis as shown in figure 3.2.

Different cyclic test setups gives different results as can be seen in figure 3.3. Apart from the initial plasticity developed in DSS (Andersen, 2015), the response is fairly symmetric. On the contrary the triaxial test does not develop a symmetric response. One explanation is anisotropy. For a given initial stress state, an anisotropic material loaded in one particular direction may cause more plastic deformation than going the other way. There might be a certain stress state that can cause symmetric response, but the situation may also be instable altogether which causes the response to drift off as is indicated in figure 3.3.

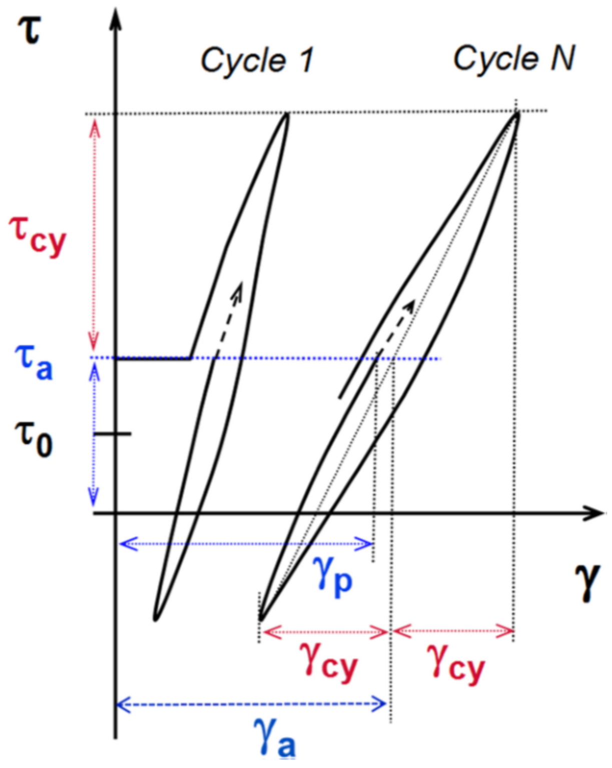


Figure 3.2: Illustration of stress-strain behavior after N cycles.
From Andersen (2015)

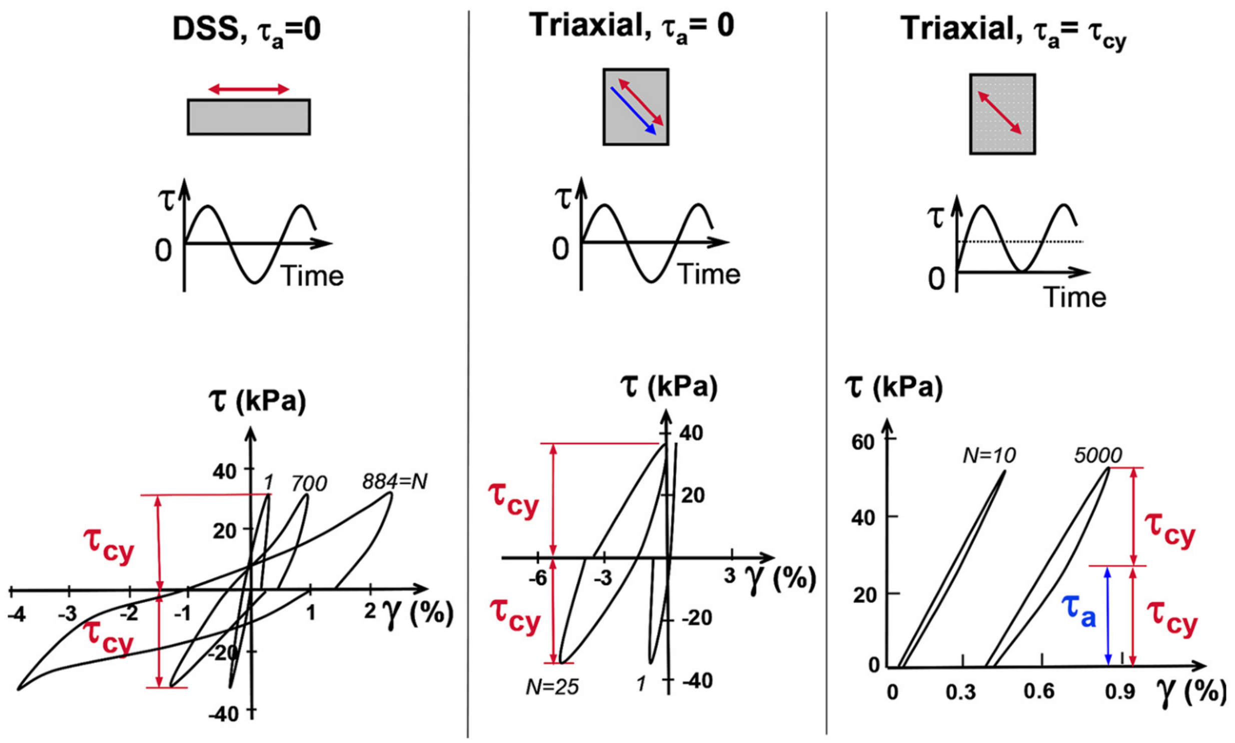


Figure 3.3: Various cyclic load conditions performed on Drammen Clay.
From Andersen (2015)

Though often being modeled as an elasto-plastic material, undrained soil does in fact exhibit permanent deformation for small stresses upon reloading. In figure 3.4 the green points represents one cycle for a low cyclic loading. For small changes in stresses there is generated a small permanent pore pressure (Andersen, 1976). For repeated cycles the pore pressure will accumulate and eventually the soil will either fail or reach a stabilization. In the latter case the soil has gotten used to the cyclic loading and a shakedown is reached (Juspi, 2007). The pore pressure change is then negligible. Increasing the stress will continue the process of pore pressure accumulation. For larger cyclic shear stresses the material will fail faster due to higher generated pore pressure and hitting the failure line earlier.

For a test sample subjected to harmonic loading it can be seen that the pore pressure and strains varies with time. In figure 3.5 the pore pressure and strains has been split up into an average and a cyclic part. It's clear from the figure that the harmonic loading causes an accumulation of pore pressure and strains. While the development of strains and pore pressure certainly are interesting, it's the permanent trend that is vital in design calculations.

The pore-pressure accumulation may even be more distinct for dense sand. Often the dilatancy in dense sand is quite strong, causing more plastic deformations in the material when undrained (Jostad et al., 2015).

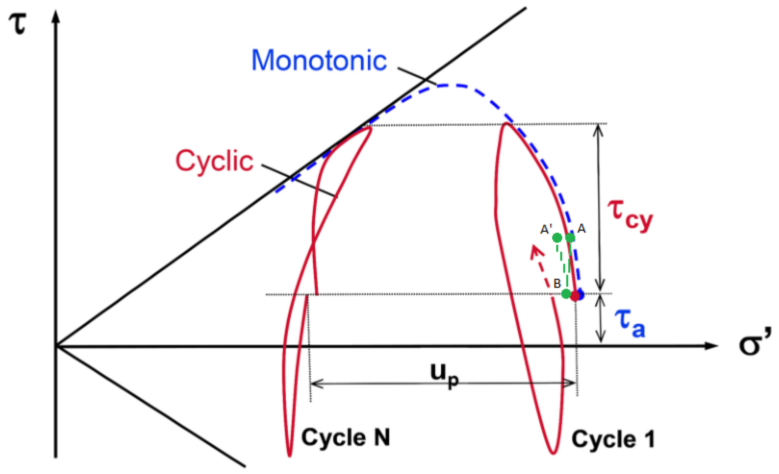


Figure 3.4: Montonic and Cyclic failure of soil tests.
From Andersen (2015)

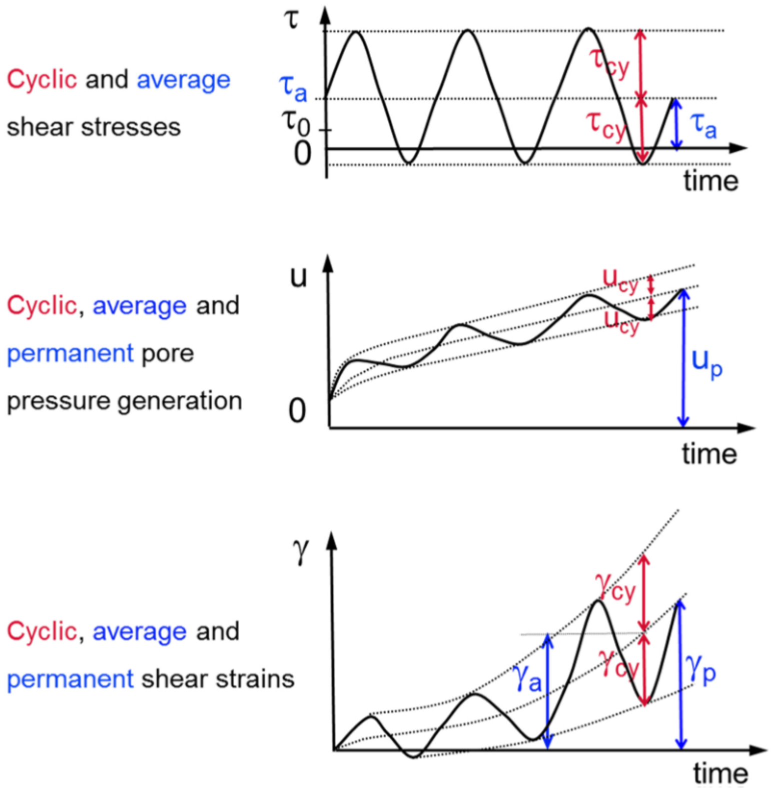
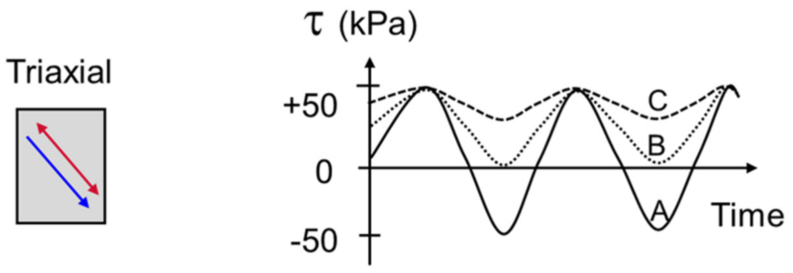
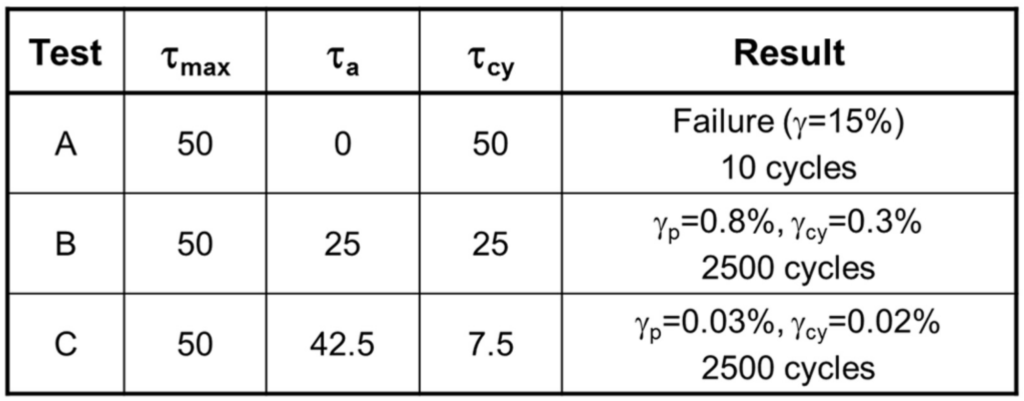

Cyclic and average shear stresses
τ
τa
τ0
0
τcy
τcy
τa
time
Cyclic, average and permanent pore pressure generation
u
0
ucy
ucy
up
time
Cyclic, average and permanent shear strains
γ
γcy
γa
γcy
γp
time

Figure 3.5: Accumulation of pore pressure and shear strains due to harmonic loading.
From Andersen (2015)

Tests has shown that it's not the maximum shear stress that is vital, but rather the actual size of cyclic shear stress. In figure 3.6 three samples with the same cyclic frequency and maximum shear stress has been tested under different stress conditions. It's clear that a higher cyclic shear stress causes more degradation in the material. One should also notice that the results shown here are from a triaxial test, meaning the average shear stress may influence the stress-strain pattern.

It's here assumed that cycling the material will primarily cause soil deterioration and thereby plastic deformation. The rearrangement of the soil particles causes a build-up of pore pressure on average. Higher cyclic shear stress gives more degradation while lower shear stress gives less degradation.





Test	τ_{max}	τ_a	τ_{cy}	Result
A	50	0	50	Failure (γ=15%) 10 cycles
B	50	25	25	γ_p=0.8%, γ_{cy}=0.3% 2500 cycles
C	50	42.5	7.5	γ_p=0.03%, γ_{cy}=0.02% 2500 cycles

Figure 3.6: Triaxial results from Drammen clay subjected to the same maximum shear stress. From Andersen (2015)

3.3 Development of Cyclic Contour Diagrams

For the development of cyclic contour diagrams, a lot of tests are needed. The static-monotonic failure tests serves as a reference line for the contours. For each cycle a dot can be determined in the diagram as shown in figure 3.7. The position of the dot is given by the average and the cyclic shear stress, which may be normalized. In addition the dot is also given an average and a cyclic shear strain. From all the available dots one can try to estimate which lines that corresponds to a constant average strain. The same is also done for the cyclic shear strain. This creates a grid-like figure as shown in figure 3.8 for a given cycle.

The same procedure can be repeated for each cycle. Thus the axis coming out of the plane is the cycle number. Seen in a 3D-space the full soil behavior can be observed. The same can also be applied to pore pressure.

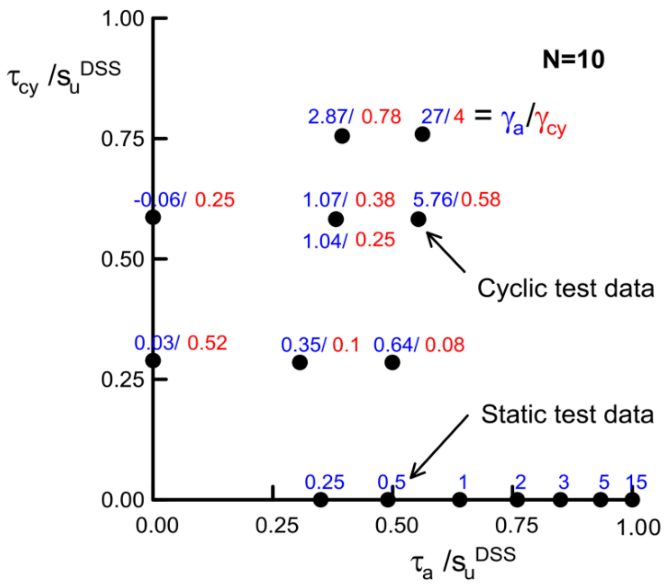
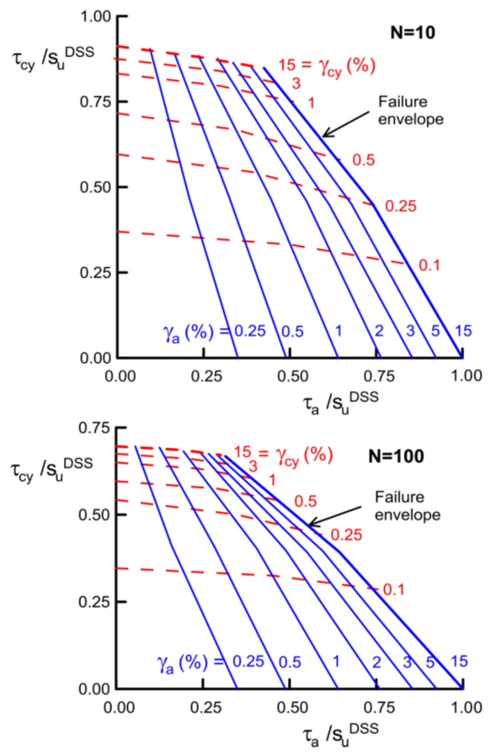


Figure 3.7: Contour diagram of DSS tests. Each dot represents one DSS test. From Andersen (2015)

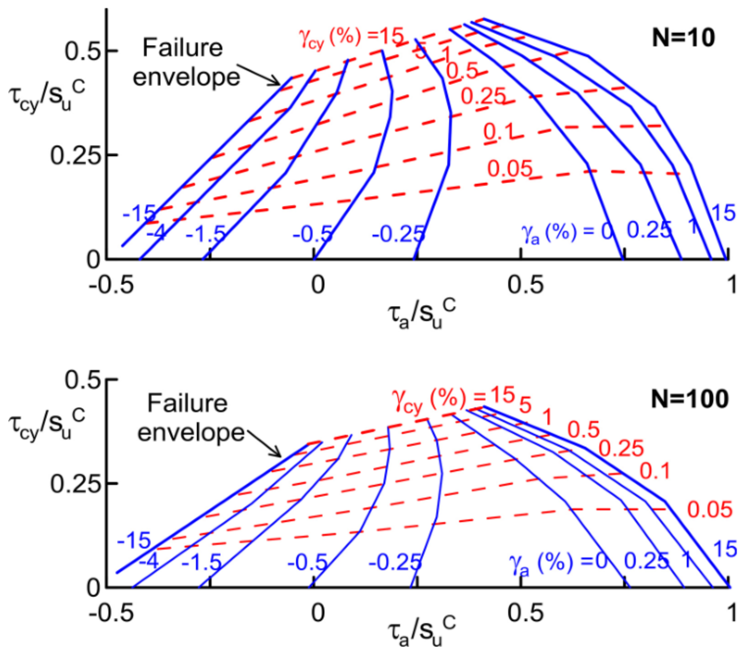
The cyclic contour diagrams reveals a lot of information. Firstly it can be seen from the triaxial tests that the contours are effected by the strength anisotropy. Secondly it can be seen that moving vertically up in the diagram means that both the average and the cyclic shear strain must change. Thirdly the outskirts of the diagrams represents plastic failure. At these lines the strains are deemed too large to be acceptable.

Ideally a soil model should be able to replicate the contours. However achieving a perfect fit with the DSS contours and the triaxial contours may prove difficult. And even though if a model could simulate a perfect match, a stress situation which is neither DSS nor triaxial might be entirely wrong.

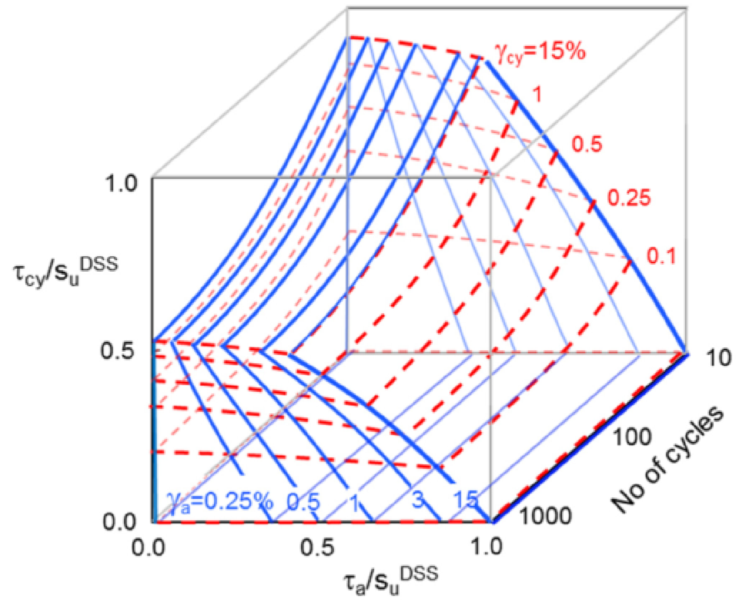
Depending on the soil modeling approach there are two viable options. One way is to try to create one or two coupled soil models to describe the cyclic contours. This means a fit is made for triaxial or DSS, and afterwards the other one. Ideally this should give sensible results in a boundary value problem and be the most accurate. A potential issue with this approach is that the soil models might not be generalized enough or they might be too little restrictive. Alternatively one can try to model the main features observed in the contours. The verification will then be exclusively based on the boundary value simulations. Even though the cyclic contours don't agree entirely, it may all-in-all give a good sensible solution



(a) DSS



(b) Triaxial



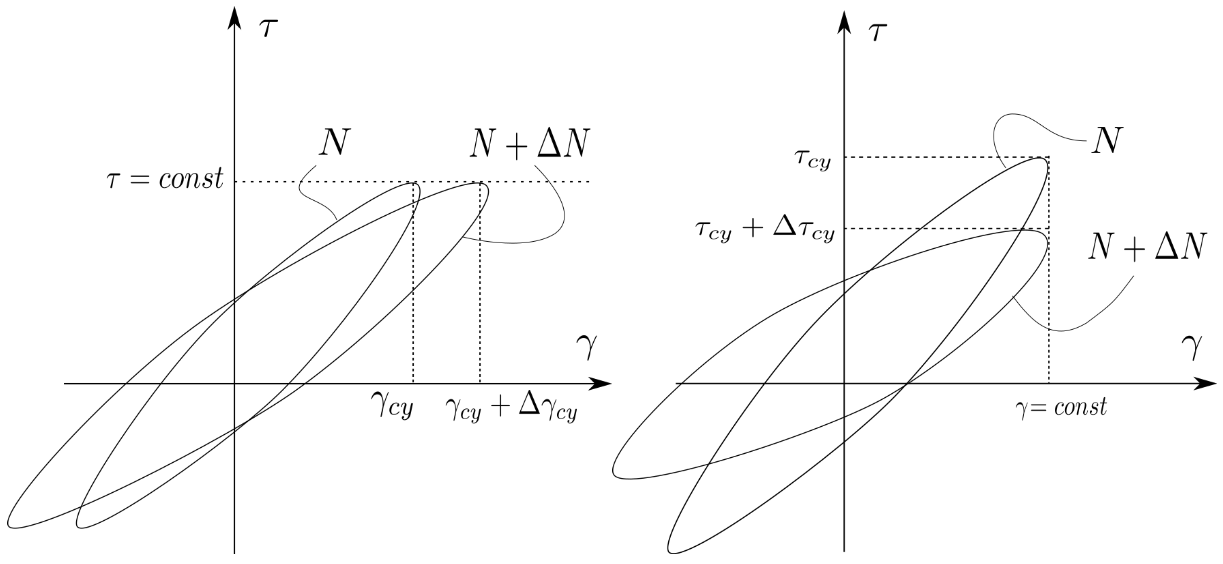
(c) 3D-DSS

Figure 3.8: Construction of cyclic contours diagrams.
From Andersen (2015)

3.4 Stress vs Strain controlled testing

The cyclic tests can either be done stress or strain controlled. Each setup will give different cyclic contour diagrams. The evolution of stress or strains will also be entirely different as one would expect. Irregardless of the test setup, the material will still degrade. The main difference is the degradation rate.

In figure 3.9 a comparison of the two methods are shown. In the stress controlled setup the shear stress is constant while the shear strains increases for each cycle. For the strain controlled test the shear strains are constant, but now the shear stresses reduces with each cycle. By comparing the energy density one can see that in the stress controlled setup the energy increases with each cycle. The strain controlled test will however have less energy stored in each cycle. The energy density comparison is shown mathematically in equation 3.1 and 3.2.



(a) Stress controlled (b) Strain controlled

Figure 3.9: Stress-strain curves for cycle N and cycle $N + \Delta N$

$$\int_{\gamma} \tau_{cy} \cdot d\gamma_{cy,N} < \int_{\gamma} \tau_{cy} \cdot d\gamma_{cy,N+\Delta N}, \quad \text{Stress Controlled} \tag{3.1}$$

$$\int_{\gamma} \tau_{cy} \cdot d\gamma_{cy,N} > \int_{\gamma} \tau_{cy} \cdot d\gamma_{cy,N+\Delta N}, \quad \text{Strain Controlled} \tag{3.2}$$

If the degradation is a function from the energy density, then the evolution of degradation will consequently be entirely different. This is shown in figure 3.10. It shows that the strain contours are steeper for higher stresses when strain-controlled. This is caused from having different stress-cycles paths. For lower constant strains the two methods give more similar results.

One question is whether or not the actual degradation around a structure is stress or strain controlled. NGI has claimed that stress-controlled is the best representation when exterior load is present (Andersen, 2015). The load can then be directly correlated to the stresses. It has also been suggested by Cai (2015) that due to redistribution of stresses the situation may actually be closer to being strain controlled.

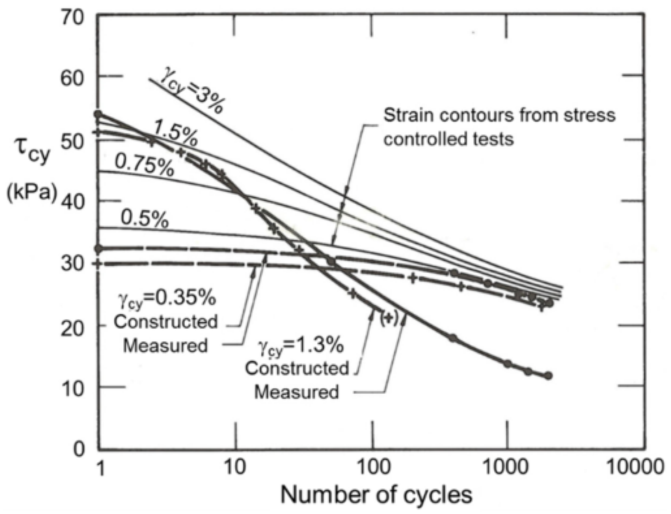


Figure 3.10: Cyclic contour diagram constructed from stress & strain controlled tests. From Andersen (2015)

3.5 Design Calculations

The principle idea for cyclic design calculations are that the accumulation of degradation can be expressed in an equivalent amount of cycles. A low cyclic shear stress with N cycles should give the same effect as for a higher cyclic shear stress with fewer cycles called N_{eq}.

The method is best shown with an example. In figure 3.11 the pore pressure cyclic contour has been determined for a soil material. Assuming the average shear stress is zero one gets the cross-section as shown. The load is by some method divided into constant cyclic load parcels, meaning there are certain predefined shear stress levels with N amount of cycles in each parcel. Assuming the material is completely undrained and no correction are made to the pore pressure, cycling from A to B will create an excess pore pressure in the soil for a given load. If one now were to increase the load, then the excess pore pressure would have to stay the same. In other words the contour line is followed up to the next stress level, denoted as C in the figure. This implies going from A to B should give the exact same effect as going from E to C. The next load parcel would start at point C and then move to point D. The pore pressure has now increased due to accumulation. Afterwards one follows the contour lines up to the next stress level and the process is repeated until the last load parcel or failure has been reached.

For the strain accumulation principle the same methodology is adopted. Instead of using the pore pressure, cyclic strain contours are used. Each load parcel causes an accumulation of cyclic shear strain. Since a change in just the cyclic shear stress causes both average and cyclic strains, a correction is made with the accumulation. Andersen (1976) proposed that the stress-strain curve from the first cycle could be used. The change in strain followed by the stress jump is simply added to the accumulation for $N = 1$. So instead of ending up in point C, extra strain is added such that the new point is now G. It's also stated that this method has been successful in predicting permanent strains for varying cyclic stress.

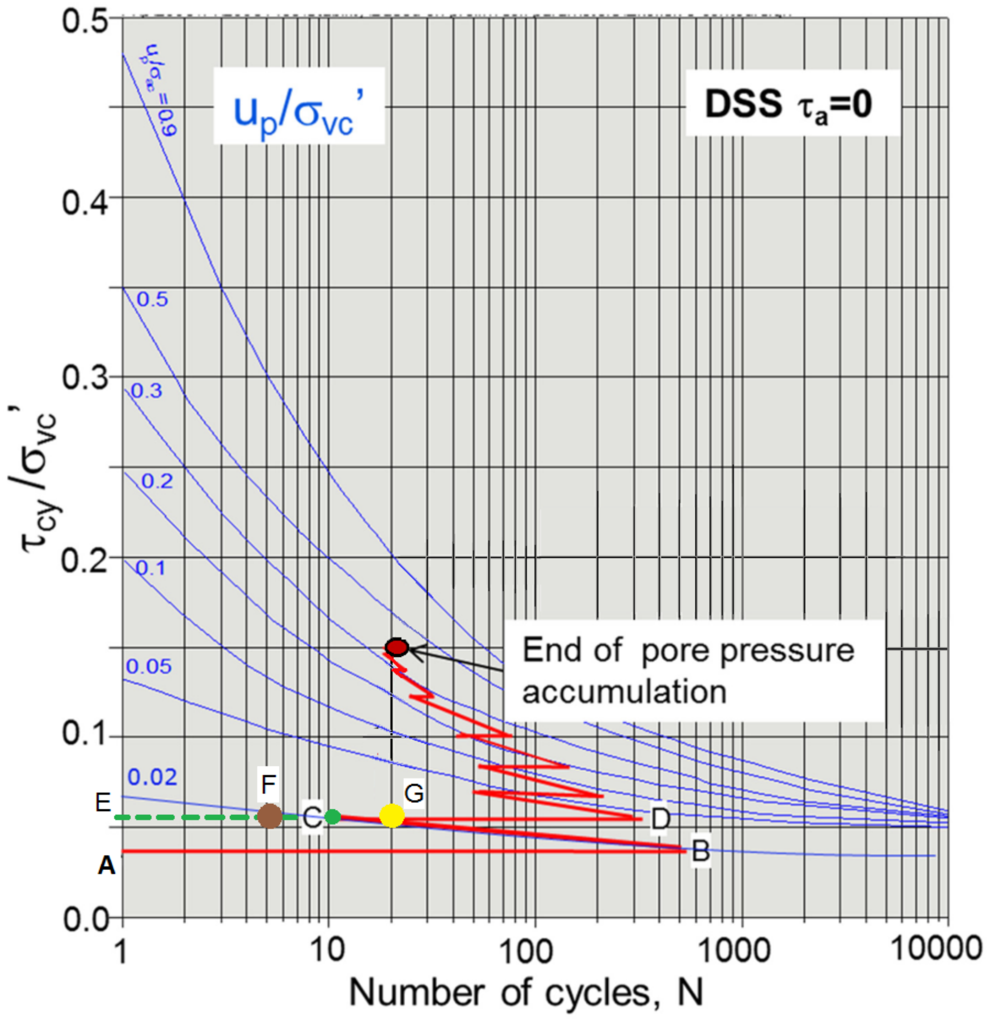


Figure 3.11: Principle of equivalent degradation accumulation.
From Andersen (2015)

In the case of partly drained conditions the soil will dissipate water during the cyclic loading. In this case the model is tweaked by assuming that some of the water is dissipated. So instead of ending up in point C one now ends up in point F. The effects from cyclic degradation are now less since the pore pressure is lower. By assuming that each load parcel is done undrained the same procedure as described above is calculated and thereafter the correction is done. Alternatively dissipation can occur meanwhile a load parcel is going on. In this way the correction is done prior to following the contour up to the next stress level.

In triaxial contours the pore pressure is normally also corrected. It's stated that only permanent pore pressure are of interest. Therefor the pore pressure is corrected by subtracting the total octahedral normal stress, $\Delta\sigma_{oct}$. This is supposed to make the diagram independent of the average shear stress (Andersen, 2015).

Although these two methods have commonly been used in design calculations, they might not be good methods for calculating the soil degradation. Instead one might imagine that the cyclic contours are derived from the degradation process. Assuming that the main principle holds true, it should be able to draw contours based on the change in degradation.

3.6 FE analysis with cyclic contour diagrams

Based on the cyclic contour diagrams, a FE method called UDCAM has been proposed by Jostad et al. (2014). Prior to doing the calculations the cyclic contours must first be determined. Later points on the contour diagram are digitized as matrices and stored in a file. This is done for a few selected cycles. The files along with some input parameters are used in the subsequent calculations Grimstad et al. (2012). The degradation is calculated based on the strain accumulation principle. The dots are linear interpolated, and the emphasize on triaxial or DSS is determined from a ratio of the vertical and the second invariant of the deviatoric strain. The ratio is then used to determine the shear stress based on an elliptical interpolation of the DSS and triaxial stress state.

The procedure described above has been verified by several models and is commonly used by NGI in offshore structure design. The main drawback with this method is that the calculation procedure is labour-intensive. It also requires the establishment of cyclic contour diagrams, which in reality requires a lot of test samples (although correlations exists). In addition the method gives no insight in the material behavior since the cyclic effects are predetermined.

Jostad et al. (2015) has also proposed a method for partially drained soils called PDCAM. Unlike the other model, the average material is effective stress based. In this way pore pressure dissipation may occur. The model also requires the cyclic contour diagrams for pore pressure given for some selected cycles and additional parameters. The model share similarities to the Mobilised Friction Model and the high cyclic accumulation model for sand (Jostad et al., 2015).

Chapter 4 Average Model

4.1 Purpose of Average Model

The average model can for all intents and purposes also be regarded as a static model. The model can be time dependent (i.e. viscoplastic) or simply elasto-plastic. The main idea is that the average model has a state variable which accounts for the soil structure. This variable can be thought of as a health bar for the material and is initially 100%. When the material is subjected to cyclic loading the material will lose its structure and will ultimately be complete remoulded with 0% health left. Although the material is completely remoulded there will always be an intrinsic strength capacity (Burland, 1990) regardless of the actual initial structure. Another approach could be to view the structure parameter as a measure of resistance. A high value will give a lower plastic response. As the structure approaches zero the plastic response can increase.

With the average soil model one may calculate settlements, consolidation, ultimate bearing capacity and etc. When subjected to cyclic loading the exterior load is thought to be divided into an average part as shown in figure 4.1 and a cyclic part. Though the average and cyclic calculations are done in different phases in a FE program, the two situations actually occur simultaneously. It's therefor necessary to iterate between these two models.

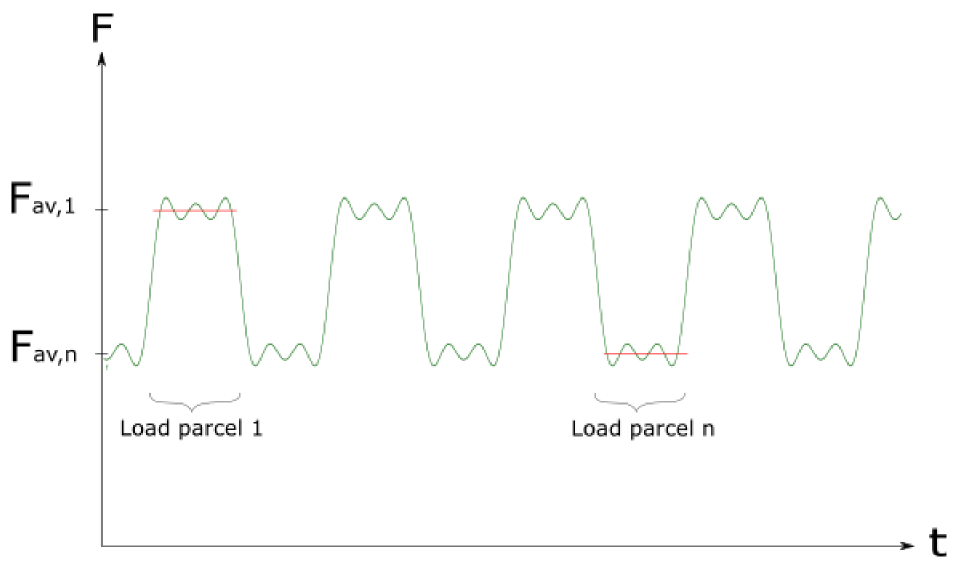


Figure 4.1: Average load determined from a periodic load.

In certain materials like for an instance sand, drainage can happen during cycling. If the soil model is an effective stress model then consolidation can be calculated for the corresponding amount of cycles by assuming a constant angular velocity for the load.

4.2 Structure in Average model

Soil out in the nature has been deposited over time and subjected to weathering processes, creep due to self-weight and maybe even cementation because of chemical processes. It's believed that the soil particles will get a special orientation during deposition (Burland, 1990). The arrangement of the soil particles and the anisotropic stress state caused by the self-weight will in general make the material response anisotropic. This effect can be achieved in soil models by introducing a rotational tensor, sometimes called a fabric tensor. By describing the yield surface in terms of stresses and the fabric tensor one can create rotated yield surface as shown in figure 4.2 (a).

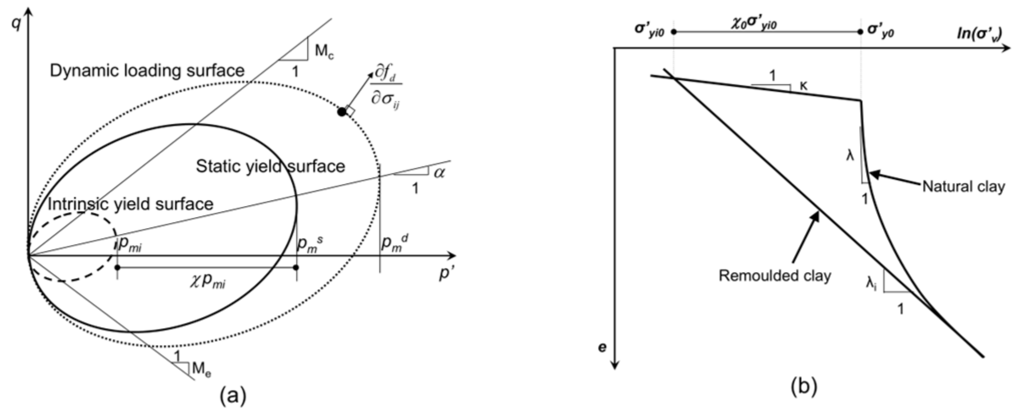


Figure 4.2: Illustration of degradation in p-q space and isotropic compression. The yield surface is initially the one denoted static and shrinks towards the intrinsic surface as plasticity increases.
From Yin and Karstunen (2008)

Even though a fabric tensor is introduced, the soil model cannot simulate the softening that is observed in soil tests. To achieve this effect a hardening rule is introduced to cause degradation in the material. The soil is given an initial structure, χ_0, and will during plastic flow decay to zero. This causes the initial yield surface to shrink towards an intrinsic yield surface as shown in the figure above. If the material is now unloaded and reloaded the response will be much softer due to remolding as shown in figure 4.2 (b). The simplest form of modeling destructuration can be achieved by using the following equations

$$F = F(\boldsymbol{\sigma}, p'_m) \tag{4.1}$$

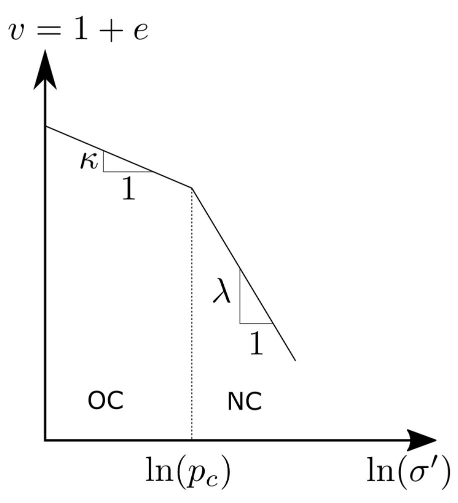
$$p'_m = (1+\chi)\,p'_{mi} \tag{4.2}$$

$$\frac{d\chi}{d\lambda} = -\chi \cdot \left(a \left\| \frac{\partial Q}{\partial p'} \right\| + b \left\| \frac{\partial Q}{\partial q} \right\| \right) \tag{4.3}$$

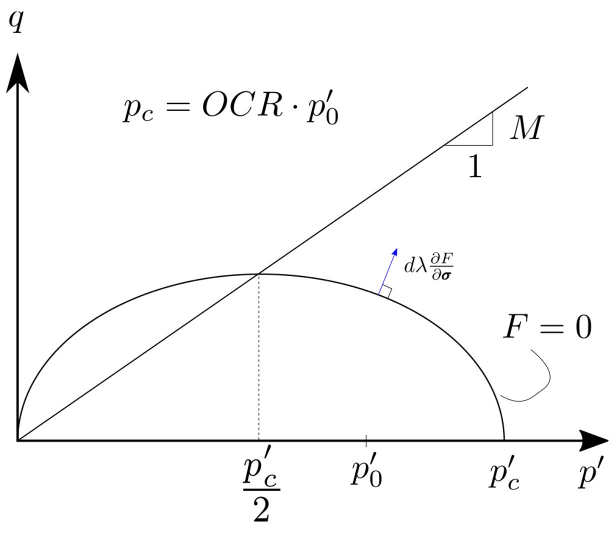
where a and b are fitting parameters (Rønningen et al., 2014). Note that the structure and fabric tensor is independent of each other.

4.3 Modified Cam-Clay with Destructuration

The Modified Cam Clay proposed by Roscoe and Burland (1968) is a model based on the concept of Critical State Soil Mechanics. A critical state is reached when there are no changes in the effective stress or in volume for large shear strains. In contrast to many other material models it uses volumetric hardening. The yield surface is an ellipsoid in the principal space and is determined from the Critical State Line (CSL) and the preconsolidation stress. The inclination of the CSL is calculated from the residual (intrinsic) Mohr-Coulomb friction angle, while the preconsolidation stress is normally given indirectly as the Over-Consolidation Ratio, OCR. Key parameters are the compression index λ and the swelling index κ. These values are determined from triaxial isotropic compression or indirectly from an oedometer test. For simplicity the model is assumed to be associated, meaning the plastic potential is the same as the yield surface shown in figure 4.3.



(a) Isotropic compression.



(b) Yield surface in p-q plane.

Figure 4.3: Graphical formulation of Modified Cam Clay.

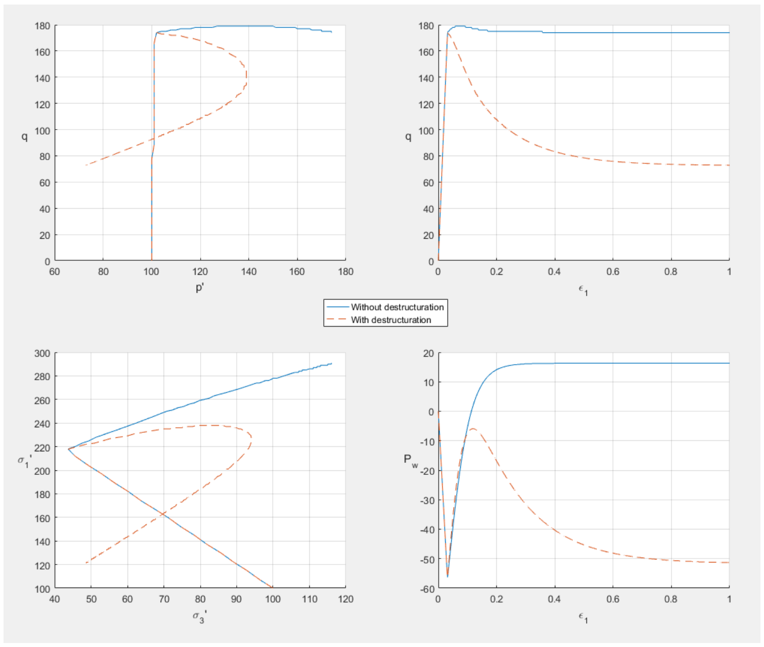
Even though it's possible to simulate softening with the Modified Cam Clay model, the effects are not in the same scale as observed in laboratory tests. To capture the wanted effects it's necessary to include destructuration in the soil model. By utilizing the equation listed in subsection 4.2, the Modfied Cam Clay can model more distinct softening. First off the yield surface is rewritten as

$$F = q^2 - M^2 p'(p'_{mi}(1+\chi) - p'). \tag{4.4}$$

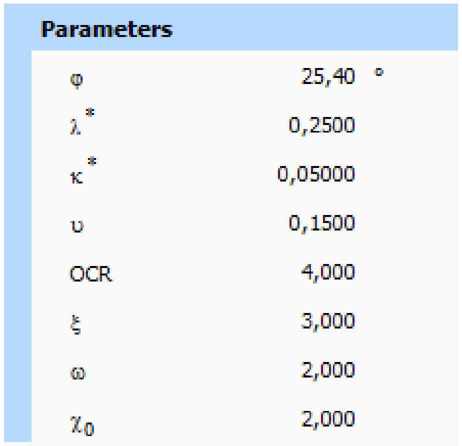
Secondly the degradation rule must be defined. Here the same equation as proposed in the Geofuture soft clay model will be used, which is

$$\frac{d\chi}{d\lambda} = -\chi \cdot \xi \cdot \sqrt{\left(\frac{\partial Q}{\partial p'}\right)^2 + \omega^2 \cdot \frac{2}{3}\left(\frac{\partial Q}{\partial q}\right)^2} \tag{4.5}$$

where ξ controls the absolute destructuration rate and ω tells how much deviatoric shear influences the destructuration rate. The effects of including destructuration in the Cam Clay model is clearly shown in figure 4.4.



(a) Dashed orange curves are with destructuration.



Parameters

φ	25,40	°
λ^*	0,2500	
κ^*	0,05000	
υ	0,1500	
OCR	4,000	
ξ	3,000	
ω	2,000	
χ_0	2,000	

(b) Material input for destructuration.

Figure 4.4: Comparison of Modified Cam Clay with and without destructuration.

4.4 Geofuture Soft Clay Model

It has long been known that the soil behavior is controlled by the effective stresses. Many effective stress models has over the years been proposed, some with greater success than others. The complexity of soil mechanics makes it difficult to create a model that's completely accurate and easy to use. Anisotropy, non-linearity, creep, destructuration, load hysteresis, drainage and high small strain stiffness are all common features observed in soil tests. The simplest models can be great at describing a special feature of interest. While more complex soil behavior requires more advanced models, often at the cost of many fitting parameters. For the working engineer it's better when the input parameters are easily obtained and has a clear causality in the model.

It's the goal of Geofuture Soft Clay Model to be capable of simulating natural soft Scandinavian clay. Even though the model is effective stress based, the input parameters is of the total stress type. From field measurements and laboratory tests one can determine key parameters that can easily be interpreted and afterward used directly in the model. Based on these input parameters, fitting variables are determined for active and passive undrained triaxial tests by the model. Hence the parameter for structure is given indirectly by specifying the undrained compression shear strength at 20% strain, $S_{20\%}^{TXC}$.

The model features isotropic mean stress dependent elasticity, anisotropic yield[1] surface of the Lade type criteria as shown in figure 2.6 (a) and can be associated or non-associated. The yield and potential surface may rotate during

[1] Since the model is visco-plastic there are actually no yield surfaces, only reference surfaces. It's here simplified to be the same although they´re not.

plastic flow and will reach a steady state for large shear strains. Destructuration and hardening of the intrinsic yield surface are also included. Another key feature is volumetric creep of the Janbu type formulation. This makes the model visco-plastic since it's time-dependent. For more in-depth description one is referred to the user manual for Geofuture soil model (Rønningen, 2017).

The destructuration is in this model coupled together with rotation of the yield surface. As previously mentioned the yield surface and the plastic potential will rotate during plastic flow until a steady state is reached. The rotational hardening function for the yield surface is written as

$$\frac{d\boldsymbol{\beta}_d}{d\lambda} = \frac{\mu}{1+\chi}\frac{p}{p_{mi}}(\boldsymbol{\beta}_{d,t} - \boldsymbol{\beta}_d) \tag{4.6}$$

where μ is the absolute rotation rate, $\boldsymbol{\beta}_d$ being the rotational tensor for the yield surface and $\boldsymbol{\beta}_{d,t}$ the rotational tensor at steady state. During plastic flow χ will decrease towards zero while p_{mi} will increase according to the hardening rules. A large structure χ implies that there will be a smaller change in rotation and vice versa for a low value. Often followed by a rotation is more plastic strains.

It's here proposed to use a cyclic degradation material model coupled with the Geofuture soil model. Together it might be possible to simulate the cyclic degradation explicitly. The structure χ, and the average stresses if needed, is passed to the cyclic material model. Due to some cyclic degradation rule in the cyclic model, the structure will be reduced and will cause a change in the average model. After a few iterations both models should converge to a steady-state, but now with some plastic deformation. In that way the Geofuture Soft Clay model would be extremely versatile since it could also be used in situation where cyclic loading also is important.

Chapter 5 Cyclic Model

5.1 Purpose of Cyclic Model

The cyclic soil model is primarily meant to deteriorate the structure from the average model. By cycling the material the soil particles will be shaken and moved around, which causes loss in structure. Depending on the average model formulation the loss of structure may decrease the yield surface, make the yield surface more susceptible to rotation or perhaps even both.

It's here assumed that an irregular load can be transformed into load parcels. In each parcel the cyclic load is considered constant and acts for a given amount of cycles. The transformation can be visualized in figure 5.1. In the cyclic model only the cyclic load amplitude is of interest. In reality the load varies back and forth and thus also the soil. Therefor the overall response in the cyclic model should be symmetric or antisymmetric depending on what is measured.

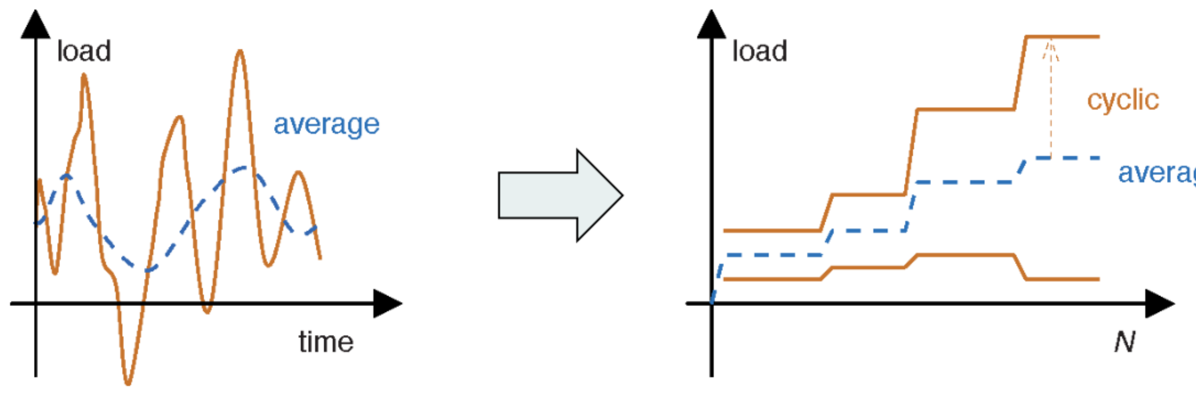


Figure 5.1: Transformation of transient load to load parcels.
From RISUEÑO et al. (2013)

How to create an explicit cyclic degradation model is currently uncharted territory. Here one is trying to take a shortcut by explaining the major permanent effects caused by cycling. It's therefor proposed here to start with simple models. An important feature in the model is the degradation formulation. As previously stated the load can be simplified as a constant that acts over a duration of time. The longer the duration, the more soil degradation occurs. The soil should also be weightless, so that the bearing capacity is only taken care of by the average part. It's also reasonable to assume undrained conditions during the cyclic phase due to the short time frame.

The easiest cyclic models are either formulated as entirely elastic or elasto-plastic. The important issue is how the degradation is calculated. That being said, the model should soften with increasing strains. In that way the stress is distributed downward in the soil as expected (Matlock, 1970). The simplest solution is using just a non-linear elastic model. Then the stresses and strains can be used to define the degradation. By introducing plasticity, new variables are introduced and can be used in degradation calculations.

5.2 Mathematical Description of Cyclic Contours

To mathematically describe the cyclic contour diagrams correctly it is, in the authors opinion, necessary to create a Partial Differential Equation (PDE). Given the complexity of formulating and solving a PDE it's much easier to try solving for one plane at a time. If one assumes that the average and cyclic stress plane is independent of the amount of cycles N, then the problem can be simplified and formulated as

$$f(\tau_a, \tau_{cy}, N) = g(\tau_{cy}, \tau_a) \cdot h(\tau_{cy}, N). \tag{5.1}$$

In this case the average and cyclic stress plane, $g(\tau_{cy}, \tau_a)$ (figure 3.8 (a) & (b)) is simply scaled down with the increasing amount of cycles. The cyclic stress and cycles plane $h(\tau_{cy}, N)$ (figure 3.11) will have to be a function that decreases with the amount of cycles. Note that a point in the cyclic stress and cycles plane can be explicitly defined by two of three parameters τ_{cy}, N and γ_{cy}, while the average and cyclic stress plane can be defined by two out of four parameters τ_{cy}, τ_a, γ_{cy} and γ_a.

Looking at the the cyclic stress and cycles plane, $h(\tau_{cy}, N)$, the contours will vary whether the pore pressure or cyclic strain is drawn. They will also be material dependent. The contours can be mathematically described by establishing an Ordinary Differential Equation (ODE) on the form

$$\frac{d\tau_{cy}}{dN} = h(\tau_{cy}, N). \tag{5.2}$$

Solving the ODE gives a *family of solutions* (Kreyszig, 2011). One contour line is explicitly defined when a boundary conditions is given. This can for an instance be given by the cyclic stress-strain curve from the first cycle. One should also note that it's possible to create contours with a double curvature that approaches a steady state as shown in figure 5.2. For more complicated curvatures one might be forced to use a second order ODE. However this means two boundary conditions are needed which further complicates the problem.

Prior to cyclic loading there is an average stress and strain. Once the cycling commences there will also be a cyclic stress and strain. This implies an iteration between the two material models is needed to determine the location in the average and cyclic stress plane. If $h(\tau_{cy}, N)$ is determined for zero average stresses and assuming τ_{cy} is not defined by the average plane, a bold simplification can be done. By letting τ_a be predefined from the average model, then $g(\tau_{cy}, \tau_a) = g(\tau_a)$ can be viewed as a scaling function dependent on τ_a. In this way the anisotropic effect and the variation with τ_a can easily be included. Examples of $g(\tau_{cy}, \tau_a)$ limits are shown in figure 5.3. In both cases an ellipse has been scaled and rotated to look similar to the contours in figure 3.8. A more refined method would be to introduce strain-cutoff for $\gamma_a > 15\%$, making it possible to create functions more similar to the cyclic contours.

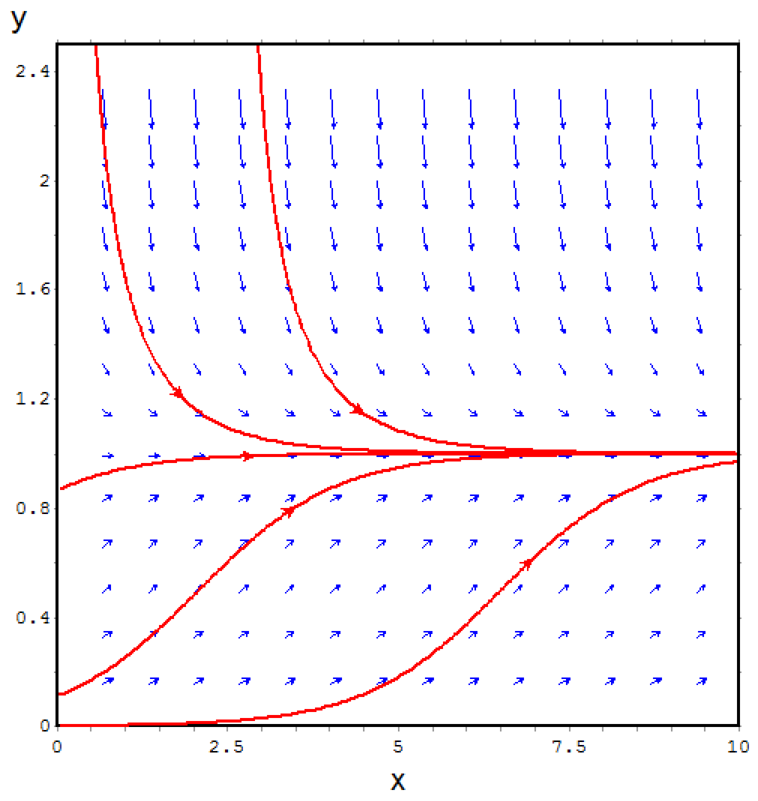
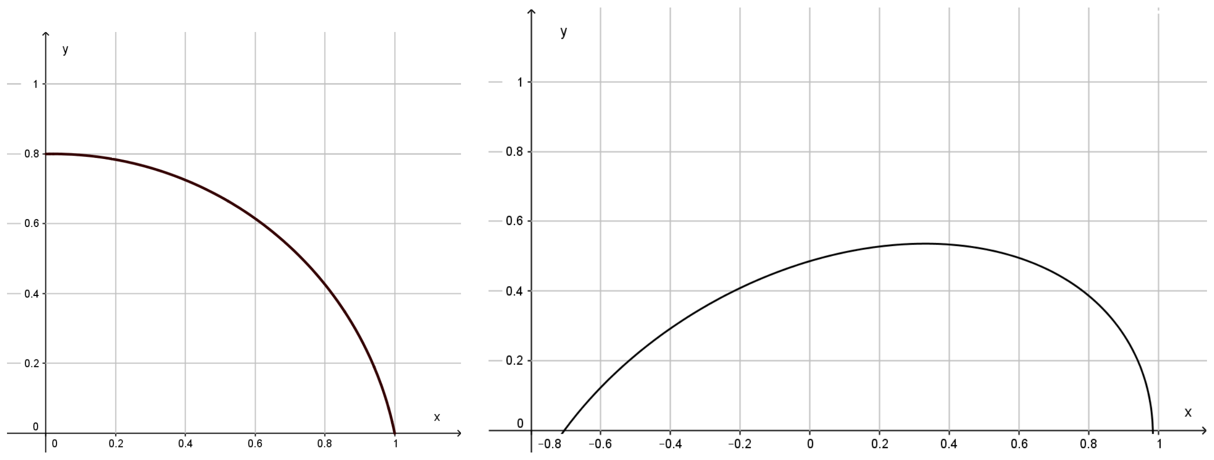


Figure 5.2: Direction plot of $y' = y - y^2$ with some level curves.



(a) DSS roof (b) Triaxial roof

Figure 5.3: Average-plane scaling functions.

5.3 Simple Cyclic Material Model

It's here proposed to use an isotropic non-linear elastic model of the Duncan-Chang formulation. A Von Mises yield surface has been included, although it should never be needed since the strains approaches infinite in the elastic region (i.e. no energy dissipation)[1]. The non-linear elasticity has been introduced by setting:

$$\begin{bmatrix} p' \\ q \end{bmatrix} = \begin{bmatrix} K & 0 \\ 0 & 3G \end{bmatrix} \cdot \begin{bmatrix} \epsilon_{vol} \\ \epsilon_q \end{bmatrix} \tag{5.3}$$

$$K \to \infty \tag{5.4}$$

$$G = \frac{E}{3} \cdot \left(1 - \frac{q}{2S_u}\right) \tag{5.5}$$

The stress-strain curve for this formulation is shown in figure 5.4 for $S_u = 50$ and $E = 200$. The material is initially stiff and soften as it approaches the specified threshold. This is similar to the response for a pile as the one used in $P - Y$ calculations (Matlock, 1970).

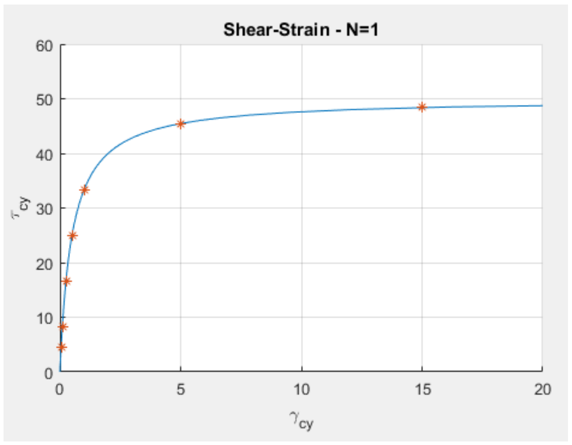


Figure 5.4: Cyclic stress-strain curve from the first cycle.

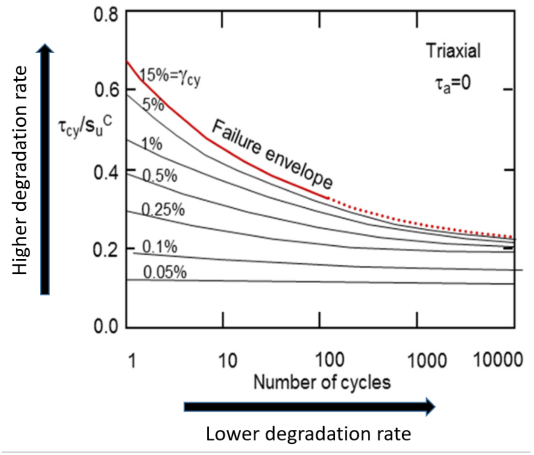
[1]In fact the yield surface is disabled in the Fortran code since it caused numerical issues when inverting the Jacobian matrix.

Based on the information given in section 3.2 & 3.4 it's here assumed that higher cyclic stresses causes more degradation. As the number of cycles increases the degradation rate reduces. This is illustrated in figure 5.5 (a). A similar contour has been expressed as an differential equation on the form

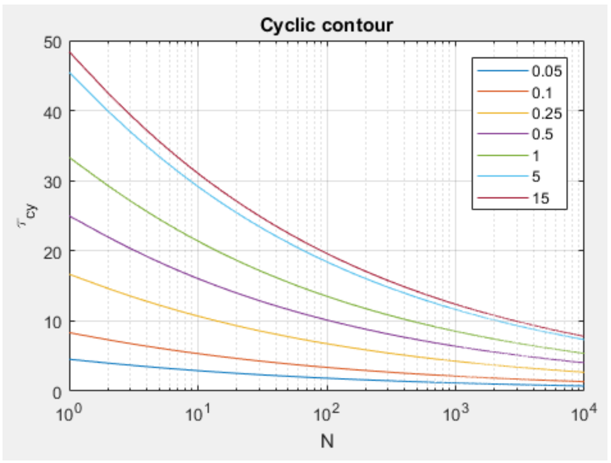
$$\frac{d\tau_{cy}}{dN} = \frac{-a\tau_{cy}}{b+cN} \tag{5.6}$$

and is shown in figure 5.5 (b) for some loosely fitted parameters. In this case the parameters has been set to $a = 2, b = 1$ and $c = 10$. The contours shown in the figure has been determined from the curve in figure 5.4. For an instance the 15% contour level has been determined by using the shear stress in the stress-strain curve (red asterisks in the figure) as a boundary condition in the differential equation. For simplicity the average contribution has been ignored, meaning

$$\begin{aligned} f(\tau_a, \tau_{cy}, N) &= g(\tau_{cy}, \tau_a) \cdot h(\tau_{cy}, N) \qquad (5.7) \\ &= 3\tau_{cy,N=1}^{0.2} \cdot (1+10N)^{-0.2}. \end{aligned}$$



(a) Degradation feature wanted.



(b) Fitted differential equation.

Figure 5.5: Degradation rate and contour development.

To capture the degradation feature as described above it has simply been stated here that the degradation is the same as the vertical component of the gradient of equation 5.7. This can be expressed as $d\chi = \alpha \cdot d\tau_{cy}$, where α is a fitting parameter. Increasing the stress implies higher degradation, and as N increases, the degradation rate decreases. The degradation accumulation of each load parcel is calculated as

$$\Delta\chi = \int_{N_{eq}}^{N_{eq}+\Delta N} d\chi = \int_{N_{eq}}^{N_{eq}+\Delta N} \frac{d\tau_{cy}(\tau_{cy,N=1}, N)}{dN} dN. \tag{5.8}$$

Based on the specified formulation some simple simulations has been carried out in Matlab. The first load is supposed to represent an idealized storm load. For calculating N_{eq} it's been assumed that one can simply follow the contour line up to the next stress level. Two tests of constant cyclic load has also been carried out. The material has been cycled with 5% and 80% of the given S_u. Lastly one test was meant to simulate strain-controlled cycling. Here the cyclic strain was set to constant $\gamma_{cy} = 15\%$.

The simple model is able to capture the wanted effects as shown in figure 5.6 for $\alpha = 0.3$. The degradation rate reduces as the amount of cycles N increases. Low cyclic shear stress causes negligible destructuration as is expected. Both the high cyclic shear stress and the strain controlled test causes a lot of degradation in a short amount of time, but the high stress test was stopped earlier due to large strains. The strain-controlled test can on the other-hand can go on for quite some time and will cause higher degree of degradation. The idealized storm also shows that each time the stress is increased, the degradation rate also increases.

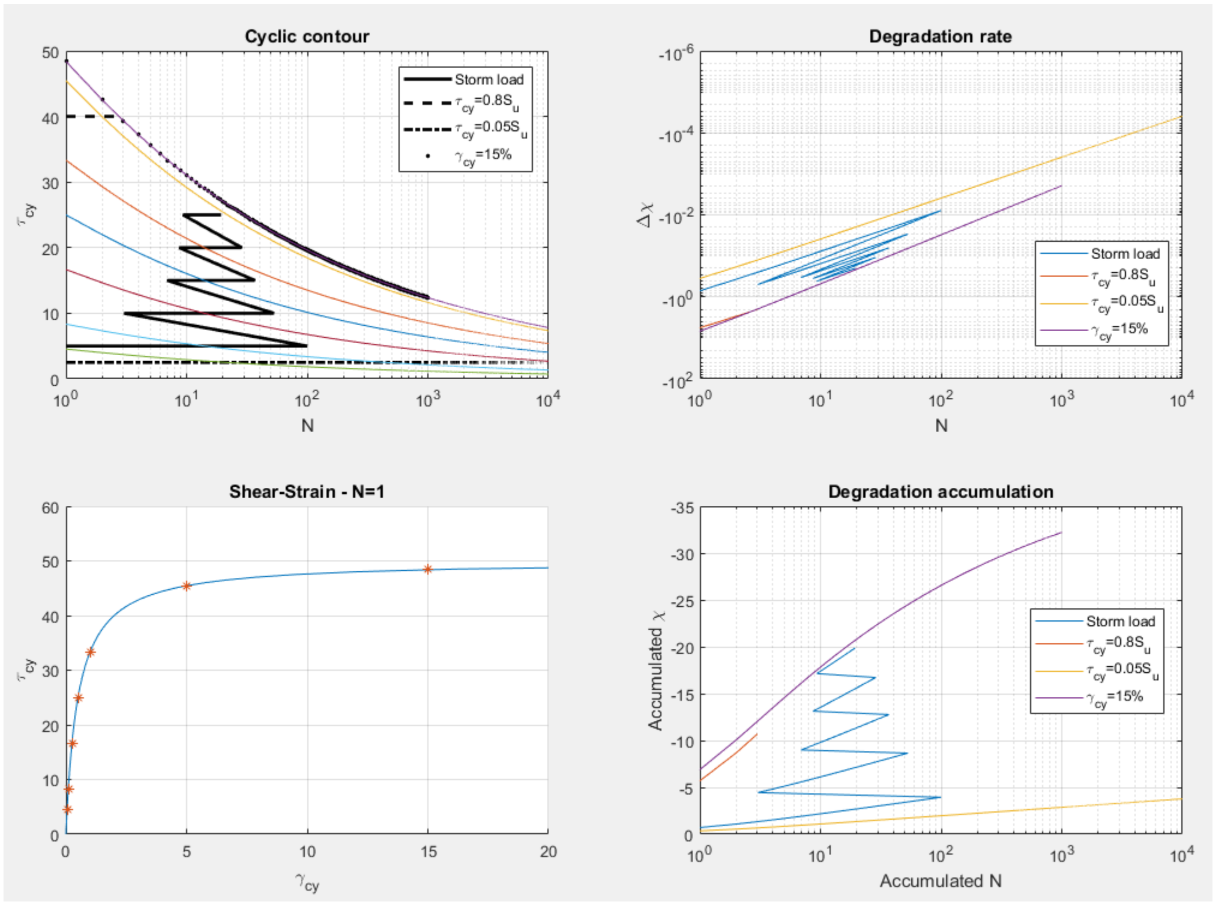
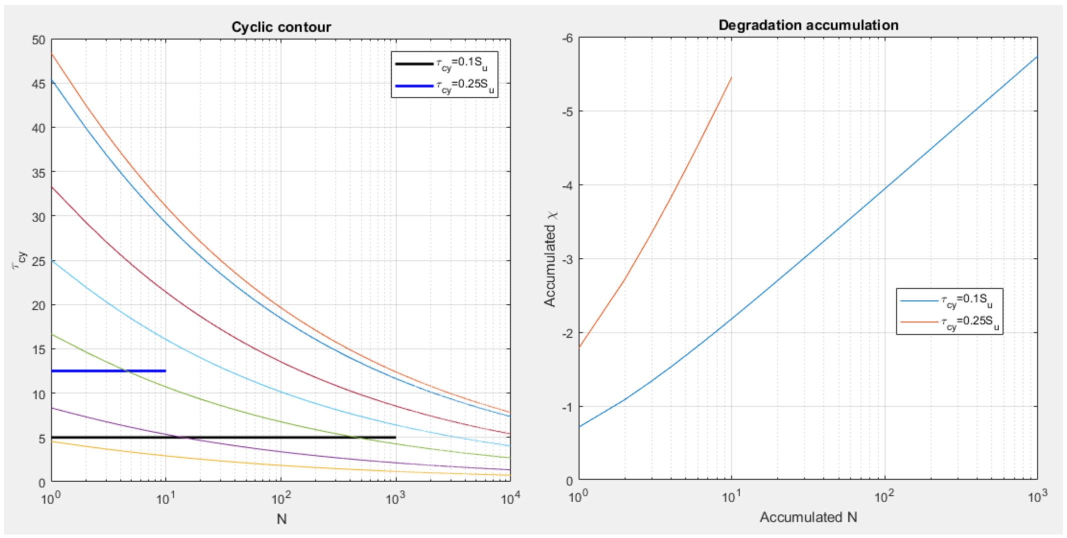


Figure 5.6: Accumulation of degradation for different cyclic loading.

The main principle idea that the degradation can be expressed for different stress levels with an equivalent amount of cycles, N_{eq}, requires some extra consideration. It's clear that following the contours up to the next stress level will not be possible if there's a large stress jump. Figure 5.7 shows that even for small stress changes, some error is introduced. A remedy is to calculate the equal amount of stress degradation as

$$\int_1^N d\chi\Big|_{\tau_{cy,0}} = \int_1^{N_{eq}} d\chi\Big|_{\tau_{cy,1}}. \tag{5.9}$$



(a) Stress paths

(b) Accumulation of degradation.

Figure 5.7: Different stress paths ending up at the same contour level does not give the same degradation.

Even though the equation in 5.9 can be used to calculate N_{eq}, there might still be some issues. Given $\Delta\chi$ is known after some cyclic loading, then calculating the N_{eq} that corresponds to the same degradation might be impossible at a low stress level. One example is first cycling at a high stress level and afterwards cycling at a low stress level. This may require $N_{eq} \rightarrow \infty$ if not an appropriate ODE is selected in the first place. Yet another simplification can be to introduce a cut-off. If $N_{eq} > N_{max}$ then the degradation can be assumed to be zero. This way one avoids the risk of numerical issues.

5.4 Further Thoughts on Degradation

If one assumes that equation 5.9 is true, then there should be an explicit way to express the degradation caused by cycling. In section 3.5 two methods proposed by Andersen (2015) is shown. Both methods uses the cyclic contour diagrams to calculate the accumulated degradation. One should also remember that both methods has also been verified with laboratory tests to a satisfying degree. It's therefor reasonable that a degradation equation can come from studying the cyclic contour diagrams.

An alternative approach might be to think of the problem as an energy issue. Since the material prior to cycling has an initial structure, the degradation at $N = 1$ must be zero. Irregardless of the cyclic history, a certain amount of degradation will occur, where higher penalty is given for higher stresses. For a potential field this means the degradation contours must be tightly packed for increasing τ_{cy} at $N = 1$. Alternatively the problem might be formulated as a non-conservative field, something that implies the degradation is in fact stress dependent.

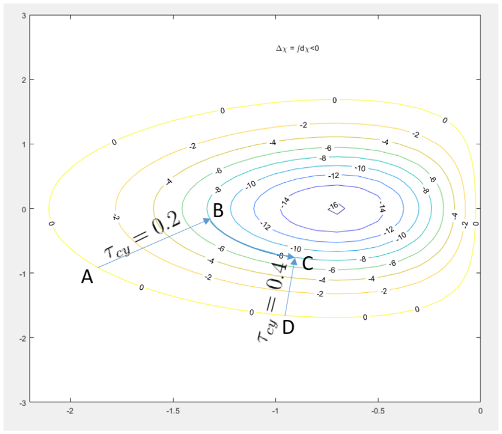


Figure 5.8: Thought experiment of degradation as a potential.

Chapter 6 Python Framework

6.1 What is Python and why Python?

When solving a problem, using the right tools makes all the difference. The same is true for programming. There are different levels of programming languages. The low-level languages focuses on the bits and bytes, and is more closely related to binary code than an actual spoken language. High-level languages on the other-hand has greater focus on being more self-explanatory for humans. Though this often comes at greater cost of language precision for computers. Python falls into the high-level category and is often used as an introduction to programming due to it's versatility and simplicity.

Python is an interpreted language, meaning that the machine code is compiled every time a script is run. The benefits are that the code can be written once and run on any platform so long as an compatible interpreter is installed. Due to Pythons popularity most platforms supports it. In bare essentials Python is free and there are many external libraries that also are free. Some libraries requires a paid license when used for commercial purposes. The language supports different programming styles like functional and object-oriented programming. It's also worthwhile mentioning that there is a large and growing community, and finding active forums & tutorials are readily available.

Some FE programs even has built-in Python support. Examples are Abaqus, Ansys, OpenFOAM and PLAXIS. While some projects will be better off without

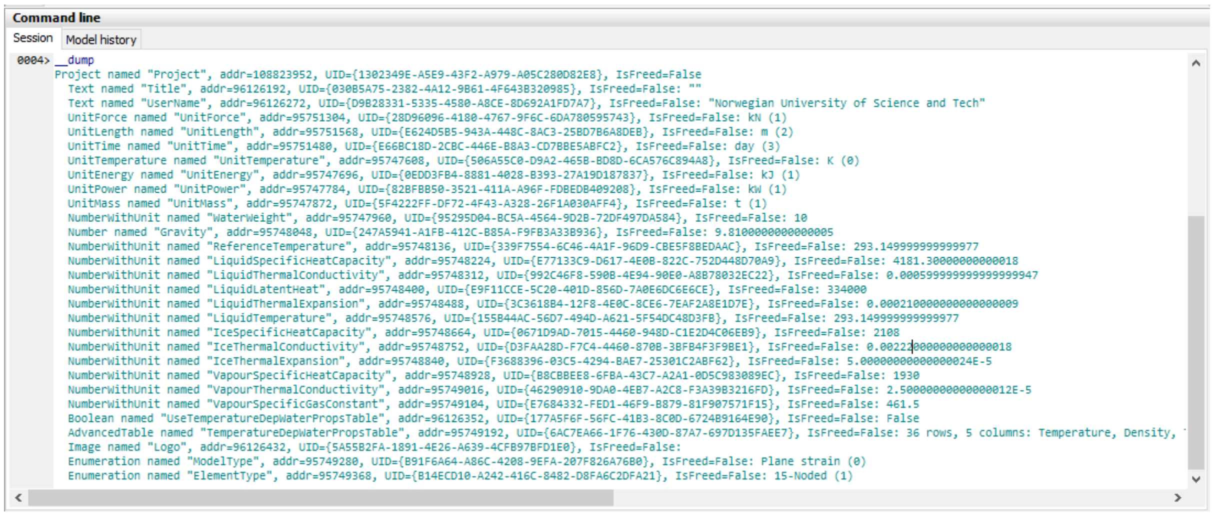
Python, others can benefit greatly from it. Repetitive, labor-intensive input to the FE model can in many cases be fully automated in a script, although it requires some extra consideration regarding the design of the model. It may also minimize unintended mistakes and speed up the whole process.

In the development of the Python framework the following libraries has been used: *PyQt4*, *os*, *subprocess*, *imp*, *ctypes*, *time*, *NumPy* and *matplotlib*. Note that *PyQt4* is based on the *QT* GUI framework and might require licenses for both depending on the distribution and commercial application.

6.2 Python and PLAXIS

In PLAXIS a Python wrapper has been created which allows for Python commands to be sent to PLAXIS. This feature is only available for VIP licenses. To establish a connection between Python and PLAXIS a server must first be opened and started in PLAXIS. This allows PLAXIS to be controlled by exterior programs. The information is sent via HTTP based API (Brinkgreve et al., 2017) so the server does not have to be on the same computer as the script. In the Python script a boilerplate code provided by PLAXIS must always be included. This code establishes a connection to the PLAXIS server and imports the necessary libraries. When the script is run, two variables are returned from the boilerplate code. The first variable represents the PLAXIS Input Application program and can be used to create new projects. The second variable represents the actual model in the project, also called the global object, and is the most used variable. The second variable will hereby be denoted as `plxModel`.

Once a connection is made one can control PLAXIS as a marionette puppet. To check which value or settings that is assigned one must query PLAXIS by either checking the actual value or by listing (dumping) all the available information related to the given object. The commands in Python are similar to the once shown in PLAXIS Input. Whenever something is done in the model, the corresponding PLAXIS command will be listed in the Command line. An example with the `__dump` command is shown below in figure 6.1



```
0004> __dump
Project named "Project", addr=108823952, UID={1302349E-A5E9-43F2-A979-A05C280D82E8}, IsFreed=False
  Text named "Title", addr=96126192, UID={030B5A75-2382-4A12-9B61-4F643B320985}, IsFreed=False: ""
  Text named "UserName", addr=96126272, UID={D9B28331-5335-4580-A8CE-8D692A1FD7A7}, IsFreed=False: "Norwegian University of Science and Tech"
  UnitForce named "UnitForce", addr=95751304, UID={28D96096-4180-4767-9F6C-6DA780595743}, IsFreed=False: kN (1)
  UnitLength named "UnitLength", addr=95751568, UID={E624D5B5-943A-448C-8AC3-25BD7B6A8DEB}, IsFreed=False: m (2)
  UnitTime named "UnitTime", addr=95751480, UID={E66BC18D-2CBC-446E-B8A3-CD7BBE5ABFC2}, IsFreed=False: day (3)
  UnitTemperature named "UnitTemperature", addr=95747608, UID={506A55C0-D9A2-465B-BD8D-6CA576C894A8}, IsFreed=False: K (0)
  UnitEnergy named "UnitEnergy", addr=95747696, UID={0EDD3FB4-8881-4028-B393-27A19D187837}, IsFreed=False: kJ (1)
  UnitPower named "UnitPower", addr=95747784, UID={82BFBB50-3521-411A-A96F-FDBEDB409208}, IsFreed=False: kW (1)
  UnitMass named "UnitMass", addr=95747872, UID={5F4222FF-DF72-4F43-A328-26F1A030AFF4}, IsFreed=False: t (1)
  NumberWithUnit named "WaterWeight", addr=95747960, UID={95295D04-BC5A-4564-9D2B-72DF497DA584}, IsFreed=False: 10
  Number named "Gravity", addr=95748048, UID={247A5941-A1FB-412C-B85A-F9FB3A33B936}, IsFreed=False: 9.8100000000000005
  NumberWithUnit named "ReferenceTemperature", addr=95748136, UID={339F7554-6C46-4A1F-96D9-CBE5F8BEDAAC}, IsFreed=False: 293.149999999999977
  NumberWithUnit named "LiquidSpecificHeatCapacity", addr=95748224, UID={E77133C9-D617-4E0B-822C-752D448D70A9}, IsFreed=False: 4181.30000000000018
  NumberWithUnit named "LiquidThermalConductivity", addr=95748312, UID={992C46F8-590B-4E94-90E0-A8B78032EC22}, IsFreed=False: 0.000599999999999999947
  NumberWithUnit named "LiquidLatentHeat", addr=95748400, UID={E9F11CCE-5C20-401D-856D-7A0E6DC6E6CE}, IsFreed=False: 334000
  NumberWithUnit named "LiquidThermalExpansion", addr=95748488, UID={3C3618B4-12F8-4E0C-8CE6-7EAF2A8E1D7E}, IsFreed=False: 0.000210000000000000009
  NumberWithUnit named "LiquidTemperature", addr=95748576, UID={155B44AC-56D7-494D-A621-5F54DC48D3FB}, IsFreed=False: 293.149999999999977
  NumberWithUnit named "IceSpecificHeatCapacity", addr=95748664, UID={0671D9AD-7015-4460-948D-C1E2D4C06EB9}, IsFreed=False: 2108
  NumberWithUnit named "IceThermalConductivity", addr=95748752, UID={D3FAA28D-F7C4-4460-870B-3BFB4F3F9BE1}, IsFreed=False: 0.00222000000000000018
  NumberWithUnit named "IceThermalExpansion", addr=95748840, UID={F3688396-03C5-4294-BAE7-25301C2ABF62}, IsFreed=False: 5.00000000000000024E-5
  NumberWithUnit named "VapourSpecificHeatCapacity", addr=95748928, UID={B8CBBEE8-6FBA-43C7-A2A1-0D5C983089EC}, IsFreed=False: 1930
  NumberWithUnit named "VapourThermalConductivity", addr=95749016, UID={46290910-9DA0-4EB7-A2C8-F3A39B3216FD}, IsFreed=False: 2.50000000000000012E-5
  NumberWithUnit named "VapourSpecificGasConstant", addr=95749104, UID={E7684332-FED1-46F9-B879-81F907571F15}, IsFreed=False: 461.5
  Boolean named "UseTemperatureDepWaterPropsTable", addr=96126352, UID={177A5F6F-56FC-41B3-8C0D-6724B9164E90}, IsFreed=False: False
  AdvancedTable named "TemperatureDepWaterPropsTable", addr=95749192, UID={6AC7EA66-1F76-430D-87A7-697D135FAEE7}, IsFreed=False: 36 rows, 5 columns: Temperature, Density,
  Image named "Logo", addr=96126432, UID={5A55B2FA-1891-4E26-A639-4CFB97BFD1E0}, IsFreed=False:
  Enumeration named "ModelType", addr=95749280, UID={B91F6A64-A86C-4208-9EFA-207F826A76B0}, IsFreed=False: Plane strain (0)
  Enumeration named "ElementType", addr=95749368, UID={B14ECD10-A242-416C-8482-D8FA6C2DFA21}, IsFreed=False: 15-Noded (1)
```

Figure 6.1: Command line in PLAXIS with the command `__dump` invoked.

Inserting a borehole at $x = 3$ and setting the groundwater-level to $y = 0$ returns two sets of commands. These commands and the Python equivalent is written below. Remember that `plxModel` is the second variable from the boilerplate code and contains information of the actual model in the project.

```
PLAXIS:         _borehole 3
PLAXIS:      _set borehole.Head 0


Python: plxModel.borehole(3)
Python: plxModel.set(plxModel.boreholes[0].Head, 0)
```

The variable `plxModel` which is a global object contains attributes, constructors and methods, as is often the case in object-oriented programming. Attributes are placeholders for information like text or numerical values. The constructors are used to create new objects from classes (blueprints), while the methods are used to interact with attributes (Horstmann, 2013).

The command `plxModel.borehole(*)` is a constructor that creates new borehole objects. Typing the same command creates new boreholes which is added to a list. The groundwater-level is an attribute stored in an unknown variable and cannot be change directly. By using the method `plxModel.set(*)` one can change the attributes in an object. Each object when created is given an ID which tells where in the computer memory it can be found. If a new variable is created when the object is constructed, then the variable can be used as a reference. Otherwise one may use the list where all the similar objects are contained. The method shown below gives the same result as above. Notice that `plxModel` is not necessary when the object location is known.

```
newVariable = plxModel.borehole(3)
plxModel.set(newVariable.Head, 0)
```

In PLAXIS all lines, loads, points, phases, soil, materials and etc. are considered objects. Creating objects and changing attributes are similar to the methods shown above. A particular phase object can be found in a list of phases `plxModel.phases[#value]` and a new phase is created after the referenced phase when the command is called. The new phase will be stored in the same list.

```
i = 2 # In Python this is the third item in the list!
newPhase = plxModel.phase(plxModel.phases[i])
newPhase == plxModel.phases[-1] # This is true
```

Some attributes are public, meaning they can be changed directly without invoking any methods. Though some attributes and methods requires that the correct mode is chosen.

```
plxModel.gotomesh() # Mesh mode must be selected prior to meshing
plxModel.mesh(0.06)

plxModel.gotostages()
loadPhase = plxModel.phase(plxModel.phases[-1])
loadPhase.Identification = "Generate stress"       # String
loadPhase.TimeInterval = 10                        # Value
loadPhase.Deform.ResetDisplacementsToZero = True   # Logical
```

There are two ways to access the attributes stored in an object. As previously shown it is possible to dump all the available information. The more subtle approach is to use the method `plxModel.echo()`. This returns a formatted text containing the information stored in the attribute.

```
plxModel.TimeInterval.dump()  # Returns a lot of information
plxModel.TimeInterval.echo()  # Returns one text (string) line
```

The easiest way to learn new commands is to simply do the action in PLAXIS Input and afterwards interpret it to Python code and test it. PLAXIS provides a Command Reference Manual that can be found in the tab *Help*. The commands is written for PLAXIS and must also be interpreted and tested in Python. Lastly it's also possible to query the objects, methods and attributes for information with the built-in functions in Python. Normally `Help(arg)` will be documentation of the selected argument (`arg`). It turns out that the Python documentation is a bit lacking. Another Python function is the `dir(arg)`. It returns a list of attributes and methods available in the object[1].

[1]Actually it returns a string with the scope of the argument.

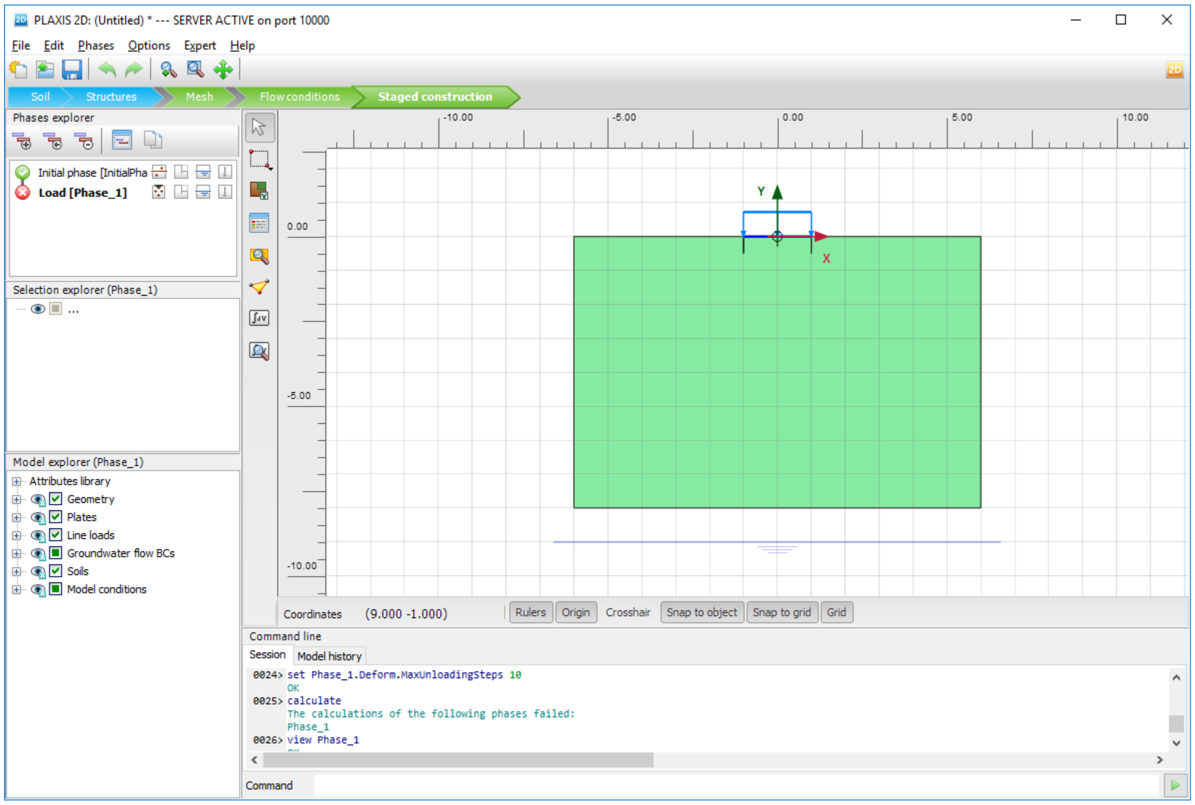
It's also possible to view and collect the calculated results with Python. The results can be further evaluated in Python, which also is an excellent tool for plotting results. It's highly recommended to read and go through the examples given in Appendix D in the reference manual of PLAXIS 2D (Brinkgreve et al., 2017).

6.3 Examples of Python usage in PLAXIS

When a new soil model is created it's recommended practice to test the model on a boundary problem. Different calculations can be of interest, like for an instance ultimate bearing capacity, settlements & creep and active & passive triaxial tests. The latter example can be simplified as an axi-symmetric problem with close enough resemblance in 2D. Writing scripts used in benchmarking allows for quick testing of new and existing material models.

An example of ultimate bearing capacity is shown below. The code is written in an object-oriented style and is based on the example "*A class that can be used to quickly create simple projects*" in Appendix D in the reference manual of PLAXIS 2D (Brinkgreve et al., 2017). The class `BearingCap` in the shown script creates a predefined model with a plate and high load. The new class `TestUSR` inherits everything in the `ElemTriaxialTest` class. The only change done in `TestUSR` is the selected material. In this way it's a simple matter of changing the material or material parameters, which can also be done in the script. The model and failure mode is shown in figure 6.2 and 6.3.

```
import sys, os
import numpy as np
from matplotlib import pyplot

from bearingCap import BearingCap, run

class TestUSR(BearingCap):
  pass
  def apply_soil_material(self):
    """
    Text displayed when help(TestUSR.apply_soil_material) is called
    """
    MATERIAL_PARAMETERS = [
    ('MaterialName', 'Userdefined Material'), ('Colour', 10676870),
    ('SoilModel', 100), ('DrainageType', 'Drained'),
    ('gammaUnsat', 20),   ('gammaSat', 20),
    ("InterfaceStrength", 1), ("Gref", 1), ("cref", 1),
    ('UserDLLName', 'usrmod.dll'),
    ('UserModel', "Geofuture soft clay model"),
    ('User1',50), ('User2',20), ('User3',100),
    ('User4',500),  ('User5',0.5), ('User6',0.35),
    ('User7',0.4), ('User8',0.1), ('User9',28),
    ('User10',0.7), ('User11',1.2), ('User', 2.3),
    ("K0PrimaryIsK0Secondary", True),("K0Determination", 0),("K0", 0.6)
    ]

    soilMaterial = self.model.soilmat(*MATERIAL_PARAMETERS)

    for soil_layer in self.model.SoilLayers:
    self.model.setmaterial(soil_layer, soilMaterial)

if __name__ == '__main__':
  run(testUSR)  # Execute script when run in Python
```



```
0024> set Phase_1.Deform.MaxUnloadingSteps 10
      OK
0025> calculate
      The calculations of the following phases failed:
      Phase_1
0026> view Phase_1
```

Figure 6.2: Predefined model for all soil models.

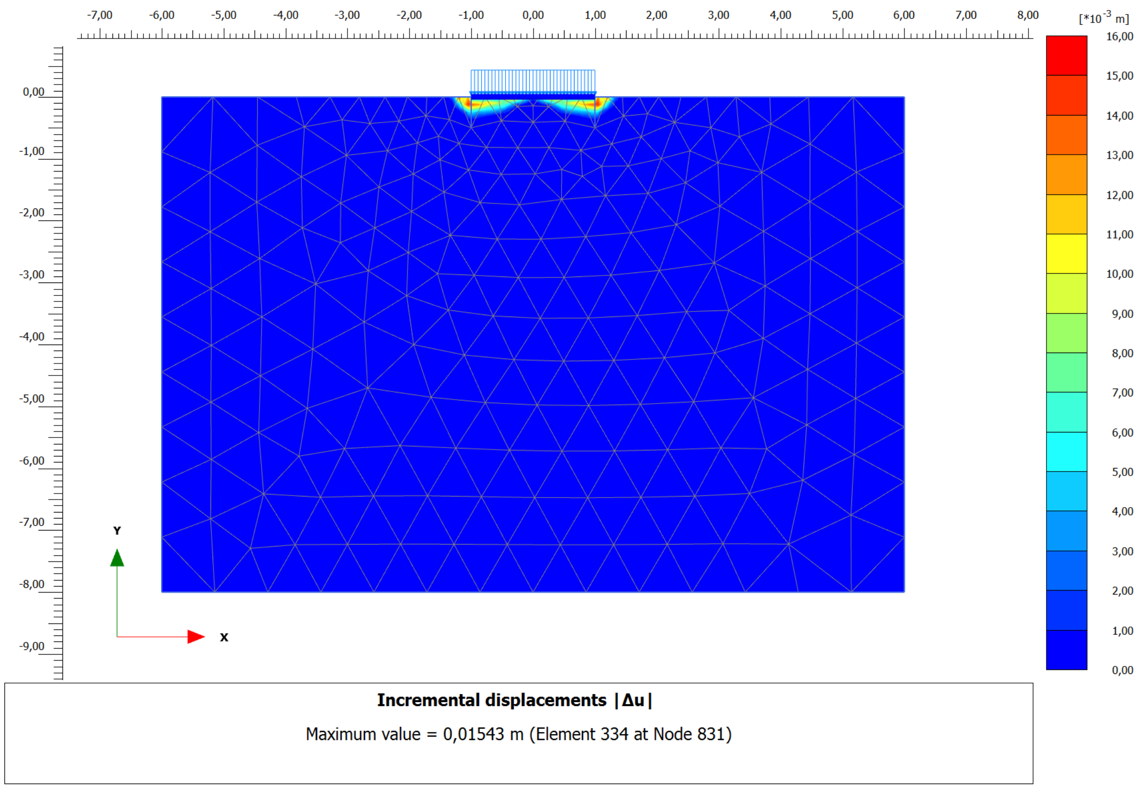


Figure 6.3: Incremental displacement at failure.

6.4 PyQt4 GUI

Often the popularity of a program or an application depends on whether or not it's intuitive. While a computer code may produce extraordinary results, the usage of the code may be severely delimited due to lack of knowledge. It's not reasonable to expect that the end user has a background in programming. It's therefor the programmers duty to ensure that the code can be reused by others.

One way of achieving this is by using a Graphical User Interface (GUI). The flow of a program can be more easily understood and can in many cases make the program self-explanatory. The end user can focus on input and output that follows. The programmer can on the other-hand control what is, and isn't allowed to do in the program. In that way the user must not search through ambiguous variable names in the code to change a certain setting.

Several GUI libraries exists for Python. Some are free while other requires a license for commercial purposes. It's here proposed to use the library PyQt4 or PyQt5. For free open source software and educational purposes no licenses are needed. PyQt provides bindings to the cross-platform Qt framework, which mainly uses standard C++ code. Both PyQt and Qt has good documentation and forums for Python or C++ can be used to solve Qt problems after some experience with the syntax.

Ideally the GUI design should be done in code. This ensures that the program is optimized, when done correctly. The code presented here is object-oriented since it makes more sense to do so when a program is event-driven, meaning the flow of the program is determined by the user actions like mouse clicks, key press and scrolling (Horstmann, 2013).

The figure 6.4 below shows a GUI window created from a Python script. The GUI is created from a `Window` class and is run until it's closed by the user, as stated in `sys.exit(app.exec_())`. The class has three functions defined. The `__init__(self)` function sets up the window, changes text, icons and calls `self.initialize()`. The `super(Window, self).__init__()` ensures that the object ID can be referenced as `self`, regardless of the object name (which in this case is `GUIobject`). The `self.intialize` creates a button and a link to the `self.exit_gui`. Whenever the GUI registers a mouse click on the button, the function `self.exit` is called and closes the GUI.

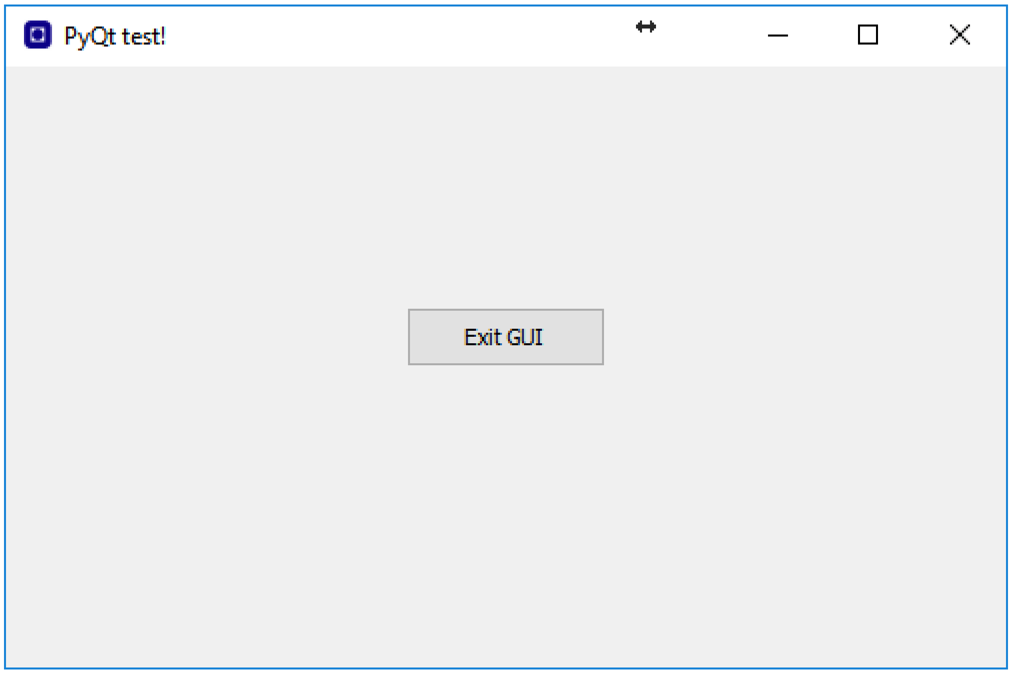
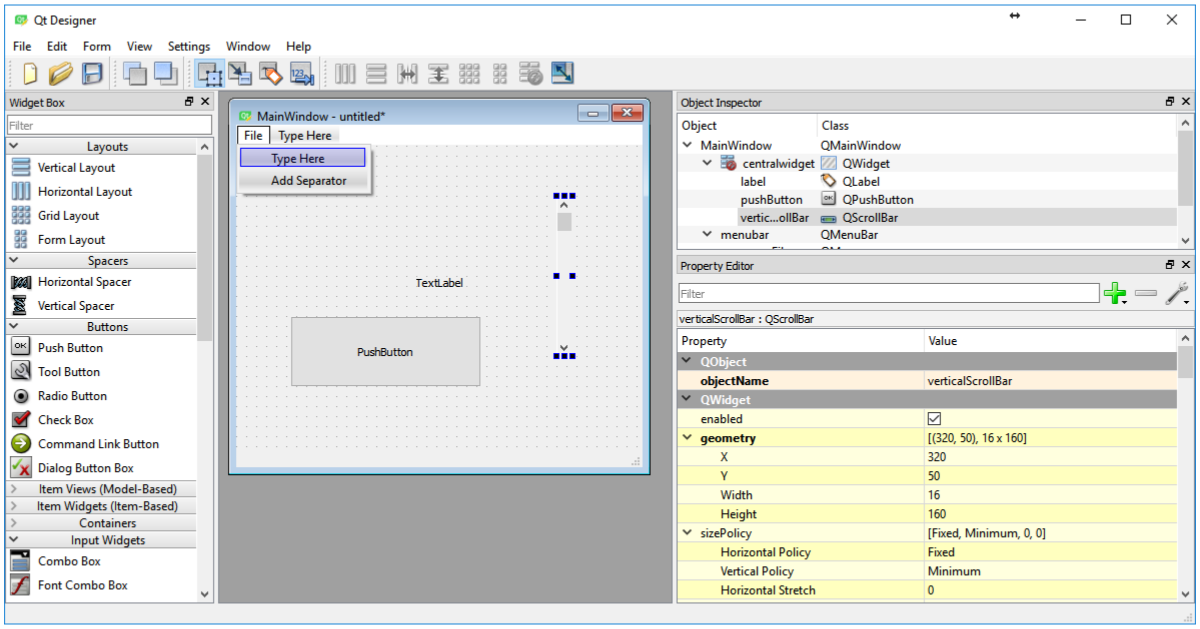


Figure 6.4: GUI window created by the Python code.

```
import sys
from PyQt4 import QtGui, QtCore

class WindowClass(QtGui.QMainWindow):
  def __init__(self):
    super(Window, self).__init__()
    self.setGeometry(50,50,500,300)
    self.setWindowTitle("PyQT tutorial 3!")
    self.setWindowIcon(QtGui.QIcon('ntnu.ico'))
    self.initialize()

  def initialize(self):
    btn = QtGui.QPushButton("Exit Gui", self)
    btn.clicked.connect(self.exit_gui)
    btn.move(200,120)
    self.show()

    def exit_gui(self):
    print("Text written in Python shell")
    self.close()

def runGUI():
  app = QtGui.QApplication(sys.argv)
  GUIobject = WindowClass()
  sys.exit(app.exec_())

runGUI()
```

A simpler approach is to use the Qt Designer program which is displayed in figure 6.5. This allows for rapid creation of the designs. Layouts, buttons, labels, sliders and many more GUI widgets are simply dragged out onto the window. Each widget (like a button) has many attributes that can be edited in the Property Editor. The Qt file is saved as XML code with the file-extension `.ui`. *PyQt4* provides an interpreter *pyuic4* which can be used to interpret the XML code to Python code. Another solution is to reference the `.ui` file directly in the Python script as shown below. This gives no control over the created GUI code, but modifications in the design can easily be done with Qt Designer afterwards.



Object	Class
MainWindow	QMainWindow
centralwidget	QWidget
label	QLabel
pushButton	QPushButton
vertic...ollBar	QScrollBar
menubar	QMenuBar

Property	Value
QObject	
objectName	verticalScrollBar
QWidget	
enabled	✓
geometry	[(320, 50), 16 x 160]
X	320
Y	50
Width	16
Height	160
sizePolicy	[Fixed, Minimum, 0, 0]
Horizontal Policy	Fixed
Vertical Policy	Minimum
Horizontal Stretch	0

Figure 6.5: Qt Designer environment.

```
from PyQt4.uic import loadUiType
Ui_MainWindow, QMainWindow = loadUiType("fileName.ui")

class Main(QMainWindow, Ui_MainWindow):
def __init__(self):
  super(Main, self).__init__()
  self.setupUi(self)
```

6.5 Destructor

A Python script named *Destructor* has been developed. Its purpose is to be an interface between PLAXIS and the cyclic calculations. The script automates the cyclic analysis by adding new phases, adding materials, changing load and ensuring coupling between material models. The script works in 6 steps and is shown in figure 6.6.

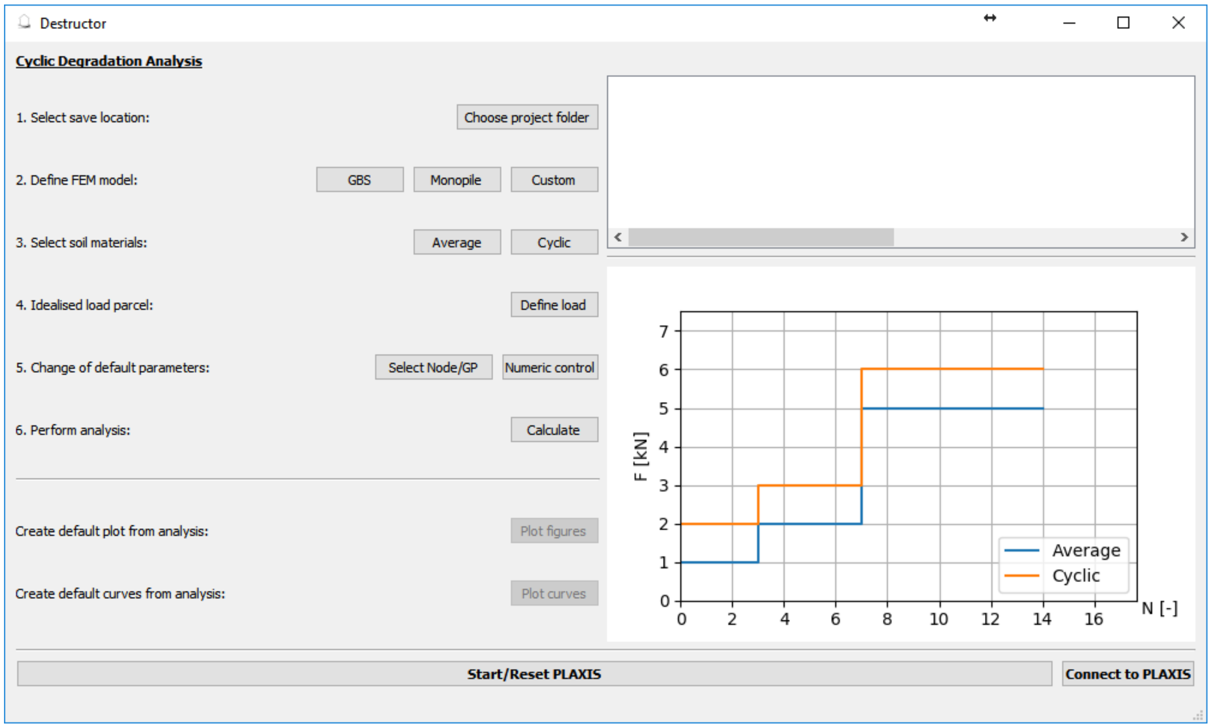


Figure 6.6: GUI interface of the Destructor script.

1. First a folder where the project will be saved is selected. It serves no purpose during the actual calculations. When the calculations are finished the model is saved to this folder.

2. Secondly a FE model needs to be defined. At the moment there are two predefined models, Gravity Based Structure (GBS) and Monopiles, represented as a wall in 2D. It's also possible the create a custom model of the type shown in figure 6.7.

3. Thirdly the materials used in the model must be defined, which is done in the Python script. The user-defined DLL files in the PLAXIS UDSM are imported and read using the *cython* library. The intention is that the material input is a stripped version of the one found in PLAXIS. This is to reduce confusion and to ensure some parameters are kept constant, as in weightless undrained soil during cycling.

4. Afterwards the horizontal average and cyclic load and number of cycles needs to be defined. It's here assumed that the load parcels are already determined. A plot of the load parcels is shown after they have been defined. It was also the intention that zero cyclic loading corresponds to consolidation in the average phase for the equivalent amount of time. This has not yet been implemented.

5. The fifth step is optional and allows the user to select Nodes or Gauss Points in PLAXIS. The numerical control button was supposed to allow the user to change the amount of iterations between the average and cyclic models, mesh fineness and default numerical controls in PLAXIS. Currently only the amount of iteration can be changed from the GUI, and the rest must be redefined in either the code or in PLAXIS directly.

6. The last step is where it all comes together. The script will try to calculate the cyclic degradation. If one of the steps are skipped then an error box is shown. After each phase, text files used in the coupling are created. When the calculations are finished the text files are removed.

The white canvas up to the left in figure 6.6 was intended to be an interactive time series of the phases. Since finding a particular phase in PLAXIS becomes messy after many calculations, it would be better to have a different way of examining a certain phase. In the time series plot the load, cycles and parcel number could easily have been displayed prior to opening PLAXIS output.

There was also the idea of creating default plots and figures after calculation. But this has not yet been implemented due to lack of time and not knowing which information is the most relevant when regarding cyclic degradation. One solution is to let the user to decide which figure and plots to save.

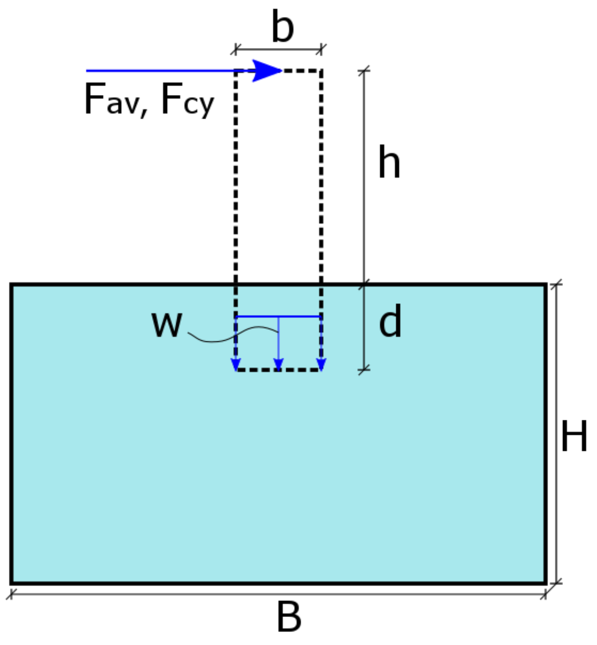


Figure 6.7: Base model used in *Destructor*.

Chapter 7 Generating Soil Models with Fortran & Matlab

7.1 Why Fortran and Matlab

Although technically being a high-level programming language, today Fortran seems quite primitive compared to other high-level languages, especially Python and Matlab. Fortran was released in the late 50's and is still very much alive. Historically the computer hardware was pretty limited in speed and capacity, so optimization in code and compiling[1] was vital. Even though many modern programming languages has impressive numerical speed performance, well written Fortran code is hard to beat due to it's history of constant optimization Chapman (2003). Over the course of years Fortran has been updated to keep up with the present time, while ensuring that legacy (old) code will still work. In later years Fortran has also been improved for object-oriented, parallel and mixed-language programming. A comment found on the internet stated that "*Fortran is like Rock 'n Roll, it will Never Die!*"

For implementation of material models in a FE program normally a compiled file is required. In PLAXIS a Dynamic-Link Library (DLL) is needed. The DLL file is used for every Gauss Point in the model and local equilibrium must be established for each point. A well written optimized code can here save precious computing time.

[1]Compiling is a computer program that transforms the source code to binary code.

When it comes to matrix calculations Matlab is one of the easiest platforms to start with. While being simple in use it's still fully capable of handling advanced numerics. Over the years many toolboxes (similar to libraries) has been developed which makes Matlab an good all-rounder for solving all sorts of problems. That being said Matlab is a proprietary product of MathWorks and requires a license to use, and each toolbox comes at an additional fee.

Matlab is also an interpreted language like Python and the software is developed in scripts. Unlike some programming languages Matlab has an extensive documentation both for the language itself and for mathematical issues. Due to its easy syntax Matlab is actively being used by engineers, scientists and students. The are many active forums with a thriving community.

Although many of the same problems could be handled by Python, Matlab has been chosen for generating Fortran code since the ground work were already made by Jon A. Rønningen. Files for generating Fortran code, solving for multi-dimensional gradients and a working strain driver was preexisting thanks to him. In the authers opinion Matlab is also an easier language to start with and might let the user focus more on the equations and material formulation rather than actual programming.

7.2 Object-Oriented Fortran

Since Fortran 90 the program unit `module` has existed. Modules is neat way to bundle together small packages of variables, subroutines and functions. The main difference between simply including additional code versus using modules, is that an explicit interface is created. All the details in a module are available during compiling and the compiler may discover common errors. This also allows for more sophisticated software design (Chapman, 2003).

An example of using modules can be in the case of defining precision. In Fortran there are several ways of specifying double precision. Depending on the Central Processing Unit (CPU) a float value may occupy 4, 8 or more bytes. By ensuring enough bytes is available one can make sure double precision is kept, but at the cost of memory. Since *enough* is a vague definition for processors it's better to have it explicitly defined. By specifying the precision (significant digits) and range (power of 10 exponent) Fortran can determine the appropriate byte size. An example where the precision can easily be changed is shown below.

```
MODULE precision
  INTEGER, PARAMETER :: prec = SELECTED_REAL_KIND(p=15, r=10)
END MODULE precision

PROGRAM testPrecision
  USE precision
  REAL(kind=prec) :: a, b, c
  a = 1.0D0
  b = 3.0D0
  c = a/b   ! Returns 0.333... with 15 digit precision
END PROGRAM testPrecision
```

In Fortran 2003 new features where introduced to improve the object-oriented programming flow. One feature was the type-bound procedures which ensured that variables, subroutines and functions could not be accessed without referencing the object itself (Metcalf et al., 2004). That way multiple subroutines with the same name could be called, while giving different results. In a module several types ("classes") would be stored. Each type could be given its own type-bound variables and procedures. A new variable of the same type ("object") in the main program could then be created and use the type-bound procedures. An example of a counter is given below.

```
MODULE counterLibrary
  IMPLICIT NONE
  PRIVATE      ! Only the type is accessible in the outside program
  TYPE, PUBLIC :: CounterClass
    INTEGER :: counter
    CONTAINS
      PROCEDURE :: addOne => internalFunctionName ! Type-bounding
  END TYPE CounterClass
  CONTAINS
  SUBROUTINE internalFunctionName(this) ! Arbitrary subroutine name
    CLASS(CounterClass), INTENT(INOUT) :: this  ! Define variable
    this%counter = this%counter + 1      ! Use variable inside object
  END SUBROUTINE internalFunctionName
END MODULE counterMod

PROGRAM testCounter
  USE counterLibrary
  IMPLICIT NONE
  TYPE(CounterClass) :: objectCounter ! Define object variable
  objectCounter = CounterClass(1)     ! Create object & set counter = 1
  CALL objectCounter%addOne           ! counter = counter + 1 = 2
END PROGRAM testCounter
```

While the object-oriented Fortran seems a bit more verbose, there are some clear advantages. Many counter objects can be created, while the procedures stays same for each object. Another advantage is that types can *inherit* the procedures from one type and can be expanded. In this way new type can be molded from an old type as shown below.

```
  MODULE counterLibrary
  IMPLICIT NONE
  PRIVATE
  TYPE, PRIVATE :: CounterClass ... ! Type cannot be used in main code
  TYPE, PUBLIC, EXTENDS(CounterClass) :: CounterClassExt
    LOGICAL :: logicalValue
    CONTAINS
      PROCEDURE :: switch => newFunctionName
  END TYPE CounterClassExt
  CONTAINS
  SUBROUTINE internalFunctionName(this) ...
  SUBROUTINE newFunctionName(this)
    CLASS(CounterClassExt), INTENT(INOUT) :: this
    IF (this%logicalValue) THEN
      this%logicalValue = .FALSE.
    ELSE
      this%logicalValue = .TRUE.
    END IF
  END SUBROUTINE newFunctionName
END MODULE counterMod
  PROGRAM testCounterExt
    USE counterLibrary
    IMPLICIT NONE
    TYPE(CounterClassExt) :: objectCounter
    objectCounter = CounterClassExt(1, .TRUE.)
    CALL objectCounter%addOne       ! counter = 2
    CALL objectCounter%switch       ! logicalValue = .FALSE.
  END PROGRAM testCounterExt
```

7.3 Generating Soil Models

A framework for generating soil models to use in PLAXIS has been developed. The code is heavily based on the soil models created by Jon. A. Rønningen. The largest change is the flow and design of the code. A flow chart is shown in figure 7.1. The soil models are written in an object-oriented Fortran style. Where possible modules has been used. Clarity in the code has been preferred over performance, although there's room for more of both.

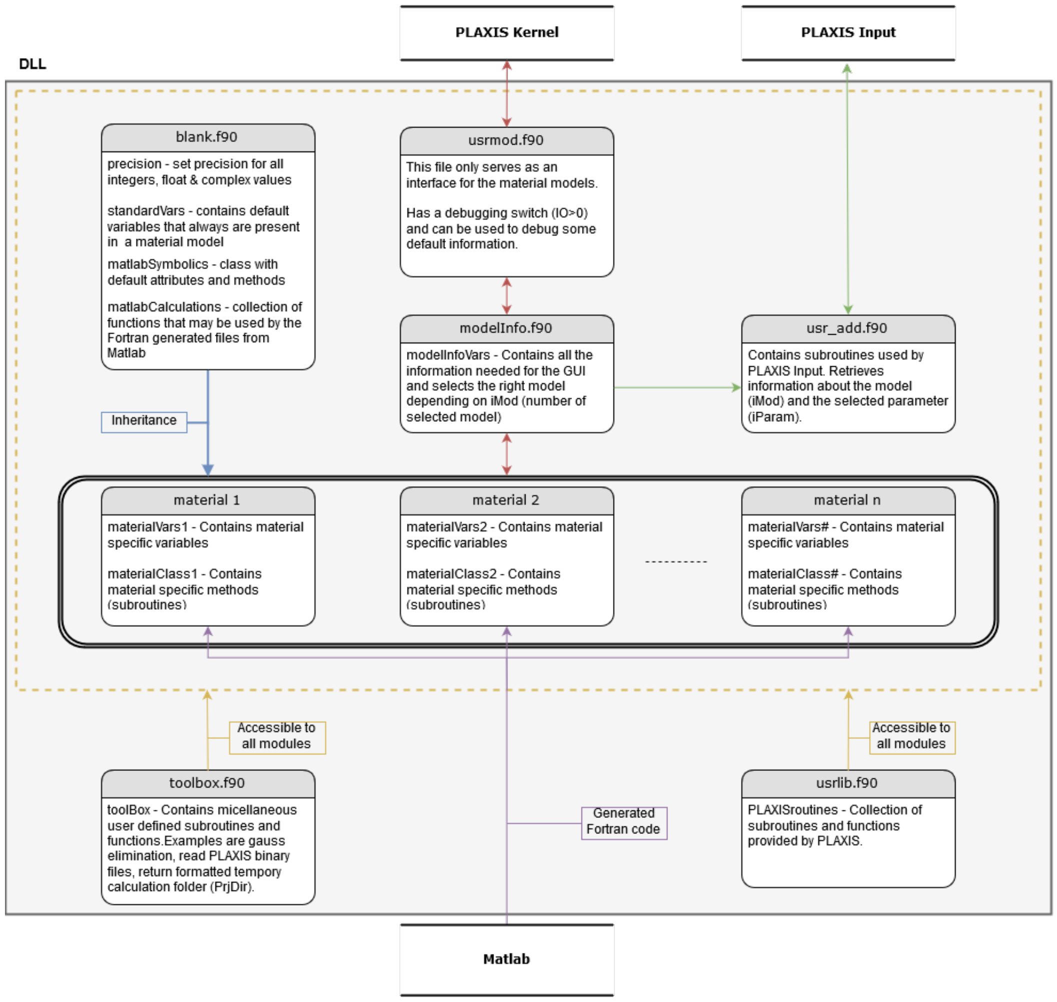

PLAXIS Kernel
PLAXIS Input
DLL
blank.f90
precision - set precision for all integers, float & complex values
standardVars - contains default variables that always are present in a material model
matlabSymbolics - class with default attributes and methods
matlabCalculations - collection of functions that may be used by the Fortran generated files from Matlab
usrmod.f90
This file only serves as an interface for the material models.
Has a debugging switch (IO>0) and can be used to debug some default information.
modelInfo.f90
modelInfoVars - Contains all the information needed for the GUI and selects the right model depending on iMod (number of selected model)
usr_add.f90
Contains subroutines used by PLAXIS Input. Retrieves information about the model (iMod) and the selected parameter (iParam).
Inheritance
material 1
materialVars1 - Contains material specific variables
materialClass1 - Contains material specific methods (subroutines)
material 2
materialVars2 - Contains material specific variables
materialClass2 - Contains material specific methods (subroutines)
material n
materialVars# - Contains material specific variables
materialClass# - Contains material specific methods (subroutines)
Accessible to all modules
Accessible to all modules
toolbox.f90
toolBox - Contains micellaneous user defined subroutines and functions.Examples are gauss elimination, read PLAXIS binary files, return formatted tempory calculation folder (PrjDir).
Generated Fortran code
usrlib.f90
PLAXISroutines - Collection of subroutines and functions provided by PLAXIS.
Matlab

Figure 7.1: Flow chart of material DLL design.

usrmod.f90 This file can be considered as the "main program". Whenever the DLL file is used in PLAXIS, this file will serve as an interface. The information from here is passed to *modelInfo.f90* There is also a debugging switch in the usrmod subroutine which can be used in the models, although it's not necessary to use it when debugging.

modelInfo.f90 In this file the module modelInfoVars can be found. Here the number of material models, model names, parameter & state variable font and units are defined. The information sent from PLAXIS is also redirected to the right model. Ideally this is the only Fortran file that needs to be edited prior to compiling.

usr_add.f90 If this file is not included then PLAXIS will show a generic material input window. There will be no information about which parameter goes where. When this file is included the information given in *modelInfo.f90* will be displayed correctly in the PLAXIS Input program.

blank.f90 This file is the parent file of all materials. Whenever a new material is created, it will inherit the procedures given in *blank.f90.* There's also a module for precision and a module for Matlab procedures used in Fortran.

toolbox.f90 It's the idea that the module *toolBox* consists of many user defined procedures. In here one may find procedure for linear algebra, reading binary files from PLAXIS, formatting strings and vectors and so on. The toolbox should be available for every module if needed.

usrlib.f90 In the example user defined materials from PLAXIS this file is included. Here there are all sorts of procedures as solving for invariants and eigenvectors, neatly formatting vectors and matrix and so on. This file should also be available for every module if needed.

material files These files are entirely generated by Matlab and should ideally not needed to be edited. The material rules and equation are defined and solved with the Computer Algebra System (CAS) *Symbolics Toolbox* in Matlab. The Matlab symbolics are afterwards saved as optimized Fortran code. It's the goal that everything regarding the material definition can be written in Matlab. Using explicit or implicit solving techniques, visco-plasticity or stress dependent stiffness matrices should all be just a matter off flipping a switch. In that way one can focus more on the essence of soil modeling instead of Fortran programming. At the moment the Matlab script is pretty limited and can only be used to model elasto-plastic materials, implicit Euler with a modified Newton-Raphson iteration. The script also requires further testing before it can be used in all sorts of advanced material models. It's also worthwhile mentioning that Python also can do CAS calculations and can convert symbolics to Fortran code.

7.4 Verification of Generated Soil Models

For verification an Isotropic Von-Mises and a Modified Cam-Clay model has been created. The Von-Mises model is an isotropic linear-elastic, perfectly plastic model while the Modified Cam-Clay is supposed to be the same as the one provided by PLAXIS given that the void ratio, e_0, is zero.

The input parameters in the Von-Mises model has been given as $S_u = 50kPa$, $E = 3000kPa$ and $\nu = 0.3$. The material has been tested both in drained and undrained triaxial. In table 7.1 the input parameters are compared with the values determined in Soil Test. In figure 7.2 the incremental displacement at failure fits well with the theoretical bearing capacity failure mode. The ultimate load capacity should be close to the theoretical Tresca $q_{ult} = (2+\pi)S_u = 257.1kPa$. The calculated capacity in PLAXIS was $q_{failure} = 308.6kPa$ which is considered close enough when accounted for plate flexibility and presence singularities.

Table 7.1: Gauss Point verification of Von Mises.

Unit	User model	Theoretical
S_u	49.9	50
E	3026.4	3000
E_u	3447.9	3450

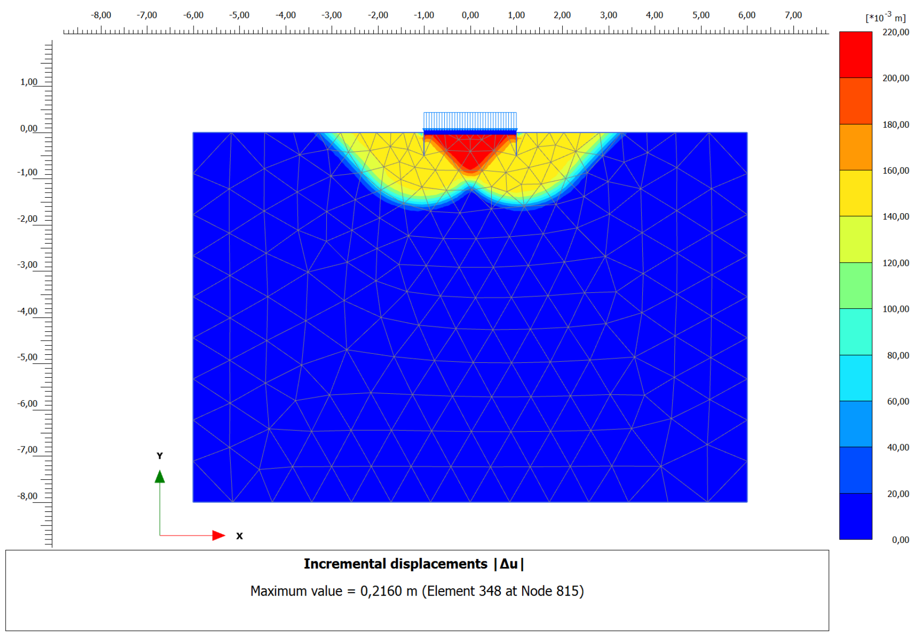


Figure 7.2: Incremental displacement at failure - Von-Mises.

The input parameters in the Modified Cam-Clay where $\phi = 25.4$, $\lambda = 0.25$, $\kappa = 0.05$, $\nu = 0.15$, and $OCR = 4.0$. The user defined material has been tested in undrained triaxial Soil Test and been compared with hand calculations. The results are shown in table 7.2. The material has also been compared with the in-built model in PLAXIS. Both materials has been compared in ultimate bearing capacity. The results are considered good enough, although discrepancies are observed. This might be due to different numerical schemes and defining $e_0 = 0$ in the user model. A comparison of the deviatoric stress is shown in figure 7.3. The failure load was calculated as $q_{usrMatr} = 249 kPa$ for the user material and $q_{plxMatr} = 218 kPa$ for the in-built PLAXIS material.

Table 7.2: Gauss Point verification of Modfied Cam-Clay.

	Undrained Triaxial		
	Unit	User model	Theory
$OCR = 4$	τ_{yield}	262	261.2
	S_u	261.91	259.8
	P_w	−48.96	−48.2

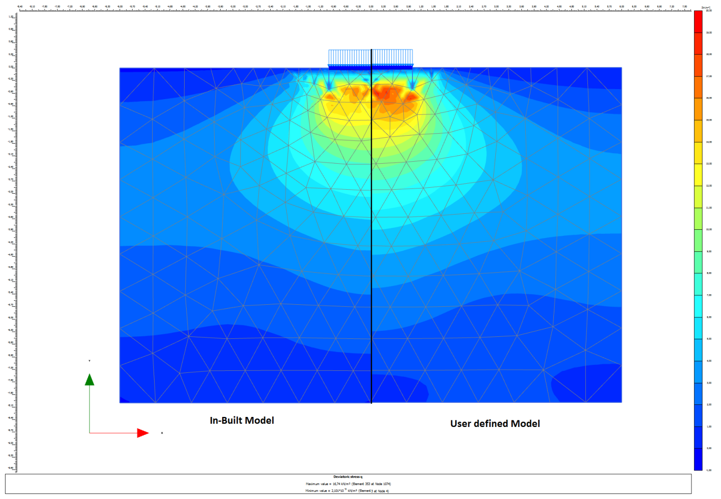


Figure 7.3: Deviatoric stress at failure - Modified Cam-Clay.

Chapter 8 Simulation and Results

8.1 Simulation Model

A Modified Cam-Clay with destructuration and a non-linear elastic model has been coupled together. Predefined FE model described in section 6.5 has been used in the cyclic analysis. Four simulations are shown as a proof of concept. They also show some important details and been tested with different load combinations. Ideally the framework should also have been able to generate a cyclic contour diagram based on simulation results, but this has not been implemented.

8.2 Material Coupling

The simulation starts with the script Destructor. From here the FE model, soil materials and loads are defined. During the calculations the script will play a vital part in the coupling. Firstly each phase is calculated one by one and checked for failure. If success is registered then text files are created. These files contains information about which previous PLAXIS binary file with state variables are stored and the initial structure χ_0 is saved. A file used for tracking Gauss Point ID is also reset when a phase is done. The presence of these files dictates whether or not coupling occurs in the soil models.

Without the text files created by Python, the soil models can be used independently of each other. But they may not transfer any state variables. Since the same mesh is used throughout the simulations, the Gauss points will be calculated in the same pattern each time. One of the external text files is used to keep track of which Gauss point is being accessed and updated when finished. In the case of the first Gauss point the state variables in the PLAXIS binary file will be copied to a new binary file with direct access Chapman (2003). By knowing the actual Gauss point tracker value and the length of the state variables in the last soil model, the coupled state variables can be directly accessed from the new binary file.

8.3 Results

The material parameters has been used consistently in each simulation. The material input parameters are provided in table. 8.1. The unit weight of the average material was set to $\gamma = 10\frac{kN}{m^3}$ and consolidated with $K_0 = 1.0$. The structure weight was set as a line load at $w = 10kPa$. The cyclic phases was done weightless, meaning no vertical forces.

Table 8.1: Material parameter used in simulations.

Cam-Clay		Non-linear Elastic		Plate	
ϕ	25.4	S_u	$50kPa$	EA	$10.4E6kPa$
λ	0.25	E	$200kPa$	EI	$104.3E3kPa$
κ	0.05	α	0.3	w	$0kPa$
OCR	4.0			ν	0
ν	0.15				
ξ	0.5				
ω	0.5				
χ	20				

Constant low loads - Wall

The first simulation is to show that there is a small degradation when a low load acting shortly is applied, and that the degradation converges. Both the cyclic and average load is $5kN$ and only acts for one cycle. The iteration between the models can diverge if care is not taken. In the degradation formulation there is a cut-off at $N > 10000$ cycles and the deviatoric stress must minimum be 5% of S_u. If the formula for N_{eq} is not explicit then the calculation can be extremely prolonged and one might risk that $N \rightarrow \infty$. The second rule is introduced to speed up calculation, basically ignoring degradation effects if the shear stress is too low. The structure χ after a certain amount of cycles is shown in figure 8.1. The required intermediate steps are load and material dependent, although two or three iterations seems to capture the major effects.

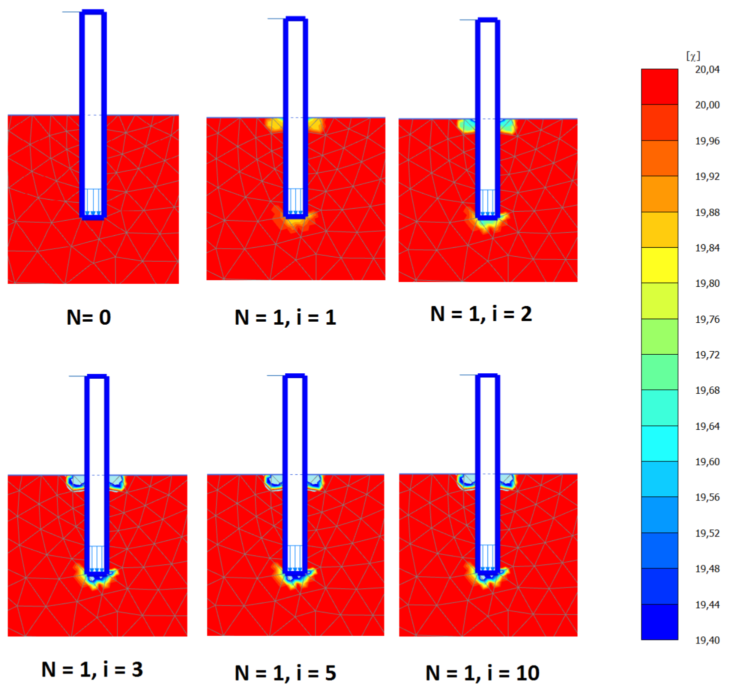


Figure 8.1: Intermediate calculations of degradation.

Reduction of capacity - Wall

The simulation here is to verify that the strength capacity is reduced after cyclic loading. While the average load is the same, the cyclic load will reduce the yield surface. For equilibrium to be reached the pore pressure must therefor increase to compensate for the yield reduction. The average load was set to $F_a = 50kN$ and the FE model was cycled at $F_{cy} = 75kN$ for $N = 1000$. The average phase was brought to failure before and after cyclic degradation by applying a higher horizontal load of $F_a = 100kN$. The reduction of capacity is shown in figure 8.2. The change in pore pressure is shown in figure 8.3.

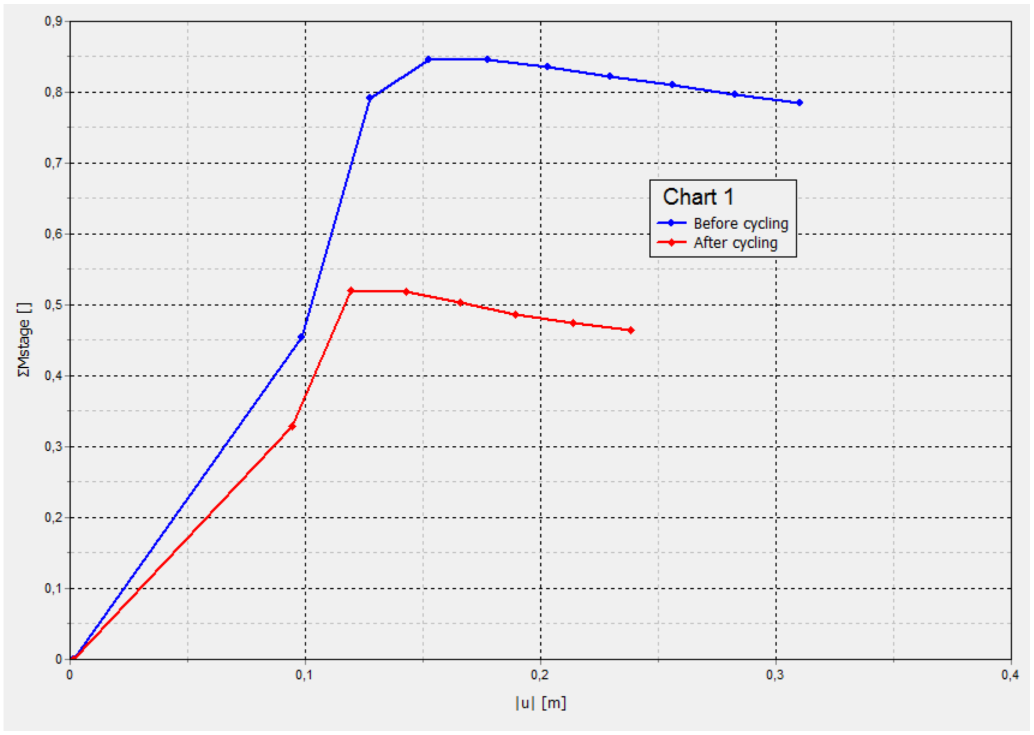
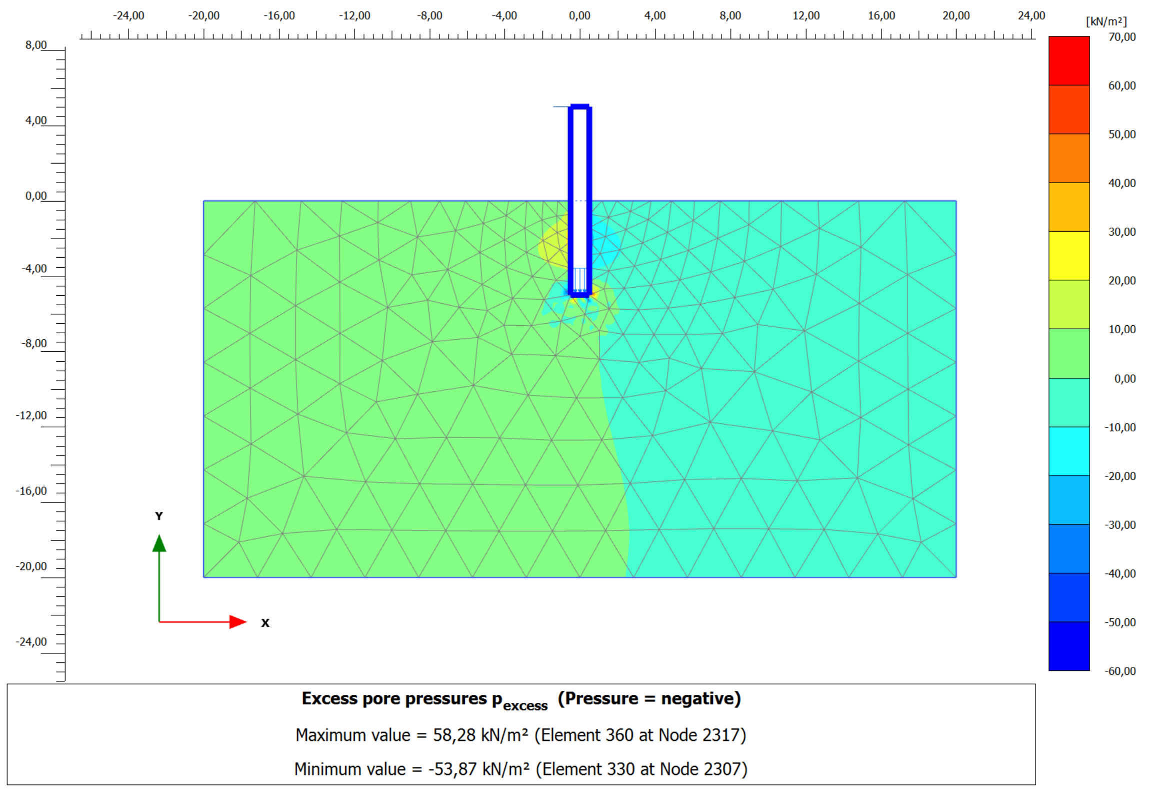


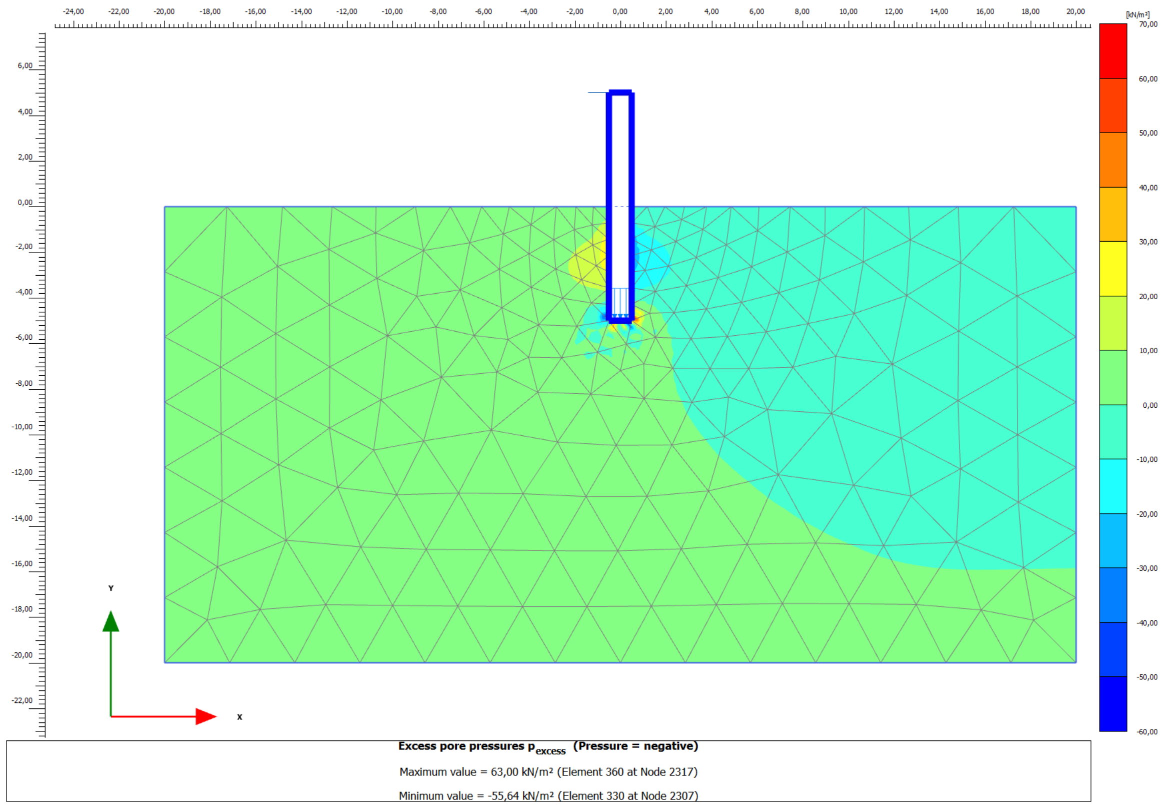
Figure 8.2: Reduction of strength capacity after cyclic degradation.



Excess pore pressures p_{excess} (Pressure = negative)

Maximum value = 58,28 kN/m² (Element 360 at Node 2317)

Minimum value = -53,87 kN/m² (Element 330 at Node 2307)



Excess pore pressures p_{excess} (Pressure = negative)

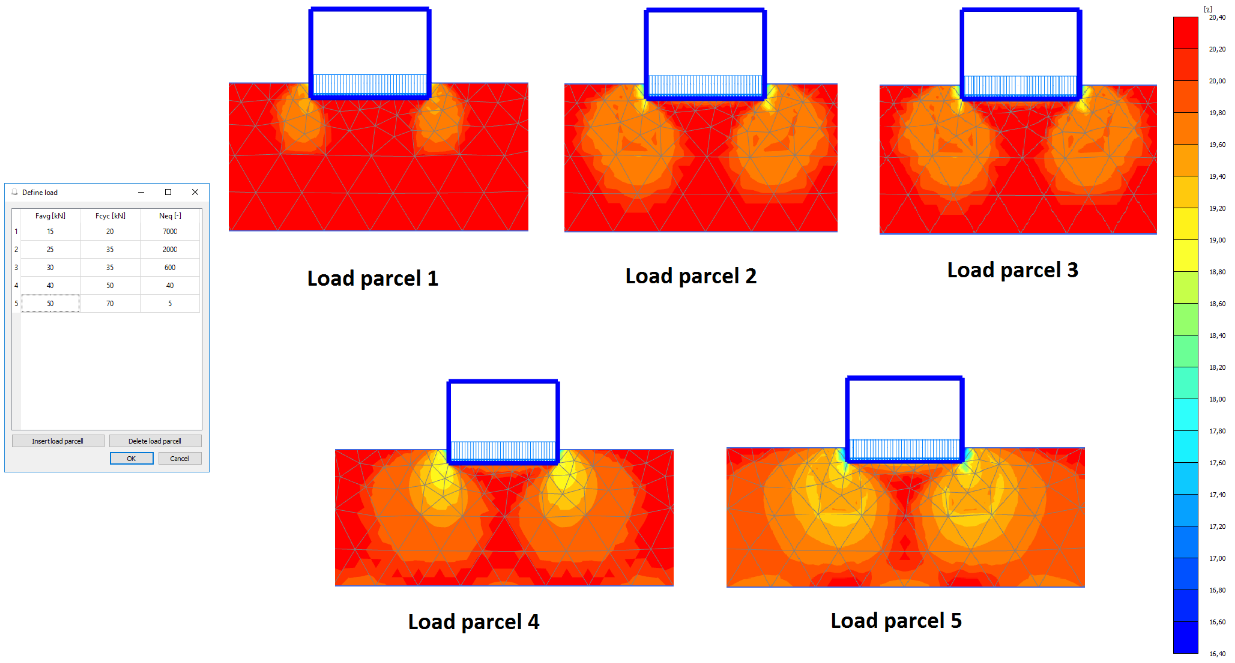
Maximum value = 63,00 kN/m² (Element 360 at Node 2317)

Minimum value = -55,64 kN/m² (Element 330 at Node 2307)

Figure 8.3: Comparison of pore pressure change due to cycling.

Storm load - GBS

The method shown in section 3.5 assumes that the load level increases for each load parcel. The idea is that by doing so, all the degradation up until failure is accounted for. Since this has been the practical design method for many years then this framework should also be able to handle the same. For the right material models these kind of simulations can be used as a verification. It's here assumed that a fictive number of load parcels with increasing load can be a representation for a design storm. The degradation of the structure parameter χ is shown in figure 8.4. The model also accumulates permanent deformation.



	Favg [kN]	Fcyc [kN]	Neq [-]
1	15	20	7000
2	25	35	2000
3	30	35	600
4	40	50	40
5	50	70	5

Figure 8.4: Structure χ after each load parcel.

Settlements - GBS

Tests done on GBS shows that cycling increases the long-term settlements (Andersen et al., 1988). In the simulation a GBS has subjected to a horizontal of $F_a = 20kN$ and $F_{cy} = 50kN$ for $N = 1000$ cycles. The permeability was set to $k = 5 \cdot 10^{-5} \frac{m}{day}$, although in this case it's simply used as a scalar. Prior and after cycling the soil is allowed to fully consolidate with one-way vertical drainage. The results shown in figure 8.5 matches well with the observed behavior in the real world.

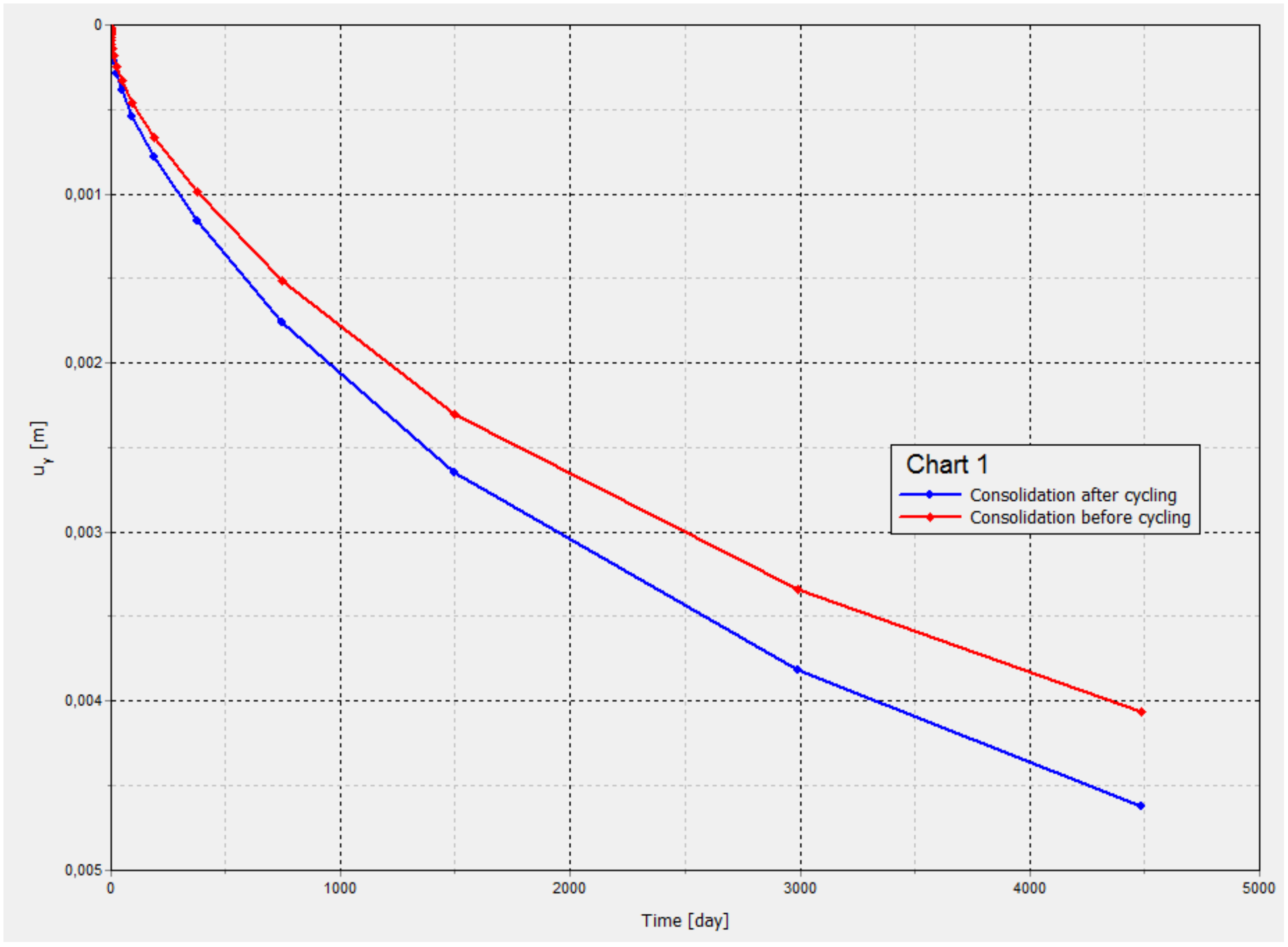


Figure 8.5: Comparison of settlements with and without cycling.

Simulation remarks

The introduction of cut-offs in the degradation formulations ensures that convergence is achieved. But it also creates a barrier from introducing more degradation. This may be remedied by a better cyclic model. There's also the issue of numerical instability. Since the structure χ is not used in the equilibrium and compatibility of the model, this may create sudden singularities in the average model. Even worse it gets for high cyclic loads since strains gets absurdly large.

Although some tests are shown with great success, the capability of the framework and soil models is not without its faults. Some common errors are load advancement procedure failure, Fortran error and numerical instability. The first problem is partly solved by introducing an error to the applied load, such that there is a difference. However this greatly depends on the magnitude of the load. The second problem should not really be an issue, since the script removes all files after calculation. On the off-chance this is not done, simply restarting PLAXIS will often solve the problem. The last problem can maybe be solved by introducing explicit ODE schemes with sub-stepping of increments. When rapid changes happens in the model, explicit methods tends to be the better choice, and inverting ill-condition matrices is avoided.

As observed in laboratory tests the degradation causes an increase in pore pressure. This effect can be simulated here two ways. The first approach is that the material fails in the cyclic phase. To get a numerical well-behaved failure mode it's recommended to use an elasto-plastic model. The second approach is that the average model fails due to extensive degradation. This is perhaps the best method since the initial stress state will get close and closer to the failure point. As the total amount of cycles increases, so does the degradation. Ultimately the material fails due to loss of structure. If the material is undrained then the pore pressure must compensate for the degradation.

The stress is actually created with an elastic model since there were some problems with switching materials. The K_0 procedure generates a stress which increases with depth. When the unit weight is removed it's the same as applying an upside-down hydrostatic pressure, effectively a reverse K_0 procedure. A temporary material can be introduced to reset displacements, but there will still be some deviatoric stresses in the bottom. One remedy can be to change the boundary conditions such that the bottom is not entirely fixed.

Chapter 9 Summary and Recommendations for Further Work

9.1 Summary and Conclusions

The theory covered in chapter 3 seems to suggest that cyclic loading causes degradation in soils. Accumulation methods based on contour diagrams has been successful used in design calculations and may serve as a basis for a degradation formulation. It's also noted that strain controlled tests causes more degradation than stress controlled tests.

Suggestions for requirements in the average and cyclic models have been defined. A state variable called structure, χ, is introduced and linked with the destructuration of the average model. A cyclic material with a simple degradation formulation based on the contour diagrams has been developed. The degradation formulation in itself is able to create the wanted effects viewed in laboratory tests. The models have been created by using the Fortran & Matlab framework for generating soil models.

A Python framework has been developed and tested for multiple simulations. Although the framework is not completely finished, the main functionalities are present. The coupling of state variables has successfully been implemented and works in conjuncture with Python. The simulation shows promising results. Cycling a material causes reduction in ultimate capacity when undrained, degradation and deformation is accumulated for increasing cyclic loads and cycling causes extra settlements.

The developed framework has a great potential. It's possible to create soil models rapidly with the material generation, even though the available material options is pretty limited. The Python script Destructor and material models has proven that they can replicate common features observed in real tests and may serve as building blocks for more advanced soil models.

9.2 Discussion

Almost surprisingly the simple degradation formulation is able to capture some important effects in the FE analysis. The simulation results really shows the strength of the framework presented here. But the simulations also showed that the models a prone to numerical instability. This may be avoided by a better degradation formulation or possibly by using an explicit ODE solver schemes with small step increments.

The script Destructor works as intended, and it's very pleasing to see that everything is working together.

9.3 Recommendations for Further Work

Regarding the theory it's recommended to further study the actual cause of degradation. A good knowledge of the particles behavior at a micro-level can be the key to create an elegant explicit degradation formula. With regards to this it might be interesting to carry out some discrete element models to see how the particles rearranges.

Although presented, the material Geofuture has not been tested in degradation simulation. It would be very interesting to see the effects of using a model with anisotropic yield surface and mean pressure dependent elasticity. The excess pores pressure created during undrained calculation may cause a stiffness reduction as the one observed in laboratory tests.

The simple degradation formulation is considered far from complete and is considered to crude. The current formulation cannot be used for general soils, and the material parameters are not determined by easily obtainable laboratory results. There's also the question if plasticity should be introduced when calculating degradation and cyclic failure.

The Python framework presented is able to be used in cyclic degradation. Some features has not yet been implemented as described in section 6.5, but they are not vital to the calculations. It would be nice to have a more proper way of generating stresses in the soil.

The modular build up of material models using Fortran and Matlab is currently ready to use. The next step would be to introduce more numerical schemes, options for creating purely elastic, visco-elastic and visco-plastic models. A big step would be to generate a *modelInfo.f90* file directly with Matlab. In that way no Fortran knowledge is needed to create a material model, and the threshold for creating models is thereby considerably lower.

Lastly the most important part is that simulations must be verified with actual test results. It's here recommended to compare the results with the GBS results from Andersen et al. (1988). It's also possible to verify the material models with DSS and triaxial results depending on the cyclic material formulation. An interesting task would also be to generate some cyclic contour diagrams from the proposed models.

List of Figures

List of Tables

Appendix A Programs used

In the development of the Python and Fortran framework the following programs and versions has been used.

Python

The IDE Pyzo has been used with the Miniconda interpreter (which is a modified Python interpreter). The idea of Pyzo is to be the free alternative to Matlab and therefor has a similar working environment. The Pyzo version 4.3.1 has been used together with Python 3.6.0 and the latest version of PyQt4 4.11.4.

Matlab

In the development Matlab R2017 64-bit has been used.

Fortran

Intel Fortran 2010 has been used to both compile 32- and 64-bit DLL files. The code was written in the text editor program Atom with syntax highlighting of Fortran.

PLAXIS

Different versions of PLAXIS 2D 2016 has been used in the development of the framework.

Bibliography

NS-EN 1993-1-9:2005.

Offshore wind project cost outlook 2014 edition.

Andersen, K. H. (1976). Behaviour of clay subjected to undrained cyclic loading. *Norwegian Geotechnical Institute, Oslo. Publication, 112, pp. 21-28.*

Andersen, K. H. (2015). Cyclic soil parameters for offshore foundation design. In Meyer, V., editor, *Frontiers in Offshore Geotechnics III*, volume 1, pages 5–82. CRC Press.

Andersen, K. H., Kleven, A., and Heien, D. (1988). Cyclic soil data for design of gravity structures. *Journal of Geotechnical Engineering*, 114(5).

Brinkgreve, R. B. J., Kumarswamy, S., and Swolfs, W. (2017). *PLAXIS 2D Manual.* Delft University of Technology and PLAXIS bv, The Netherlands.

Burland, J. B. (1990). On the compressibility and shear strength of natural clays. *Géotechnique*, 40(3):329–378.

Cai, F. (2015). *Modeling of Strain Accumulation in Clays under Cyclic Loading.* PhD thesis, NTNU.

Chapman, S. (2003). *Fortran 90/95 for Scientists and Engineers.* McGraw Hill Higher Education, second edition edition.

Cook, R. D., Malkus, D. S., Plesha, M. E., and Witt, R. J. (2002). *Concepts and Applications of Finite Element Analysis*. JOHN WILEY & SONS, INC, fourth edition edition.

Grimstad, G., Andresen, L., and Jostad, H. P. (2012). *User Manual for equivalent cyclic ADP model for undrained behavior - "UDCAM"*. Norwegian Geotechnical Institute.

Grimstad, G. and Benz, T. (2016). *Implementation of soil models*. NTNU.

Gundersen, A. S. and Josefsen, J.-M. (2016). Modelling of undrained clay subjected to cyclic loading, semi-explicit material model. Master's thesis, NTNU.

Hopperstand, O. S. and Børvik, T. (2015). *Lecture Notes- Materials Mechanics*. NTNU - Structural Impact Laboratory.

Horstmann, C. S. (2013). *Big Java: Early Objects*. John Wiley & Sons Inc, fifth edition edition.

Jostad, H. P., Grimstad, G., Andersen, K. H., Saue, M., Shin, Y., and D.You (2014). A fe procedure for foundation design of offshore structures - applied to study a potential owt monopile foundation in the korean western sea. *Geotechnical Engineering Journal of the SEAGS & AGSSEA*, 45(4):63–72.

Jostad, H. P., Grimstad, G., Andersen, K. H., and Sivasithamparam, N. (2015). A fe procedure for calculation of cyclic behaviour of offshore foundations under partly drained conditions. In Meyer, V., editor, *Frontiers in Offshore Geotechnics III*, volume 1, pages 153–172.

Juspi, S. (2007). *Experimental Validation of the Shakedown Concept for Pavement Analysis and Design*. PhD thesis, The University of Nottingham.

Kreyszig, E. (2011). *Advanced Engineering Mathematics*. John Wiley and Sons Ltd.

Matlock, H. (1970). Correlation for design of laterally loaded piles in soft clay. In *Offshore Technology Conference*.

Metcalf, M., Reid, J., and Cohen, M. (2004). *Fortran 95/2003 Explained (Numerical Mathematics and Scientific Computation)*. Oxford University Press.

Nordal, S. (2014). *Soil modelling*. NTNU - Geotechnical Division.

RISUEÑO, A. M. P., JOSTAD, H. P., and SAUE, M. (2013). APPLICATION OF AN UNDRAINED AND A PARTIALLY DRAINED CYCLIC ACCUMULATION MODEL FOR MONOPILE DESIGN. In *Proceedings of the 5th International Young Geotechnical Engineers' Conference*.

Rønningen, J. A. (2017). *GEOFUTURE SOFT CLAY MODEL USER MANUAL*.

Rønningen, J. A., Gavel-Solberg, V., and Grimstad, G. (2014). Effective stress model for soft scandinavian clays. In *European Young Geotechnical Engineers Conference*.

Roscoe, K. and Burland, J. B. (1968). On the generalized stress-strain behavior of wet clays. In *Engineering plasticity*.

Wood, D. M. (1991). *Soil Behaviour and Critical State Soil Mechanics*. CAMBRIDGE UNIVERSITY PRESS.

Yin, Z. Y. and Karstunen, M. (2008). Influence of anisotropy, destructuration and viscosity on the behavior of an embankment of soft clay. In *International Association for Computer Methods and Advances in Geomechanics*, pages 4728–4735.