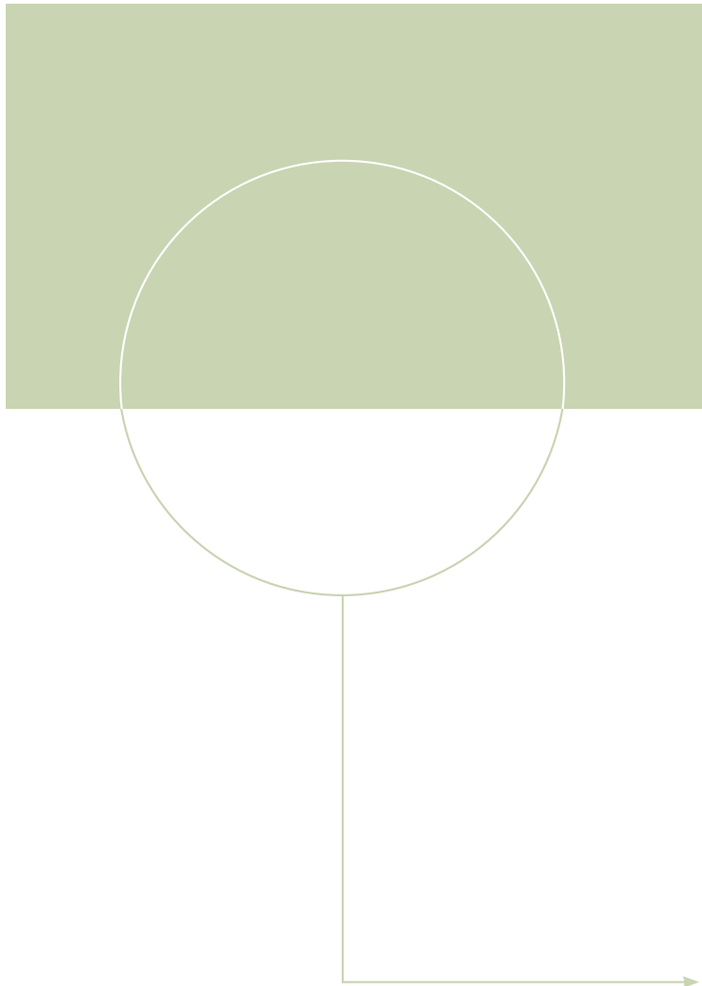


Master's thesis

2021

Dina Erika Karlsen Hansen

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

Dina Erika Karlsen Hansen

Automatisering av spuntberegninger

Automatisering av Plaxis-analyser med Python

February 2021

NTNU
Norwegian University of
Science and Technology

NTNU
Norwegian University of
Science and Technology

Dina Erika Karlsen Hansen

Automatisering av spuntberegninger

Automatisering av Plaxis-analyser med Python

Trondheim, Februar 2021

Masteroppgave: TBA4900

Hovedveileder: Gudmund Reidar Eiksund, NTNU

Medveileder: Viktor Renström, Norconsult

Institutt for Bygg- og miljøteknikk

Norges teknisk-naturvitenskapelige universitet (NTNU)

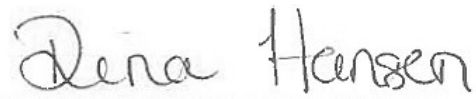
NTNU – Trondheim
Norwegian University of
Science and Technology

Forord

Denne oppgaven utgjør emnet emnet TBA4900 Geoteknikk, masteroppgave ved NTNU. Oppgaven er skrevet høstsemesteret 2020 ved Institutt for bygg- og miljøteknikk. Oppgaven ble foreslått av Viktor Renström i Norconsult i Bodø.

Jeg ønsker å takke veilederne mine, Gudmund Reidar Eiksund (NTNU) og Viktor Renström (Norconsult), for god oppfølging og veiledning i prosjektoppgaven. Jeg ønsker også å takke geoteknikerne Michael Huber (ERA Geo), Jung Chan Choi (NGI) og Jeroen Hermans (SWECO Nederland) som har har delt deres erfaring med Plaxis remote scripting gjennom epost-utveksling og digitale møter.

Trondheim, 5. februar 2021



Dina Hansen

Sammendrag

Plaxis remote scripting er et programmeringsgrensesnitt som gjør det mulig å styre Plaxis Input og Output ved hjelp av standardiserte kommandoer i programmeringsspråket Python. Når Plaxis styres med Python-kommandoer kan analyser i programmet automatiseres, noe som åpner nye muligheter for optimering og automatisering av repetitive oppgaver i programmet. Hovedmålet for denne masteroppgaven er å utarbeide et Python script som er et steg mot å automatisere dimensjonering av spuntvegger i Plaxis. For å lage et slik script må dimensjonering av en spuntvegg brytes ned i mindre steg som kan automatiseres med Python-kommandoer.

I starten av oppgaven presenteres prinsipper for spuntdimensjonering. Deretter følger et kapittel som omhandler Python og sentrale konsepter i programmeringsspråket. Videre blir fordeler, ulemper og bruksområder for Plaxis remote scripting belyst, og det gis en innføring i ulike hjelpemidler som kan brukes for å skrive Python-kommandoer til Plaxis.

Hoveddelen i oppgaven er en beskrivelse av scriptet i oppgaven. Scriptet kjører spuntanalyser i Plaxis med varierende jordmaterialer, gravedybder og spuntprofiler ved hjelp av en nøstet for-løkke. Når scriptet kjøres begynner for-løkka. En modell med en utkraget spuntvegg og beregningsfaser blir opprettet, og beregninger for modellen utføres. Dersom kravet til sikkerhetsfaktor er oppfylt reduseres spuntlengden med en meter, og beregningene utføres på nytt. Når spuntveggen ikke lenger oppfyller krav til sikkerhetsfaktor blir det satt inn et løsmassestag i modellen. Analysen for spuntvegg med stag blir så utført, og spuntlengden reduseres med en meter dersom sikkerhetsfaktoren tillater det. Når spuntkonstruksjonen ikke lenger oppfyller krav til sikkerhetsfaktor avsluttes analysen. Når scriptet er kjørt skrives resultater til et regneark i Microsoft Excel.

Python-koden er delt inn i ulike script- og modulfiler for å gjøre den oversiktlig, og for å gjøre det lettere å gjenbruke funksjoner i modulene. For å teste scriptet var det nødvendig å velge input-parametere for jordmaterialer, spuntprofiler og gravedybde. Scriptet ble kjørt med tre ulike Hardening Soil-materialer, seks spuntprofiler og syv gravedybder. Scriptet ble kjørt uten problemer med disse variasjonene, men dersom input-data i analysene endres kan det bli nødvendig å tilpasse koden.

Scriptet i oppgaven har flere begrensninger, og mye arbeid gjenstår før scriptet kan brukes til

spuntdimensjonering. Forslag til videre arbeid med scriptet er presentert i slutten av oppgaven.

Summary

Plaxis remote scripting is an application programming interface (API) that allows the user to control Plaxis Input and Output with Python commands. This makes it possible to automate analyses in Plaxis creating opportunities to optimize designs and automate repetitive tasks. In this thesis a Python script was developed which executes calculations for sheet pile walls in Plaxis. This script is a step towards automating the design of sheet pile walls in Plaxis.

Principles for designing sheet pile walls are presented in the beginning of the thesis. Then follows a chapter about the programming language Python, and an introduction to some important concepts in Python. Then pros, cons, and applications of Plaxis remote scripting are presented along with a list of tools that can be used when writing Python commands for Plaxis.

The main goal of this thesis is to develop a Python script which optimizes the design of sheet pile walls. This script divides the process of designing a sheet pile wall into steps that can be automated with Python commands. The code is divided into several Python files in order to give the code a good structure, and to make it easier to reuse functions that are defined within the modules. The results from the Plaxis analyses that are run by the script are written to Excel-files after the script is executed.

A nested for-loop in the script is used to run sheet pile wall calculations with different soil materials, excavation depths and sheet pile wall profiles. When the execution of the script starts the for-loop begins, and an analysis of a sheet pile wall without anchor will be run. If the specified safety factor is fulfilled, the length of the wall will be shortened by one meter, and the phases will be calculated again. This continues until the safety factor does not fulfil the requirement and a fixed end anchor will be added to the model. New phases will be created and calculated. The wall will be shortened by one meter if the safety factor meets the requirement and the phases will be calculated again. When the sheet pile wall no longer meets the required safety factor the analysis will end.

To test the script during development, it was necessary to determine input-parameters for materials and geometry for the model. The script is tested with three different Hardening Soil materials, six types of sheet profiles and excavation depths between four and ten meters. The script does not necessarily work for all types of materials, and unforeseen problems can occur if

input-data in the script is changed.

The script has several limitations and there is work to be done to make the script more robust before the script can be used when designing sheet pile walls. Further work with the script is described at the end of the thesis.

Innhold

Figurer

Tabeller

Kapittel 1

Introduksjon

1.1 Bakgrunn

Numerisk modellering og datasimulering er essensielle verktøy i geoteknisk dimensjonering og analyse. Formålet med modellering er å forutsi jordas oppførsel i ulike geotekniske problemer ved hjelp av matematiske modeller og analyseprogrammer (Rahman & Ulker, 2018). I geoteknikk blir numerisk modellering blant annet benyttet til beregning av jordtrykk, deformasjoner og stabilitet. Det finnes en rekke analyseprogrammer som kan benyttes til simulering i geoteknikk, og Plaxis er et av dem. I Plaxis utføres beregningene ved hjelp av elementmetoden.

Plaxis kan blant annet brukes til dimensjonering av spuntvegger. Spuntanalyser har ofte mange beregningsfaser der det er ønskelig å hente ut krefter, momenter og deformasjon i ulike faser. Det kan være tidkrevende å gå igjennom og lagre resultater fra alle beregningsfasene, og sannsynligheten for inntastingsfeil øker når brukeren må utføre mye manuelt arbeid i det grafiske brukergrensesnittet (GUI) i Plaxis. Python scripting kan brukes til å automatisere slike repetitive arbeidsoppgaver i Plaxis. På denne måten kan brukeren spare tid og sannsynligheten for inntastingsfeil reduseres.

Når en spuntvegg skal dimensjoneres er det er det ofte nødvendig å utføre iterative analyser der brukeren oppdaterer input-parametere i modellen inntil én eller flere betingelser er tilfredsstilt. Python tilbyr ulike verktøy som kan brukes til å automatisere slike iterative beregninger. Ved å bruke scripting kan altså iterative beregninger utføres i Plaxis uten at det krever brukerens oppmerksomhet.

1.2 Problemformulering

Hovedmålet for denne oppgaven er å lage et Python script som skal være et steg mot en automatisering av dimensjonering av spuntvegger i Plaxis. Følgende forskningsspørsmål vil belyses i oppgaven:

- Hvilke verktøy i Python kan brukes til å automatisere spuntberegninger i Plaxis?
- Hvordan kan tradisjonell dimensjonering av spuntvegg brytes ned i mindre steg og automatiseres ved hjelp av et Python script?
- Hvilke utfordringer kan oppstå når man automatiserer Plaxis-analyser med Python?

1.3 Avgrensning

Masteroppgaven gjennomføres i løpet av en kort tidsperiode. For å avgrense oppgaven er det det satt en rekke begrensninger for Plaxis-modellen som genereres med scriptet:

- Spuntveggen som modelleres er en svevespunt.
- Spuntveggen har ett avstivningsnivå.
- Ankeret er horisontalt og har den samme plasseringen i alle modellene som genereres.
- Grunnvannstanden er konstant og er satt på samme kotenivå som bunnen av utgravingen i modellen.
- Scriptet er testet for utgravinger med fire til ti meters gravedybde, og kun heltall er testet.
- Scriptet er testet for tre jordmaterialer og fem spuntprofiler.
- Det stilles ikke krav til deformasjoner og maksimalt moment i spuntveggen i scriptet.

1.4 Rapportens oppbygning

Masteroppgaven er delt inn i fem hoveddeler, inkludert denne innledningen. I tillegg kommer vedleggene. Kapittel 2 omhandler spuntvegger og prinsipper for dimensjonering. Kapittel 3 gir en introduksjon til Plaxis, Python og Plaxis remote scripting. I denne delen presenteres fordeler, ulemper og bruksområder for Plaxis remote scripting, samt nyttige hjelpemidler for å skrive

Python-kommandoer med korrekt syntaks. Deler av kapittel 3 er gjenbrukt fra forstudiet til oppgaven, som ble skrevet våren 2020. Kapittel 4 gir en beskrivelse av scriptet som er utarbeidet i oppgaven. Her beskrives også Plaxis-modellen som genereres, resultater fra analysene og feilmeldinger som kan oppstå underveis i arbeidet med et Python script. I Kapittel 5 oppsummeres oppgaven og forskningsspørsmålene besvares. Til slutt presenteres forslag til videre arbeid med scriptet. Masteroppgaven har fem vedlegg i tillegg til en zip-mappe med filer som brukes til å kjøre scriptet som er utarbeidet i oppgaven. Vedlegg C og B er gjenbrukt fra forstudiet til masteroppgaven.

Kapittel 2

Spuntvegg

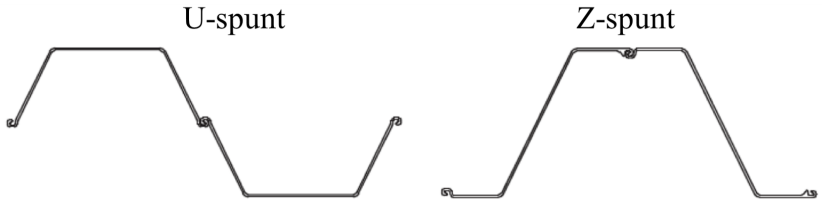
2.1 Generelt

Spuntvegger er permanente eller midlertidige støttekonstruksjoner som har som formål å ta opp jordtrykket som kan oppstå ved utgraving og tilbakefylling av jord. Spuntvegger brukes ofte i byggegroper for å sikre nabokonstruksjoner og for å hindre vanninntrengning (Statens vegvesen, 2010b), og kan tas i bruk både på land og i sjøen. Figur 2.1 viser et bilde av en spuntvegg fra bygging av Follobanen i 2017.

Figur 2.1: Bildet viser en spuntvegg som ble konstruert i forbindelsen med bygging av Follobanen. Her jobbes det med forankring av spuntveggen(Bane NOR, 2017).

Spuntvegger kan bestå av stål, betong eller tre. Stål er mest benyttet, og spuntvegger av tre

benyttes vanligvis ikke. Spuntvegger av stål består av valsede stålprofiler som kalles spuntnåler. U-spunt med lås i nøytralaksen og Z-spunt er hovedtypene spuntnåler (se figur 2.2), og det er også mulig å benytte sammensatte profiler. Et eksempel på dette er en kombinasjon av H-bjelker og Z-spunt, som benyttes når det stilles et høyt krav til motstands- og treghetsmoment.

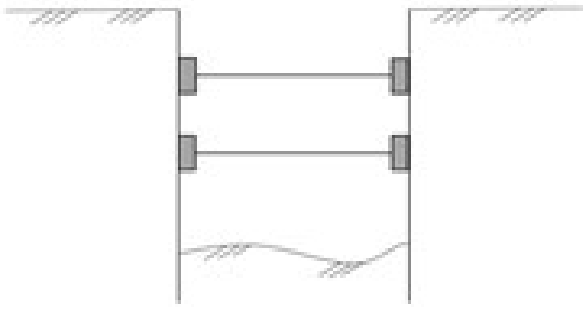


Figur 2.2: U-spunt og Z-spunt er hovedtypene spuntnåler (Standard Norge, 1999).

2.2 Avstivning

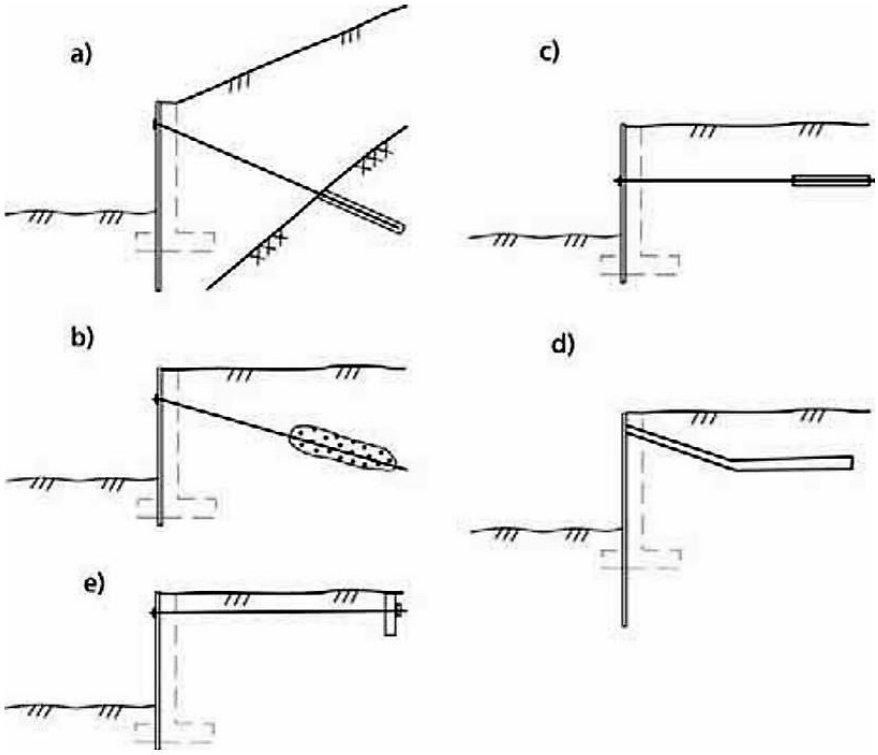
Det er ofte behov for å stive av en spuntvegg for at den skal oppfylle sin funksjon og tilfredsstille de kraven den dimensjoneres for (Statens vegvesen, 2018). Når massene foran en spuntvegg graves ut vil spuntveggen oppleve aktivt og passivt jordtrykk, og det vil oppstå et moment i spuntveggen som følge av jordtrykkene. Momentet i spuntveggen kan medføre horisontalforskyvninger i spuntveggen som resulterer i deformasjoner i bakkant av spuntveggen. Deformasjoner i jorda kan skape setningsproblemer for eksisterende bebyggelse i nærheten av spuntveggen. Deformasjoner i bakkant av spuntveggen kan begrenses ved å stive av spuntveggen.

Avstivning av spuntvegg kan utføres med innvendig avstivning eller med forankring. Innvendig avstivning skjer på gravesiden av spuntveggen, og består vanligvis av stålbjelker, samt støpte betongdekker til tak og/eller gulv. Innvendig avstivning gir mindre deformasjoner sammenlignet med stagforankring, da stålbjelkene har stor stivhet og motstår deformasjoner i større grad enn stag (Grande, 1998). Et problem med innvendig avstivning er at avstivningsmediet kan komme i konflikt med anleggsmaskiner eller konstruksjonen som skal bygges i byggegropa. Figur 2.3 viser prinsippet for innvendig avstivning.



Figur 2.3: Innvendig avstivet spunt. Avstivning skjer på gravesiden av spuntveggen. Avstivningsmediet er vanligvis stålbjelker samt støpte betongdekker til tak og/eller gulv.

Forankring med stag er den foretrukne avstivningsmetoden for større utgravinger og for permanente konstruksjoner. Byggegroper som stives av med forankringsstag blir åpne og lett tilgjengelige (Arne Engen et al., 2016). Figur 2.4 viser ulike typer forankringer.



Figur 2.4: Ulike typer forankringer: a) bergstag, b) løsmassestag, c) horisontal friksjonsplate, d) friksjonsplate, e) vertikal forankringsvegg. Figuren er hentet fra Håndbok V220 (Statens vegvesen, 2018).

Når fast fjell er utenfor rimelig rekkevidde benyttes løsmassestag. Løsmassestag er dyrere enn forankring til fjell på grunn av usikkerhet knyttet til beregning av holdekapasiteten og fordi

installasjonen av løsmassestag er ressurskrevende (Grande, 1998). Løsmassestag kan utføres med injisering eller med en fast ekspanderende kropp, som dimensjoneres etter samme prinsipper. Forankringssonen må ligge godt utenfor aktiv sone bak veggen. Staget kan bestå av en stålvaier eller stålstenger (lisser) som bores ned og injiseres fast i løsmasser.

2.3 Prinsipper for spuntdimensjonering

2.3.1 Regelverk

Følgende dokumenter utgjør vanligvis regelverket som danner grunnlag for spuntdimensjonering (Statens vegvesen, 2018):

- NS-EN 1990:2002+A1:2005+NA:2016 Eurokode: "Grunnlag for prosjektering av konstruksjoner" (Standard Norge, 2016a).
- NS-EN 1997-1:2004+A1:2013+NA:2016 Eurokode 7: "Geoteknisk prosjektering, Del 1: Allmenne regler" (heretter kalt Eurokode 7 - del 1) (Standard Norge, 2016b)
- NS-EN 1997-2:2007+NA:2008 Eurokode 7: "Geoteknisk prosjektering, Del 2: Regler basert på grunnundersøkelser og laboratorieprøver"(heretter kalt Eurokode 7 - del 2) (Standard Norge, 2008b)
- NS-EN 1998-5:2004+NA:2014 Eurokode 8: Prosjektering av konstruksjoner for seismisk påvirkning. Del 5: Fundamenter, støttekonstruksjoner og geotekniske forhold (Standard Norge, 2008a)
- Håndbok V220: Geoteknikk i vegbygging (Statens vegvesen, 2018)

2.3.2 Kontroll av spuntvegg i grensetilstander

Det er krav om at spuntkonstruksjoner og eventuelle avstivnings- og forankringselementer skal kontrolleres i brudd- og bruksgrensetilstander. Dersom det er relevant skal også ulykkestilstander og seismisk påvirking vurderes. Konstruksjonselementene skal dimensjoneres for den mest kritiske grensetilstanden. En konstruksjonsdel har nådd grensetilstanden når den ikke lenger oppfyller sin funksjon eller tilfredsstiller kravene den er dimensjonert for.

Bruddgrensetilstand

Eurokode 7 - del 1 setter krav tik kontroll av bruddgrensetilstander. Der det er relevant skal spuntkonstruksjoner kontrolleres for følgende bruddgrensetilstander:

- Tap av likevekt i grunnen eller i konstruksjonen (EQU = equilibrium").
- Intern svikt eller for stor deformasjon i konstruksjonen (STR = structural")
- Svikt eller for stor deformasjon i grunnen (GEO = geotechnical")
- Tap av likevekt pga. oppløft eller oppdrift (UPL ="uplift").
- Hydraulisk grunnbrudd (HYD = "hydraulic")

Bruksgrensetilstand

Bruksgrensetilstanden skal analyseres for å påvise at konstruksjonen ikke utsettes for forskyvning, deformasjon, erosjon eller annen nedbrytning som medfører at den ikke fungerer som forutsatt. Bruksgrensen for spuntvegger bestemmes vanligvis av krav til utbøying av spuntvegg og deformasjon i bakkant av spuntveggen.

Kontroll av grensetilstander

For å verifisere at grensetilstandene ikke overskrides benyttes en av følgende metoder eller en kombinasjon av dem (Fredriksson, Kullingsjö, Rynder & Stille, 2018):

- Erfaringsdata
- Beregning
- Prøvebelastning og modellforsøk
- Observasjonsmetode

Vanligvis benyttes beregning som verifisering av sikkerhet mot grensetilstandene. Eurokoden gir ikke konkrete krav til beregningsmetode for å kontrollere grensetilstander for spuntkonstruksjoner, men det kreves at beregningene er ”i tråd med god praksis”. Beregningsmetoder deles inn i analytiske metoder, semi-empiriske metoder og numeriske metoder.

2.3.3 Analytiske metoder: Håndberegninger

Analytiske beregningsmetoder, eller håndberegning, er basert på tradisjonell grenselikevekt. I tradisjonell grenselikevekt betrakter man en grensetilstand der et bruddkriterium er nådd. I spuntberegniner er dette en bruddtilstand der aktivt og passivt grensejordtrykk er nådd og spenningsfeltene er plastiske. Videre antar man at jordtrykksfordelingen kan bestemmes ved å gange en en faktor (κ_A, κ_P) med overlagringstrykket. Spuntveggen betraktes som uendelig stiv. Håndberegninger kan ikke brukes til å vurdere deformasjoner i bruksgrensetilstand. Slike vurderinger krever numeriske beregninger, erfaringsdata eller lignende.

2.3.4 Numeriske metoder: samvirkeprogrammer

Samvirkeprogrammer tar høyder for samvirke mellom spuntvegg og jord. I realiteten er beregning av jordtrykk et samvirkeproblem der det er et samvirke mellom spuntvegg og omkringliggende løsmasser. En myk spunt vil i større grad gi etter for jordtrykk og tillate mer deformasjon enn en stiv spunt. Dette resultere i et mindre moment i spunten og et mindre jordtrykk. Motsatt vil en stivere spunt gi mindre utbøying og mindre deformasjoner. Spuntveggen vil i dette tilfellet være påkjent av et større moment og større jordtrykk sammenlignet med en mykere spuntvegg.

I motsetning til håndberegnigner, kan brudd- og bruksgrensetilstand kontrolleres i den samme analysen i samvirkeprogrammer. I samvirkeprogrammer benyttes karakteristiske verdier for stivhet, materialfasthet og laster for å oppnå realistiske deformasjoner i bruksgrensetilstand. Bruddgrensetilstanden kontrolleres ved å redusere styrken til materialet inntil brudd er oppnådd, for deretter å finne sikkerhetsmargin mot brudd. Det er anbefalt å bruke håndberegning til å kontrollere resultater fra samvirkeprogrammer dersom det er mulig.

Det er vanlig å skille mellom to typer samvirkeprogrammer som benyttes i geoteknikk (Fredriksson et al., 2018):

- Programmer som er basert på fjærmodell, for eksempel GeoSuite. Spuntveggen modelleres som en bjelke med horisontale fjær.
- Programmer som er basert på elementmetoden for eksempel PLAXIS.

I norske bedrifter blir både GeoSuite og PLAXIS benyttet til dimensjonering av spuntvegger.

Kapittel 3

Automatisering av Plaxis med Python

3.1 Introduksjon til Plaxis

PLAXIS er et elementmetodeprogram som benyttes i geotekniske analyser. Utviklingen av Plaxis startet på Delft University i Nederland i 1987, og programmet har siden den gang blir videre utviklet gjennom et samarbeid mellom en rekke universiteter og geotekniske fagmiljøer fra hele verden, også fra NTNU (Brinkgreve, Kumarswamy, Swolfs & Foria, 2018a). I 2018 ble Plaxis kjøpt opp av det amerikanske IT-selskapet Bentley Systems.

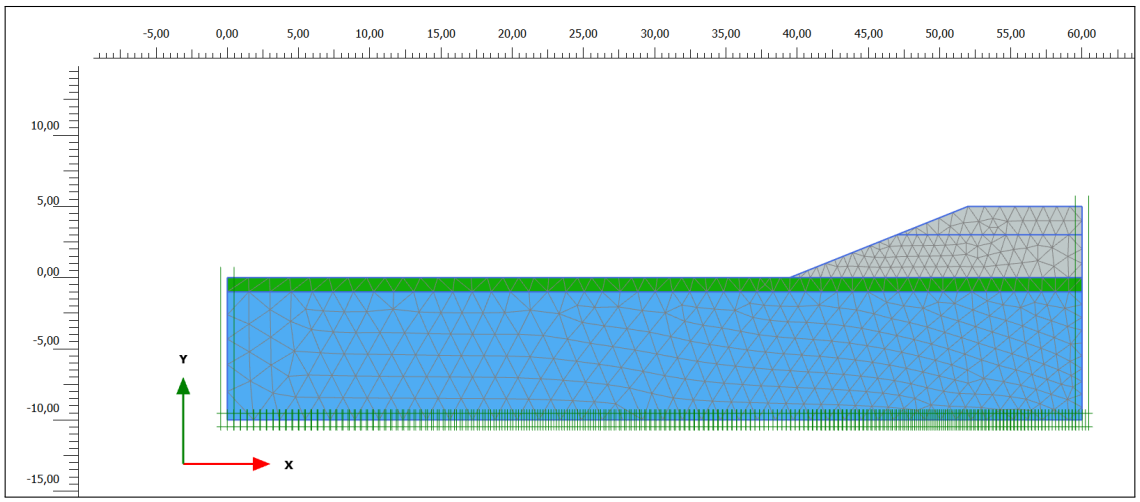
Da utviklingen av FEM-programmer innen geoteknikk startet var programvaren forbeholdt FEM-spesialister. De siste tiårene har FEM-programmer blitt videre utviklet, og mange av dem fungerer i dag som en svart boks (”black box”), som vil si at brukeren kan utføre analyser i programmet uten nødvendigvis å forstå hvordan beregningene utføres. Brukeren velger inngangsparametere og får ut resultater, som illustrert i figur 3.1. Dette gjelder også Plaxis. Brukergrensesnittet i Plaxis er utformet slik at programmet kan brukes på en intuitiv måte, og det er ikke nødvendig med en dyp forståelse av elementmetoden for å utføre analyser ved hjelp av programvaren (Nordal, 2019).

Figur 3.1: FEM-programmet Plaxis fungerer som en ”black box” der brukeren velger inngangsparametere og får ut resultater.

Plaxis brukes til å utføre beregninger for deformasjoner, spenninger, kapasitet, konsolidering og så videre. Det finnes både en todimensjonal versjon (Plaxis 2D) og en tredimensjonal versjon (Plaxis 3D) av Plaxis. I PLAXIS 2D utføres beregningene med en aksesymmetrisk modell (axisymmetric model) eller med plan tøyning (plane strain). Dersom man antar plan tøyning settes tøyning ut av planet lik null ($\epsilon_z = 0$). I mange geotekniske problemstillinger er dette en rimelig antagelse. Beregninger i PLAXIS 3D er ikke basert på denne antagelsen, og kan derfor gi mer realistiske resultater (Mykleset, 2018). Det tar imidlertid lenger tid å kjøre analyser i Plaxis 3D sammenlignet med Plaxis 2D.

3.1.1 Elementmetoden i Plaxis 2D

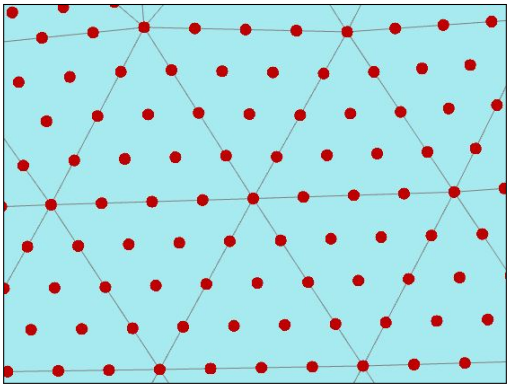
Elementmetoden (FEM) er en numerisk metode som går ut på å løse problemstillinger som kan beskrives av partielle differensialligninger eller integralligninger. Elementmetoden går ut på å dele inn et kontinuum (for eksempel et jordvolum) inn i mindre (endelige) elementer som er knyttet sammen gjennom knutepunkter (noder) (Statens vegvesen, 2010a). Elementene utgjør et elementnett. Figur 3.2 viser et elementnett som er generert i en modell i Plaxis 2D.



Figur 3.2: Eksempel på et elementnett som er generert i Plaxis 2D. Modellen er en symmetrisk veifylling som er konstruert på et jordvolum bestående av tørrskorpe og leire.

I Plaxis 2D har hvert knutepunkt 2 frihetsgrader, da spenning og deformasjoner kan opptre i x- og y-retning i knutepunktene. Det er gjort en antagelse om at deformasjon i et element kan beskrives med et sett av ligninger, én for hver frihetsgrad. Ligningssystemet for hele elementnettet har altså like mange ukjente som to ganger antall knutepunkter, og må løses slik at problemet tilfredsstiller randbetingelsene (Nordal, 2019).

Elementmetoden er en numerisk metode, og beregninger i Plaxis gir derfor approksimerte løsninger. Nøyaktigheten i Plaxis avhenger av forenklinger i selve modellen, samt innstillinger som er valgt i Plaxis. Forenklinger og antagelser i modellen kan være knyttet til geometri, grunnforhold, grensebetingelser og så videre. Innstillinger i Plaxis som har innvirkning på nøyaktigheten i beregningene er blant annet valg av elementtype, elementnett og antall knutepunkter per element. Riktig bruk av interface-elementer vil også kunne påvirke nøyaktigheten i beregningene. Et fint elementnett gir bedre nøyaktighet enn et grovt elementnett, og mange knutepunkter gir bedre nøyaktighet enn færre knutepunkter. Det tar imidlertid lenger tid å kjøre analyser med et fint nett og mange knutepunkter. I Plaxis 2D er det vanlig å bruke trekantformede elementer med 15 knutepunkter (15-Noded Elements) for å oppnå tilstrekkelig nøyaktighet i beregningene. Figur 3.3 viser et utsnitt av et jordvolum i Plaxis 2D som er delt inn i elementer med 15 knutepunkter.



Figur 3.3: Utsnitt fra en modell i Plaxis 2D der jordvolumet er delt inn i trekantformede elementer med 15 knutepunkter per element.

Plaxis-versjonen som benyttes kan også ha innvirkning på nøyaktigheten i beregningene, og man må vurdere om det er hensiktsmessig å bruke Plaxis 2D eller Plaxis 3D. For å kvalitetssikre beregninger i Plaxis er det anbefalt å kontrollere resultater med håndberegninger dersom det er mulig.

3.1.2 Materialmodeller i Plaxis

I Plaxis finnes det en rekke ulike materialmodeller som benyttes til å modellere jorda. Materialmodell velges på grunnlag av av jordarten som skal modelleres, nødvendig nøyaktighet i analysen, type belastning og geoteknisk problemstilling. Materialmodellene har ulike inputparametere som beskriver jordoppførselen. Mohr Coulomb-modellen (MC) er en forenklet materialmodell som hovedsaklig benyttes til å gjøre beregningsoverslag og for å ha et utgangspunkt når en geoteknisk problemstilling skal analyseres. Når det er behov en mer nøyaktig modellering av jorda benyttes en av de mer avanserte materialmodellene i Plaxis.

3.2 Introduksjon til Python

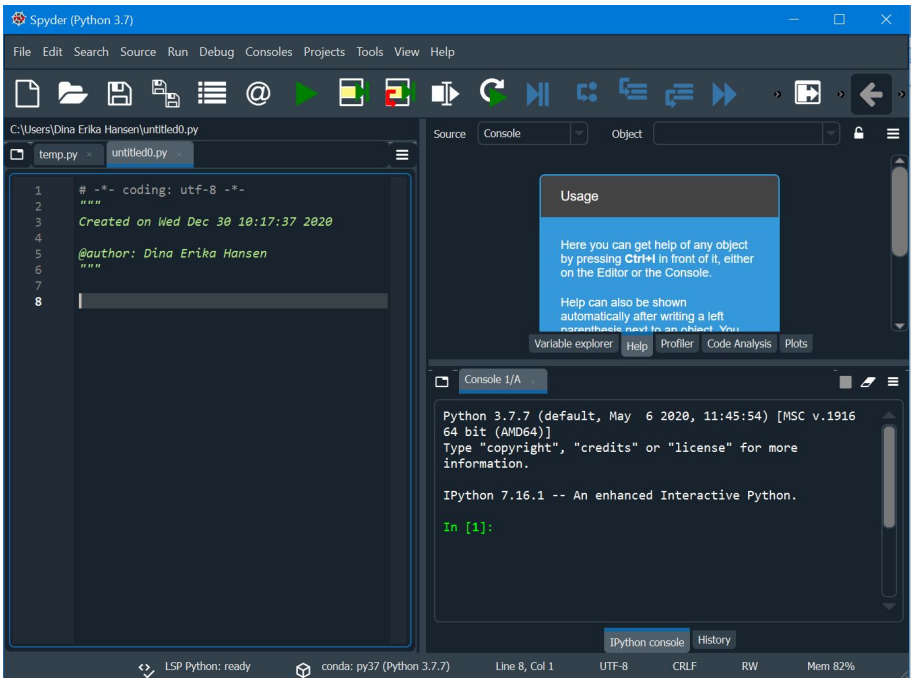
Python er et objektorientert programmeringsspråk som ble utviklet av Guido van Rossum på starten av 1990-tallet, og er i dag et av verdens raskest voksende programmeringsspråk (Lekanger, 2019). Python benyttes blant annet til dataanalyse, beregninger, maskinlæring og fullstack webutvikling. Det finnes flere versjoner av Python, og i denne oppgaven ble Python 3.7-versjonen benyttet.

Det er flere grunner til at Python er et populært programmeringsspråk innen vitenskapelige beregninger og dataanalyse. Python er basert på intuitiv koding som er lettlest og lett å skrive sammenlignet med andre programmeringsspråk (Gerhardsen & Guldstad, 2020). Dessuten er Python en open source programvare, som betyr at det er helt gratis å bruke og å lage programmer av Python (Python Software Foundation, 2020a). Andre programmeringsspråk, som for eksempel Java EE, krever lisensbetaling for å bruke det. Python er et universelt programmeringsspråk som kan brukes til integrasjon av programmer, også på tvers av fagfelt.

Det er anbefalt å ha grunnleggende ferdigheter i Python for å lage og kjøre scripts i Plaxis. De neste underkapitlene gir en kort introduksjon til konsepter i Python som er sentrale i denne oppgaven.

3.2.1 Integrert utviklingsmiljø (IDE) og kodeeditor

For å kunne lage et Python script trengs det en programvare der koden skrives. Denne programvaren kan være en kodeeditor eller et integrert utviklingsmiljø (IDE). Både IDE'er og kodeeditorer tilbyr skriving og redigering av kode. IDE'er er ofte å fortetrekke fordi de også tilbyr andre funksjonaliteter som kompilering, debugging, kjøring av kode, autofullføring og biblioteker. Det finnes et stort antall IDE'er som kan brukes til å skrive Python, og IDE burde velges på grunnlag av av operativsystem på datamaskinen, formål med kodingen, brukerens tidligere erfaring med koding og så videre. Eksempler på noen IDE'er for Python er PyCharm, Atom, Idle og Spyder. I denne oppgaven har Spyder 4 brukt som IDE. Figur 3.4 viser vinduet man får opp når man åpner IDE'en Spyder 4 med Python-versjon 3.7.



Figur 3.4: Når man åpner Spyder 4 vil man få opp et vindu som ser omtrent ut som dette. Øverst i venstre hjørne står Python-versjonen som blir benyttet, som i dette tilfellet er Python 3.7.

3.2.2 Script, modul, pakker og biblioteker i Python

Det er verdt å merke seg at begrepene script og modul ikke har strenge definisjoner, og er til en viss grad åpne for tolkning (Cepalia, 2021). Definisjonene som er presentert her er benyttet i denne oppgaven. Et script og en modul er begge filer som inneholder Python-kode og som slutter med .py. Scripts inneholder ofte kode utenfor funksjoner, og som kjøres uten at det er nødvendig å kalle på den. Moduler består i prinsippet av samlinger med funksjoner og klasser som kan importeres og tas i bruk i et script. I motsetning til scripts, utfører moduler vanligvis ikke noe når de kjøres direkte. Moduler kan sammenlignes med byggeblokker som trengs for å lage et system, og scriptet er en måte å kjøre byggeblokkene. I denne oppgaven er det utarbeidet et script og flere moduler. Når det står "kjøre script" i denne oppgaven, betyr dette at scriptet **script.py** kjøres og at modulene i oppgaven blir brukt i kjøringen av scriptet, selv modulene ikke kjøres direkte.

Pakker er samlinger av moduler og/eller andre pakker, og er ment for spesielle formål i Python (Haugen, 2019). Mange pakker blir automatisk lastet når Python lastes ned, mens andre pakker kan lastes inn ved behov. Et eksempel på en pakke i Python er *matplotlib* som inneholder funksjoner for plotting av data. Et annet eksempel er *Pandas* som er en pakke for datamanipulering

og analyse. Bibliotek er bare et annet ord for pakke.

3.2.3 Datatyper

Alle verdier i Python har en datatype. Ulike funksjoner og metoder kan brukes for bestemte datatyper, og det er derfor nødvendig å vite forskjellen mellom datatyper. Datatyper som brukes i scriptet i oppgaven er presentert her:

Flyttall(float): Flyttall er det samme som desimaltall. For eksempel: 1.36

Heltall (int): For eksempel: 44

Tekststreng (str): For eksempel: 'Hello', '98'

Lister (list): Figur 3.5 viser et eksempel på en liste kalt "liste1". Denne listen har fire listeelementer: to heltall og to flyttall. Listeelementer kan også være tekststrenger. Listeelementer i Python indekseres ikke på samme måte som for eksempel i Matlab. I Python starter indekseringen på 0, som vist i figur 3.5.

```
In [7]: liste1=[2,7,9,3] #oppretter liste
   ...: print(liste1[0]) #skriver det første listeelementet i konsollen
2
```

Figur 3.5: Indeksering av listeelementer i Python

Dictionary (dict): Dictionary er en slags oppslagsliste der data lagres som nøkkel:verdi-par som vist i figur 3.6. Verdier aksesseres ved å bruke nøkkel-verdien.

```
In [13]:
    ...: eksempel_dict={ #oppretter dict
    ...:     'Navn': 'Dina',
    ...:     'Alder': 26,
    ...:     'Land':'Norge'
    ...:     }
    ...:
    ...: print(eksempel_dict['Navn'])
Dina
```

Figur 3.6: Eksempel på dictionary i Python. Verdien 'Dina' aksesseres ved å bruke nøkkelen 'Navn'.

3.2.4 Funksjoner

En funksjon er en seksjon med kode som utfører en spesifikk oppgave, og brukes til å dele programmer opp i mindre deler slik at koden blir lettere å forstå. Funksjoner gjør det også lettere å gjenbruke kode. Funksjoner benyttes til utføre regneoperasjoner, filbehandling, dataanalyse og så videre. Man kan ta i bruk eksisterende funksjoner andre har laget, eller man kan definere funksjoner selv. I Python finnes det en rekke innebygde funksjoner som er tilgjengelige til enhver tid, for eksempel abs(), sqrt(), print(). En oversikt over de innebygde funksjonene for ulike versjoner av Python finnes på nettstedet til Python Software Foundation (Python Software Foundation, 2020b). I tillegg til de innebygde funksjonene finnes det et stort antall pakker med funksjoner som kan lastes ned ved behov.

Figur 3.7 viser et eksempel på en funksjon. Funksjoner starter med et funksjonshode bestående av ordet ”def” og navnet på funksjonen, som i dette tilfellet er ”my_function”. Funksjonsnavn skrives vanligvis med små bokstaver og deles opp med understrek. Som man kan se av figuren skriver my_function tekststrengen ”Hello World” i konsollen når den blir kalt på.

```
   ....:
   ...: #Oppretter en funksjon som skriver "Hello World"
   ...: #i konsollen når den blir kalt på.
   ...: def my_function():
   ...:     print('Hello World')
   ...:
   ...: my_function() #Kaller på funksjonen
Hello World
```

Figur 3.7: Eksempel på en funksjon i Python. ”Hello World” blir skrevet til konsollen når funksjonen kalles på.

3.2.5 Metoder

Metoder ligner funksjoner, men metoder tilhører alltid et objekt og skrives med dot-notasjon. Figur 3.8 viser hvordan metodene append() og remove() brukes for en liste (objektet).

```
In [11]: liste1=[2,7,9,3] #oppretter liste
    ...: liste1.append(3) #legger til et listeelement
    ...: liste1.remove(9)
    ...: print(liste1)
[2, 7, 3, 3]
```

Figur 3.8: Metoden append() brukes for å legge til listeelementer i lister . Metoden remove() brukes til å fjerne listeelementer.

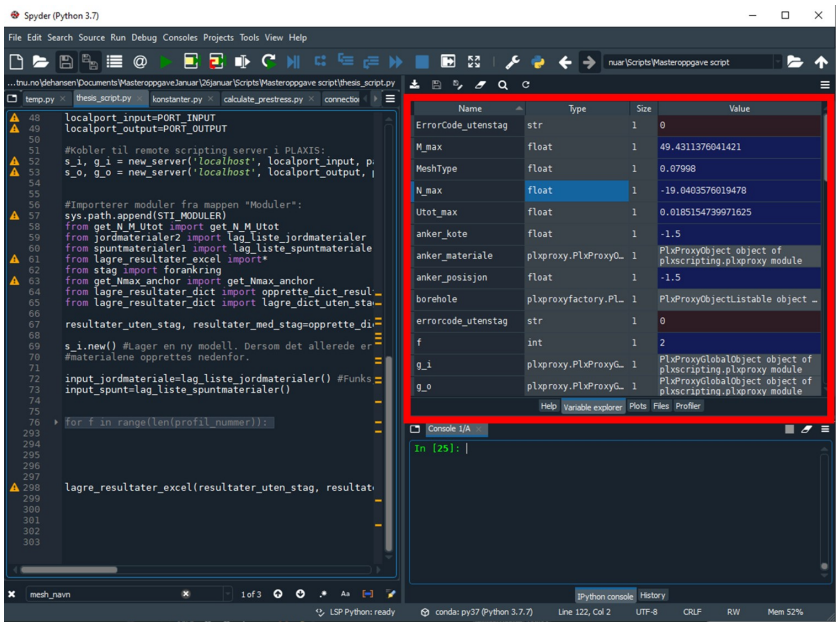
3.2.6 Variabler

Variabler er plassholdere for verdier som er lagret i datamaskinens minne. Verdien til en variabel kan endres i løpet av et program og variabler kan gjenbrukes flere ganger i koden slik at brukeren slipper å gjøre mellomregninger på nytt. En variabel opprettes ved å sette variabelnavn lik en verdi eller et uttrykk. Variabler må opprettes før de brukes i koden. Verdien eller uttrykket en variabel representerer kan være av ulike datatyper, som vist i figur 3.9. Datatypen til en variabel kan finnes ved å benytte funksjonen type().

```
In [27]: fylke = "Trøndelag" #variabelnavnet er "fylke" og
datatypen er tekststreng (str)
    ...: print(type(fylke))
    ...:
    ...: year = 2021 #variabelnavnet er "year" og datatypen
er heltall (int)
    ...: print(type(year))
    ...:
    ...: navn_liste = ["Lisa", "Mette", "Solveig"]
    ...: #variabelnavnet er "navn_liste" og datatypen er
liste (list)
    ...: print(type(navn_liste))
<class 'str'>
<class 'int'>
<class 'list'>
```

Figur 3.9: Eksempler på variabler i Python. For å finne datatypen til en variabel kan man bruke funksjonen type().

I likhet med funksjoner skrives variabelnavn vanligvis med små bokstaver og ordene skilles med understrek. Variabler som er konstante gjennom et program skrives vanligvis med store bokstaver. *Variable Explorer*-vinduet i Spyder gir en oversikt over variabler som er definert i programmet. Vinduet viser verdier og datatyper for variablene, se figur 3.10



Figur 3.10: *Variable Explorer* i Spyder gir en oversikt over variabler som er opprettet. Oversikten viser datatype og verdi for variablene, og oppdateres underveis når koden kjøres.

3.2.7 Kontrollstrukturer

Kontrollstrukturer brukes til å styre flyten i et program, og gjør det mulig å kjøre spesifikke deler av koden (Oracle, 2019). I scriptet i denne oppgaven kontrolleres programflyten av if/elif/else-setninger og løkker.

if-setninger brukes for å automatisere beslutningsprosesser i Python. Dersom en if-setning er True (boolsk verdi) kjøres kodeblokken i if-grenen. Dersom if-setningen er False kjøres ikke kodeblokken i if-grenen, og kjøring av koden som står under if-grenen fortsetter. if-setningen kan stå alene eller kan brukes i kombinasjon med elif og else.

Løkker benyttes i programmering for å repetere like eller nesten like handlinger (Sindre, 2019). Det finnes to typer løkker i Python: while-løkke og for-løkke. For-løkker kjøres et bestemt antall ganger, og while-løkker kjøres til en betingelse er oppfylt. Løkker brukes blant annet i optimering og for å løse ligninger iterativt (Haugen, 2019). Løkker kan være nøstede, det vil si at en løkke er inne i en annen løkke. Nøstede løkker blir brukt i scriptet i denne oppgaven.

3.2.8 Retningslinjer for kodestil i Python

Det finnes retningslinjer for hvordan kode i Python burde se ut. Kodestilen har ikke betydning for funksjonaliteten i programmet, men er viktig for å gjøre koden lesbar. Retningslinjene legger føringer for blant annet navn på funksjoner og variabler, antall tegn per linje, innrykk (indent) i koden og så videre. I denne oppgaven følges retningslinjer fra guiden *PEP 8 – Style Guide for Python Code* (van Rossum & Warsaw, 2013).

3.3 Plaxis remote scripting

Plaxis remote scripting er et programmeringsgrensesnitt, forkortet API, som muliggjør kommunikasjon mellom Plaxis og programmeringsspråket Python. Et programmeringsgrensesnitt er et grensesnitt i en programvare som gjør at spesifikke deler av denne kan aktiveres («kjøres») fra en annen programvare. Kommunikasjon mellom Plaxis og Python opprettes ved å sette opp en server i Plaxis og koble Python til denne (se vedlegg B og C). Kommunikasjonen mellom Python og Plaxis er kryptert for å sikre trygg overføring av informasjon mellom programmene (van der Sloot, 2019). Når Python er koblet til serveren i Plaxis, kan Input og Output i Plaxis styres ved hjelp av standardiserte kommandoer i et Python script. Plaxis remote scripting er implementert i Plaxis 2015 og nyere versjoner.

Programmeringsgrensesnittet er av typen HTTP REST API, som er en standard API-modell som benyttes i de fleste moderne web-tjenester. I denne oppgaven vil det ikke gåes i detalj rundt API i Plaxis, og mer informasjon om dette finnes i manualen *Plaxis 2D Reference Manual* (Brinkgreve, Kumarswamy, Swolfs & Foria, 2018b).

3.3.1 Fordeler og bruksområder

Automatisering av Plaxis er fordelaktig fordi brukeren slipper å utføre manuelle og repetitive oppgaver i programmet. Ved hjelp av scripting kan Plaxis-analyser effektiviseres, og sannsynlighet for inntastingsfeil i modellen reduseres. I Python finnes det en rekke funksjoner og pakker som er rettet mot beregninger og data-analyse (se underkapittel 3.2). Slike verktøy åpner nye muligheter i Plaxis for blant annet iterative beregninger, integrasjon av Plaxis og andre programmer og behandling av Output-data. Bruksområder, fordeler og ulemper for Plaxis remote scripting er nærmere beskrevet i de neste underkapitlene.

Analyser av repetitiv natur

Plaxis remote scripting egner seg for geotekniske analyser av repetitiv natur, det vil si analyser med lignende beregningsfaser, materialer og så videre (Bentley Systems, 2015b). Brukeren kan spare tid på å benytte Python scripting i analyser med lignende modeller fordi deler av scriptene kan gjenbrukes. Det er tidkrevende å lage et Python script den første gangen det skal kjøres, men scriptet kan være tidsbesparende dersom det kan gjenbrukes i flere analyser (Viktor Renström, geotekniker i Norconsult, personlig kommunikasjon, 10.02.2020). Dette er også erfaringen til Michael Huber (geotekniker i ERA Geo, personlig kommunikasjon, 16.04.2020).

I en epost-utveksling med Jeroen Hermans (geotekniker i Sweco Nederland, personlig kommunikasjon, 01.02.2021) presenterte han et prosjekt der han og andre geoteknikere benyttet Python scripting for å kjøre Plaxis-analyser. Prosjektet fant sted i nærheten av Amsterdam og gikk ut på å bygge en tunnel for å flytte eksisterende motorvei under bakken. Flere høyhus befant seg i nærheten av konstruksjonsområdet, og man ønsket å undersøke om bygging av tunnelen kunne påvirke nærliggende bebyggelse. Omtrent 90 % av input-data var identiske for høyhus-modellene. Dette gjaldt tunneltverrsnitt, beregningsfaser, jordparametere og de fleste strukturelle elementene. Siden modellene hadde mye til felles ble det besluttet å bruke Python scripting til å utføre analyser av høyhusene i Plaxis. Det ble laget et script som kunne gjenbrukes for alle høyhusene, og det eneste som var nødvendig å endre i scriptene var input-verdier som ikke var identisk for bygningene. Dersom det skulle oppstå endringer i prosjektet, for eksempel endringer i tunneltversnittet, kunne man enkelt gjøre endringer i scriptet og nesten umiddelbart generere helt nye og "rene" modeller. Alternativet hadde vært å gjøre endringer i eksisterende modeller i GUI i Plaxis, som kan føre til tregere beregninger og i verste fall numeriske problemer som ikke ville oppstått i en "ren" modell. Scriptet effektiviserte også behandling av Output-data i Plaxis-analysene. Output-data ble automatisk lest og skrevet til Microsoft Excel (se kapittel 3.3.1 og 3.3.1). Plaxis-modellen hadde over 30 beregningsfaser, og det ville vært tidkrevende å gå igjennom resultatetene manuelt i GUI i Plaxis.

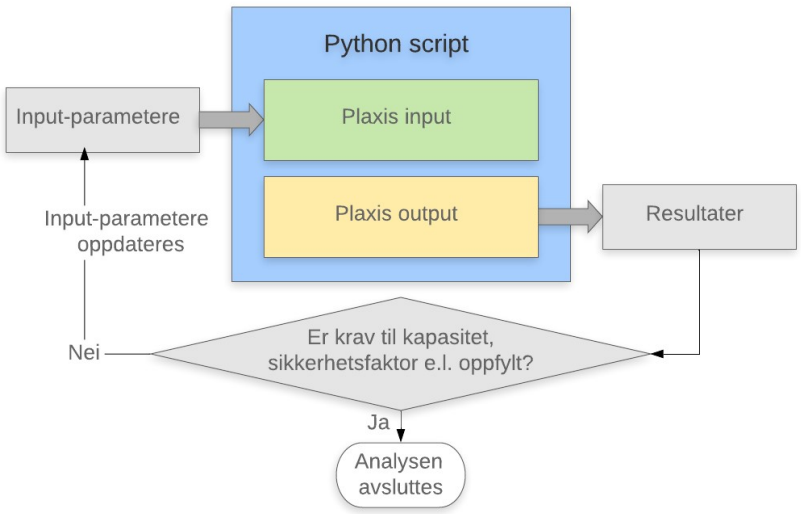
Parameterstudier

Plaxis remote scripting er svært nyttig, om ikke nødvendig, i parameterstudier i Plaxis (van der Sloot, 2019). Et parameterstudie går ut på å undersøke effekten av å endre enkelte parametere i en

modell. I denne typen studier utføres et stort antall analyser der én eller flere Input-parametere endres i modellene. Ved å benytte et Python script til å utføre beregniner kan brukeren styre Plaxis til å kjøre analyser der Input-parametere oppdateres automatisk og resultatene blir lagret underveis. I parameterstudier kan det være nødvendig å kjøre et stort antall analyser, og det ville vært svært tidkrevende å utføre parameterstudier i det grafiske brukergrensesnittet i Plaxis (Emir Oguz, doktorgradstipendiat ved NTNU, personlig kommunikasjon, 13.3.2020).

Iterative beregninger

I mange geotekniske problemstillinger er det nødvendig å bruke iterasjon for å finne et design som tilfredsstiller krav i brudd- og bruksgrensetilstand. Når Plaxis brukes til å løse slike problemer er det vanlig å kjøre én analyse av gangen, og brukeren må oppdatere input i modellen manuelt for å kjøre neste analyse. Ulike verktøy i Python muliggjør en automatisering av slike iterative prosesser. If-setninger og løkker (se kapittel 3.2.7) er svært nyttige i Plaxis-analyser. En while-løkke kan for eksempel brukes for å kjøre en Plaxis-analyse med oppdaterte Input-verdier inntil en betingelse er tilfredsstilt. En automatisering av Plaxis gjør det altså mulig å utføre iterative analyser uten at det krever brukerens oppmerksomhet, og åpner muligheten for å kjøre Plaxis-beregninger om natta (van der Sloot, 2019). Dette avhenger imidlertid av graden av automatisering av analysene. Figur 3.11 viser prinsippet for en iterativ beregning i Plaxis som er automatisert med et Python script.



Figur 3.11: Prinsippet bak en iterativ analyse som kjøres i Plaxis ved hjelp av et Python script. Figuren er basert på en figur i presentasjonen *L&L: Automation of Plaxis FEAs with python scripting* (Choi et al., 2016).

Lese og lagre Output-verdier

Ved å benytte Python kan brukeren enkelt hente ut og lagre verdier fra Output-vinduet i Plaxis. Resultater i Plaxis kan lagres i variabler eller i data-sekvenser (lister, dictionaries og så videre) underveis i en analyse. På denne måten slipper brukeren å lagre ønskelige resultater manuelt i Output-vinduet. Dette er spesielt fordelaktig i Plaxis-analyser med mange beregningsfaser der man ønsker å hente ut deformasjoner, krefter og andre resultater i beregningsfasene.

Integrasjon av Plaxis og andre programmer

Python kommer med innebygde funksjoner som muliggjør import og eksport av data mellom Plaxis og andre programmer (Choi et al., 2016), som for eksempel Microsoft Excel og Geosuite. Når data importeres direkte fra andre programmer slipper brukeren å skrive dem inn i Plaxis manuelt. Dette sparer brukeren for repetitivt arbeid i Plaxis samtidig som sannsynligheten for inntastingsfeil reduseres. Det kan dessuten være lettere å kvalitetssikre Input-verdier når de er tabellert i et regneark, sammenlignet med å kvalitetssikre Input-verdier i GUI i Plaxis (Choi et al., 2016). Figur 3.12 viser hvordan koding i Python kan brukes til å lese materialparametere fra et Excel-ark for deretter å opprette et materiale i Plaxis. Når en Plaxis-analyse er kjørt kan Python også brukes til å eksportere data fra Output-vinduet til andre programmer der resultatene kan

bearbeides.

```python
#Importerer python-biblioteket openpyxl:
import openpyxl as xl

#Definerer programsti og laster inn excel-arket:
wb = xl.load_workbook('C:/Users/Dina Erika Hansen/Documents/Masterc

#Velger arket jeg ønsker å aksessere i Excel-filen:
ws = wb["HS"]

#Oppretter materiale og henter materialparametere fra excel-arket:
material= g_i.soilmat()
material.setproperties(
"MaterialName", ws.cell(row=15,column=3).value,
"SoilModel", ws.cell(row=16,column=3).value,
"DrainageType", ws.cell(row=17,column=3).value,
"gammaUnsat", ws.cell(row=18,column=3).value,
"gammaSat", ws.cell(row=19,column=3).value,
"E50ref", ws.cell(row=20,column=3).value,
"EoedRef", ws.cell(row=21,column=3).value,
"EurRef", ws.cell(row=22,column=3).value,
"powerm", ws.cell(row=23,column=3).value,
"cref", ws.cell(row=24,column=3).value,
"phi",  ws.cell(row=25,column=3).value,
"psi",  ws.cell(row=26,column=3).value,
"nu", ws.cell(row=27,column=3).value,
"Rinter", ws.cell(row=28,column=3).value,
"K0NC", ws.cell(row=29,column=3).value,
"OCR", ws.cell(row=30,column=3).value
)
```

(a) Utklipp fra Spyder 4

14	**Jordparametere**		
15	Identification		Sand
16	SoilModel		3,00
17	Drainage type		drained
18	γunsat	[kN/m3]	17,00
19	γsat	[kN/m3]	20,00
20	E50Ref	[kN/m2]	20000,00
21	EoedRef	[kN/m2]	20000,00
22	EurRef	[kN/m2]	60000,00
23	powerm		0,50
24	c'ref	[kN/m2]	0,10
25	φ'	[°]	39,00
26	ψ	[°]	0,00
27	nu		0,20
28	Rinter		0,70
29	K0NC		0,37
30	OCR		1,00

HS

(b) Utklipp fra regneark i Excel

Figur 3.12: Figur (a) viser et utklipp fra Spyder der Python-kommandoer brukes til å lese verdier fra Excel-arket i figur (b) og opprette et materiale i Plaxis. openpyxl er en pakke som benyttes til å lese og skrive til Excel-filer. Boilerplate code (se vedlegg C) er ikke tatt med i figuren.

Michael Huber benytter Python til å importere og eksportere data i Plaxis (personlig kommunikasjon, 07.05.2020). I et av prosjektene han jobber med skal deler av jorda under et planlagt bygg erstattes med skumglass for å kompensere for last fra bygget. For å finne nødvendig mengde skumglass i prosjektet blir terrenginformasjon fra felt lest inn og analysert i Plaxis, og nødvendig mektighet skumglass blir beregnet for hvert punkt. Ved hjelp et Python script blir resultatene fra analysen skrevet til en tekstfil som konverteres til en terrengmodell som viser mektigheten av skumglass som skal legges ut i felt.

Det er også mulig å bruke Python til å importere data direkte fra Geosuite til Plaxis. Ved hjelp av Python scripting kan CPT-resultater og informasjon om grunnforhold leses og skrives til Plaxis (V. Renström, personlig kommunikasjon, 05.06.2020).

3.3.2 Plaxis remote scripting og spuntanalyser

Plaxis-analyser som involverer spuntvegger har ofte mange beregningsfaser der det er ønskelig å finne momenter og krefter i spuntvegg og forankring i mange av fasene. Ved hjelp av Python-kommandoer kan resultater i Plaxis Output lagres uten at brukeren trenger å gå igjennom Output-vinduene i Plaxis manuelt. På denne måten kan behandling av output-data effektiviseres og sannsynligheten for menneskelige inntastingsfeil reduseres, såfremt scriptet ikke inneholder feil (Michael Huber, geotekniker i ERA Geo, personlig kommunikasjon, 8.12.2020). Dessuten kan deler av scriptet gjenbrukes i andre spuntberegninger. Michael Huber har to års erfaring med Plaxis remote scripting, og bruker scripting i spuntberegninger i Plaxis. Huber bruker scripting til resultatkontroll, der han henter ut og kontrollerer resultater i Plaxis-analyser som er utført på den tradisjonelle måten. Huber mener at scripting har stort potensiale i spunt-analyser, men påpeker at det kan være tidkrevende å lage de første Python scriptene, og at det kan ta en stund fra man begynner å lage scripts til man faktisk sparer tid på det.

Det er ofte nødvendig å bruke iterasjon for å dimensjonere en spuntvegg, og løkker i Python er svært nyttige i iterative beregninger (se avsnitt 3.3.1).

3.3.3 Begrensninger for Plaxis remote scripting

Python scripting kan være fordelaktig i mange tilfeller, men det finnes også ulemper og begrensinger knyttet til Plaxis remote scripting.

Det kan være tidkrevende å lage et script som skal kjøres i Plaxis. For at scripting totalt sett skal være tidsbesparende er det derfor viktig at scriptene gjenbrukes i flere analyser og prosjekter, slik at arbeidet og tiden som er lagt ned for å lage scriptet kommer brukeren til gode. Analyser som utføres i et prosjekt kan være så spesifikke at det ikke lønner seg å lage et script.

Grunnleggende ferdigheter i Python er nødvendig for å kunne lage og bruke scripts i Plaxis, og dette er det ikke alle geoteknikere som har. Dersom Python scripts blir tatt i bruk i en bedrift, kan man risikere at geoteknikere uten erfaring med Python scripting bruker scriptene ukritisk, og dette kan medføre i feil i Plaxis-modellene (V. Renström, personlig kommunikasjon, 05.06.2020). Ferdighetene som kreves for å kunne bruke et script i Plaxis avhenger imidlertid av scriptet som skal kjøres. Dersom scriptet er skrevet på en måte slik at det kommer godt fram hvordan det fungerer, kreves det ikke dyptgående kunnskaper i Python for å kunne bruke det.

I underkapittel 3.3.1 ble det nevnt at bruk av scripting kan redusere repetitivt arbeid og inntastingsfeil i Plaxis-analyser. På en annen side vil en feil i Python-koden i et script føre til feil i alle Plaxis-analysene som kjøres med scriptet. Det er derfor svært viktig å bruke tid på å kvalitetssjekke og utføre grundige tester av et script før det tas i bruk (Choi et al., 2016).

Når man bruker det grafiske brukergrensesnittet til å kjøre analyser i Plaxis er det vanlig å inspisere Plaxis Output etter å ha utført beregninger for en modell. Når iterative beregninger i Plaxis utføres med Python får ikke brukeren muligheten til å inspisere Plaxis Output før en ny analyse kjøres. Det er derfor viktig å kontrollere at modellen i scriptet fungerer som den før scriptet kjøres.

3.3.4 Bruk av Python scripting i norske rådgiverbedrifter

I forbindelse med forstudiet til denne masteroppgaven ble en rekke geoteknikere kontaktet for spørsmål om bruk av scripting i Plaxis. I forstudiet kom det fram at et fåtall av geoteknikere i norske rådgiverbedrifter benytter Plaxis remote scripting per i dag. Det er få bedrifter som har et rammeverk for bruk av Plaxis remote scripting, og innsatsen er hovedsakelig individuell.

Det kan være flere grunner til at Plaxis remote scripting foreløpig er lite utbredt blant rådgiverbedrifter i Norge. Flere av geoteknikerne som ble kontaktet påpekte at Plaxis ikke er det mest brukte programmet i geotekniske analyser i deres bedrifter. Noen geoteknikere oppga at de er fornøyde med det grafiske brukergrensesnittet i Plaxis, og at de derfor ikke har vært på leting etter metoder for å effektivisere programmet. Enkelte var heller ikke klar over denne muligheten. For å være i stand til å bruke et script, kreves det grunnleggende ferdigheter i Python, og dette har ikke alle geoteknikere. Den snevre bruken av Python scripting kan også skyldes at mange generelt er skeptiske til nye programmer og metoder før de ser at det nye fungerer bedre enn det etablerte.

Blant geoteknikere som har erfaring med Python scripting er det bred enighet om at scripting i Plaxis-analyser har potensiale, og at automatisering generelt kommer til å bli tatt i bruk i større grad i tiden fremover.

3.4 Python-kommandoer i Plaxis

3.4.1 Opprette kobling mellom Python og Plaxis

For å styre Plaxis med Python-kommandoer er man nødt til å sette opp en server i Plaxis (vedlegg B), og kjøre en boilerplate code i Python (vedlegg C) slik at det opprettes en kobling mellom programmene. Når tilkoblingen er vellykket kan Plaxis styres med standardiserte kommandoer i Python.

3.4.2 Hjelpemidler for å skrive Python-kommandoer til Plaxis

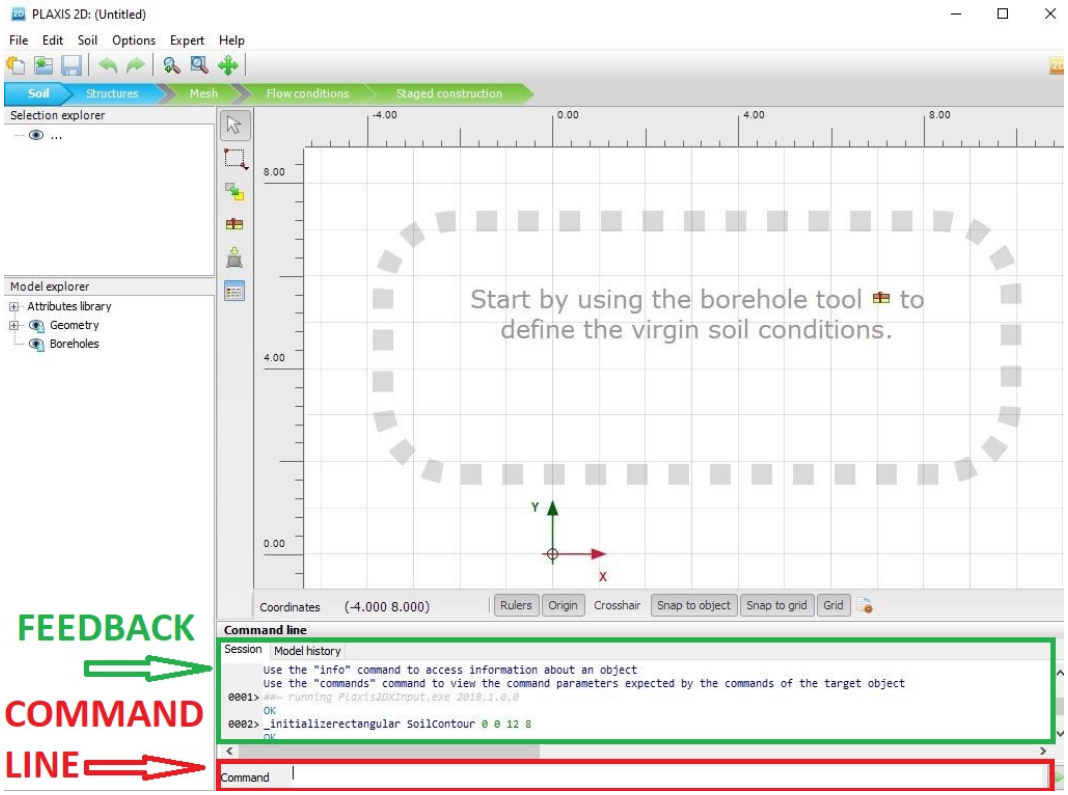
Sammenlignet med andre Python-biblioteker finnes det begrenset med litteratur som omhandler plxscripting-biblioteket. Det finnes likevel noen nyttige hjelpemidler for å lære seg syntaks for Python-kommandoer i Plaxis:

- Masteroppgaven *Loaded Bucket Foundations in Sand* (Grecu, 2018) inneholder vedlegget *Guidelines for PYTHON scripting* som gir en innføring i grunnleggende Python-kommandoer som kan benyttes i Plaxis.
- Plaxis Soilvision (Systems, 2020) på Bentley Systems' nettside er et godt hjelpemiddel for å skrive kommandoer til Plaxis. På denne siden finnes det flere eksempler på Python-kode for ulike formål, samt et diskusjonsforum der flere av innleggene er relatert til Plaxis remote scripting.
- *Scripting reference* inneholder eksempler på Python-kommandoer som kan benyttes for å styre Plaxis. *Scripting reference* finnes i PLAXIS CONNECT Edition V20 og er ikke tilgjengelig for eldre versjoner av Plaxis. *Scripting reference* dekker ikke alle Python-kommandoer i Plaxis.
- *Command line* i GUI Plaxis er et essensielt verktøy for å skrive Python-kommandoer til Plaxis.

De neste underkapitlene beskriver hvordan *Command line* og *command reference* kan brukes for å skrive kommandoer til Plaxis.

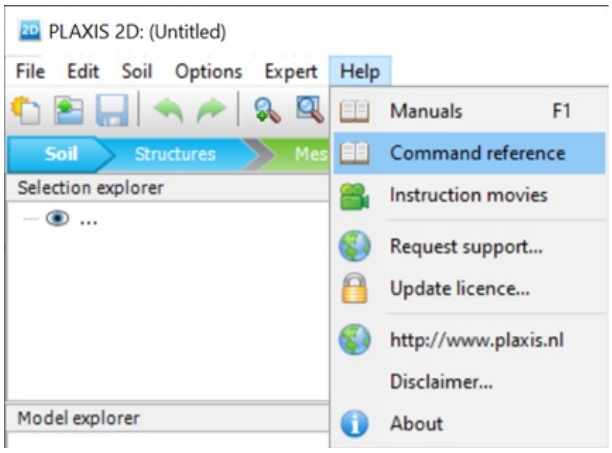
3.4.3 *Command line* og *Command reference* i Plaxis

Alle operasjoner som utføres manuelt i menyene i Plaxis blir automatisk oversatt til kommandoer. Kommandoene dukker opp i *Command line* når brukeren gjør endringer i modellen, se figur 3.13. *Feedback*-vinduet, som er plassert over *Command line*, gir en oversikt over tidligere kommandoer. Kommandoer som er utført på korrekt vis returnerer grønn skrift i feedback, mens kommandoer som inneholder feil og ikke har blitt utført returnerer rød skrift.



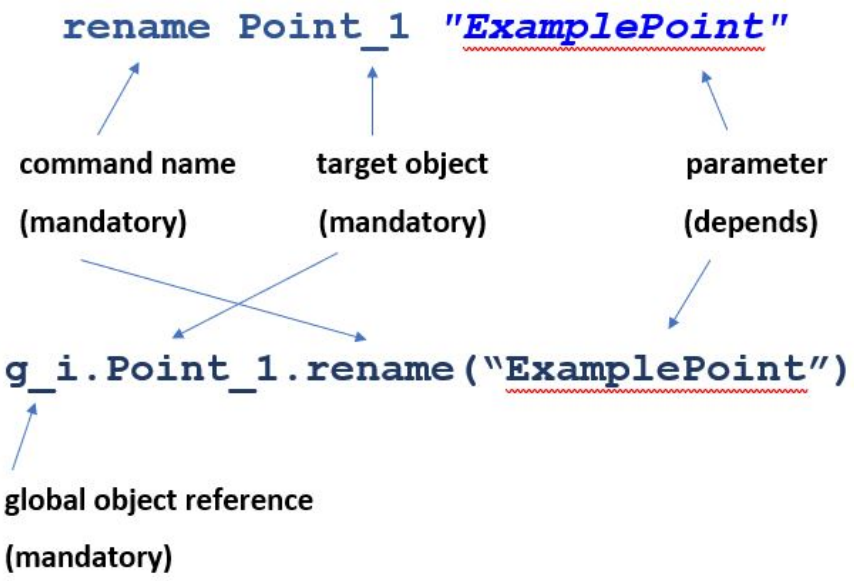
Figur 3.13: *Command line* i Plaxis

Command reference i *Help*-menyen (figur 3.14) gir en oversikt over Plaxis-kommandoer, samt eksempler.



Figur 3.14: *Command reference* i Plaxis.

Kommandoer i Python og Plaxis har lignende syntaks, og *Command line* og *Command reference* kan derfor brukes til å utforme kommandoer i Python. Dersom man er usikker på hvordan en bestemt kommando i Python skal utformes, er et godt tips å utføre ønskelige operasjoner i det grafiske brukergrensesnittet i Plaxis, for så å studere kommandoene som dukker opp i *Feedback*-vinduet. Plaxis-kommandoene er som regel overførbare til Python. Figur 3.15 viser syntaks for kommandoer i Plaxis vs. syntaks for kommandoer i Python. Eksempelet i figuren viser hvordan navnet på et punkt i Plaxis endres fra *Point_1* til *ExamplePoint*. Kommandoer består alltid av command name og target object, mens parameter må være med i enkelte kommandoer. I Python er det også nødvendig å ta med en såkalt global object reference, *g_i*, foran kommandoen. *g_i* defineres i Boilerplate code i starten av et script, som vist i vedlegg C.



Figur 3.15: Figuren viser syntaks for kommandoer i Plaxis (øverst) og i Python (nederst). Bildet er basert på en figur fra et webinar på Bentley Systems nettsider (van der Sloot, 2019).

info og *echo* er kommandoer som kan benyttes i *Command line* i Plaxis. *Info* og *echo* gir detaljert informasjon om objekter i Plaxis, samt hvilke kommandoer og egenskaper som er tilgjengelige for de ulike objektene. Dette er nyttig i Python scripting fordi mye av informasjonen kan overføres til Python kommandoer.

Figur 3.16 viser informasjonen som vises i *feedback*-vinduet når *echo*-kommandoen brukes i *Command line*.

```
0005> echo polygon_1
Polygon named "Polygon_1"
x: -28
y: -8
Features: 1
Soil named "Soil_1" on Polygon_1
Material: <not assigned>
Features: 2
WaterConditions named "WaterConditions_1" on Soil_1
Conditions: Global level (0)
VolumeStrain named "VolumeStrain_1" on Soil_1
Points:
0: (-28.00; -8.000)
1: (-11.00; -8.000)
2: (-2.000; -1.000)
3: (16.00; -1.000)
4: (14.00; -12.00)
5: (-28.00; -12.00)
```

Figur 3.16: *Echo*-kommandoen blir brukt til å få informasjon om et soil polygon i en modell (Bentley Systems, 2015a).

Figur 3.17 viser informasjonen som vises i *feedback*-vinduet når *info*-kommandoen brukes i *Command line*.

```
info borehole_1
Borehole_1
  Commands: echo, commands, rename, set, multiply, info, setproperties, move
  Attributes: Name, TypeName, Comments, UserFeatures, Parent, Generator, x, y, z, Head, FieldData, FieldDataInterpreter, FieldDataMinThickness
```

Figur 3.17: *Info*-kommandoen kan brukes til å få informasjon om tilgjengelige kommandoer og egenskaper (attributes) for et objekt. I dette tilfellet er objektet borehole_1.

Kapittel 4

Python-scriptet og Plaxis-modellen i oppgaven

4.1 Fremgangsmåte for å utarbeide scriptet

Hjelpemidlene som ble brukt for å lage scriptet er beskrevet i kapittel 3.4.2. Det finnes få tilgjengelige eksempler på scripts som kan kjøres i Plaxis. Det er relativt få geoteknikere som benytter Python scripting i Plaxis, og bedrifter er forsiktige med å dele sine scripts. Det var derfor ikke mulig å finne en form for "mal" som viser hvordan et script for spuntanalyser skal se ut, og scriptet i denne oppgaven er i hovedsak utarbeidet ved å prøve og feile.

Scriptet er skrevet i Python 3.7 og IDE'en Spyder 4 ble brukt til å skrive og kjøre koden. Scriptet ble kjørt i Plaxis 2D V20, og kan også kjøres i Plaxis 2D 2018.

4.2 Begrensninger

Scriptet har en rekke begrensninger:

- Spuntveggen som modelleres er en svevespunt.
- Spuntveggen har ett avstivningsnivå.
- Ankeret er horisontalt og har den samme plasseringen, uavhengig av gravedybde. Scriptet er testet med ankerplassering 1.5 meter under bakkenivå.

- Grunnvannstanden er satt på samme kotenivå som bunnen av utgravingen.
- Scriptet er kun testet for utgravinger med dybde mellom fire og ti meter.
- Scriptet er testet for tre jordmaterialer med materialmodellen Hardening Soil, forkortet HS.
- Deformasjoner er ikke brukt som en betingelse i kjøringen av scriptet.
- Maksimalt moment M_{max} som opptrer i spuntveggen er ikke en betingelse for kjøring av scriptet, og brukeren må selv kontrollere at momentkapasiteten ikke overskrides. Det samme gjelder normalkraften, N_{max}, som opptrer i ankeret. M_{max} og N_{max} lagres som resultater i Excel-filene i oppgaven.
- Materialfaktoren er ikke inkludert i jordparameterne. Dersom msf > sikkerhetsfaktor og kapasiteten for anker eller spuntvegg overskrides, må det kjøres en ny analyse der materialfaktoren er inkludert i jordparameterne. På denne måten vil man kunne finne maksimale krefter og momenter i bruddgrensetilstand.
- Som man kan se i flyskjemaet i tillegg E er Error Code = 0 for utgravingsfasen en betingelse for kjøring av koden. Dersom Error Code ikke er lik null på grunn av numeriske problemer (Eksempel: Error Code = 103) i Plaxis vil koden kjøres som om likevekt ikke er oppnådd i modellen (Error Code = 101). Brukeren må selv undersøke resultater i Excel-filene og vurdere hva som kan være årsaken til at Error Code ikke er lik 0.
- Scriptet kontrollerer ikke at msf i *Safety*-analysene i scriptet konvergerer, og dette kan resultere i noe unøyaktige msf-verdier i resultatene.

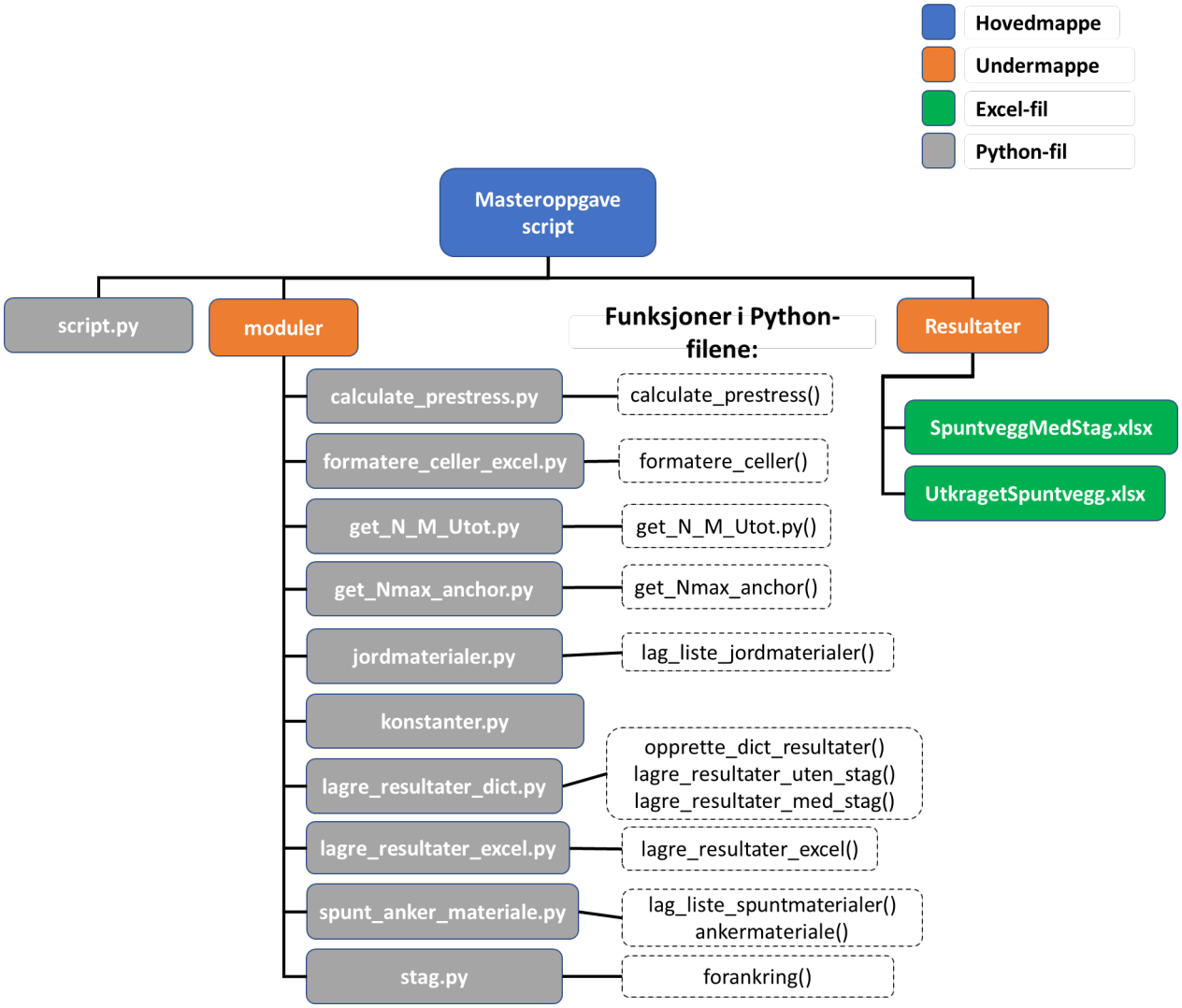
4.3 Hvordan kjøre scriptet

Scriptet i oppgaven består av *script.py* og moduler, og kjøres fra *script.py*. Før scriptet kjøres er brukeren nødt til å definere konstanter og programstier i *konstanter.py*, samt definere variablene i den første kodeseksjonen i *script.py*. Dette er forklart i detalj med kommentarer i koden. Når alle analyser er utført vil resultater automatisk skrives til Excel-dokumentene i mappen *Resultater*.

Dict'ene med resultater skrives til konsollen i Spyder underveis i kjøringen av scriptet, slik at brukeren kan følge med på resultatene samtidig som Plaxis-analysene pågår.

4.4 Filer som er utarbeidet i oppgaven.

Figur 4.1 gir en oversikt over mappen med filene som er utarbeidet i denne oppgaven.



Figur 4.1: Oversikt over filer som er utarbeidet i masteroppgaven.

4.4.1 Script

Koden i *script.py* er delt opp i ulike seksjoner for å gjøre koden oversiktlig. Kommentarer med stiplede linjer markerer skillet mellom de ulike seksjonene. Koden i *script.py* er delt inn i følgende seksjoner:

- Kodeseksjon der brukeren må definere variabler som brukes i Plaxis-analysene.
- Kodelinje der variabler importeres fra *konstanter.py*

- Pakker blir importert.
- Boilerplate code (se tillegg C)
- Funksjoner fra modulene importeres
- Nøstet for-løkke.
- Resultatene eksporteres til Excel-filene.

4.4.2 konstanter.py

I *konstanter.py* defineres variabler som er konstante gjennom kjøringen av scriptet. Denne modulen inneholder altså ingen funksjoner. Brukeren av scriptet må selv definere disse variablene, og kommentarer i modulen gir detaljerte instrukser for hvordan dette gjøres. *konstanter.py* importeres i starten av de fleste modulene siden variablene brukes i boilerplate code (se tillegg C).

4.4.3 Moduler og funksjoner

De fleste modulene starter med en boilerplate code, samt import av nødvendige pakker. Deretter defineres funksjoner. Figur 4.1 gir en oversikt over moduler med tilhørende funksjoner. Tabell 4.1 gir en kort beskrivelse av funksjonene.

Tabell 4.1: Beskrivelse av funksjoner som er definert i modulene.

Funksjon	**Beskrivelse**
calculate_prestress()	Funksjon som beregner og returnerer forspenning (prestress) for ankeret i analysene.
forankring()	Funksjonen *forankring* blir kalt på når det er behov for å legge til forankring i modellen. I denne funksjonen opprettes nye beregningsfaser for spuntvegg med forankring, og programflyten videre avhenger av sikkerhetsfaktoren i beregningene. Det rosa området i flytskjemaet i tillegg (E) viser flyten i koden når funksjonen *forankring()* blir kalt på.
formatere_celler_excel()	Formaterer cellene i Excel-filene *UtkragetSpuntvegg* og *SpuntveggMedStag*. Bredde til Excel-cellene økes og teksten i den øverste raden i tabellen brytes automatisk. Funksjonen har ingen returverdier.
get_N_M_Utot()	Etter at beregninger i Plaxis er utført blir denne funksjonen kalt på. Funksjonen returnerer maksimal aksialkraft (N_{max}), moment (M_{max}) og deformasjon (U_{tot}) for spuntvegg for plastiske beregningsfaser og for *Safety*-fasen (c/phi-reduksjon). Funksjonen har altså seks returverdier. Deler av koden er hentet fra *Plaxis Soilvision Wiki* (Bentley Communities, 2020b).
get_Nmax_anchor()	Denne funksjonen blir kalt på etter beregninger er utført for spuntvegg med forankring. Funksjonen returnerer maksimal aksialkraft i ankeret for plastiske beregningsfaser og for *Safety*-fasen (c/phi-reduksjon). Funksjonen har to returverdier. Deler av koden er hentet fra *Plaxis Soilvision Wiki* (Bentley Communities, 2020b).
lag_liste_jordmaterialer()	Jordmaterialene som skal benyttes i Plaxis-modellen er definert inne i denne funksjonen. I denne oppgaven er dette sand med løs, middels og tett pakning (se tabell 4.4). Dersom brukeren vil kjøre analyser med andre jordmaterialer må disse defineres i *lag_liste_jordmaterialer()*. Når *lag_liste_jordmaterialer()* blir kalt på opprettes jordmaterialene, samt en liste over navn på jordmaterialene. Liste med jordmaterialene blir opprettet for å kunne loope igjennom ulike jordmaterialer i den nøstede for-løkken i scriptet.
lagre_resultater_excel()	Funksjon som skriver dict'ene *resultater_uten_stag* og *resultater_med_stag* til Excel-filene *UtkragetSpuntvegg.xlsx* og *SpuntveggMedStag.xlsx*
lagre_resultater_med_stag()	Legger til resultater i dict'en *resultater_med_stag*. Resultater blir kontinuerlig lagt til i denne dict'en når beregninger for spuntvegg med anker er utført.
lagre_resultater_uten_stag()	Legger til resultater i dict'en *resultater_uten_stag*. Resultater blir kontinuerlig lagt til i denne dict'en når beregninger for utkraget spuntvegg er utført.
opprette_dict_resultater()	Oppretter og returnerer to dictionaries (dict): *resultater_uten_stag* og *resultater_med_stag*.
spunt_anker_materiale()	Når denne funksjonen blir kalt på opprettes platematerialer og ankermaterialer i Plaxis, samt en liste over platematerialene. Platematerialene brukes til å modellere ulike typer spuntprofiler i Plaxis. Tabell 4.5 gir en oversikt over spuntprofilene som er definert i funksjonen. Listen over platematerialer blir opprettet for å kunne loope igjennom ulike spuntprofiler i den nøstede for-løkken i scriptet. Tabell 4.6 viser input-parameterne for ankeret som blir opprettet når man kaller på *spunt_anker_materiale()*.

4.5 Flytskjema for programmet

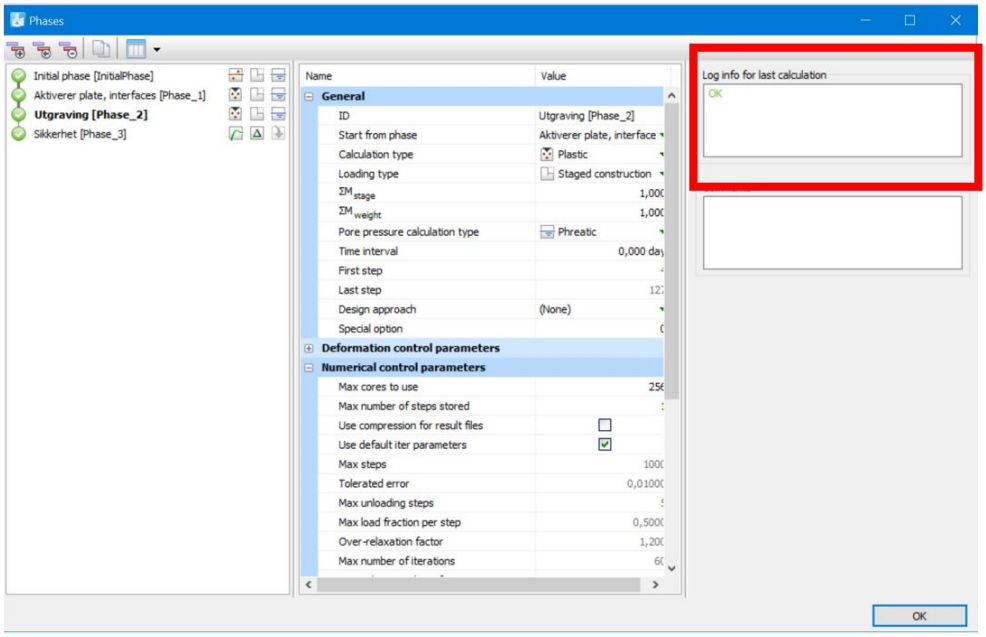
Flytskjemaet (tillegg E) viser flyten for scriptet i oppgaven. Stegene i flytskjemaet er nummererte. Resultatene i Excel inneholder en kolonne med nummere som viser hvor i flytskjemaet beregninge er utført. Rosa område i figuren viser programflyten for Plaxis-analyser med stag.

Som man ser av flytskjemaet foregår Plaxis-analysene inne i en nøstet for-løkke som er definert i *script.py*. For-løkken går igjennom elementene i lister med jordmaterialer, spuntprofiler og gravedybder. Når elementene i for-løkkene er loopet igjennom skrives resultatene fra Plaxis-analysene til Excel-filene.

Som flytskjemaet viser brukes følgende betingelser i scriptet:

- Error Code for utgravingsfasen
- Sikkerhetsfaktor i *Safety*-fasen
- Spuntlengde

Error Code = 0 betyr at likevekt i beregningsfasen er oppnådd. Figur 4.2 viser vinduet med Error Code i GUI i Plaxis. Det er nødvendig å sjekke at Error Code = 0 før msf blir lest fra Output fordi kjøring av scriptet stopper opp dersom man forsøker å lese verdier fra beregningsfaser som ikke er utført.



Figur 4.2: Error code står under *Log info for last calculation* i vinduet *Phases* i Plaxis. "OK" tilsvarer Error Code = 0.

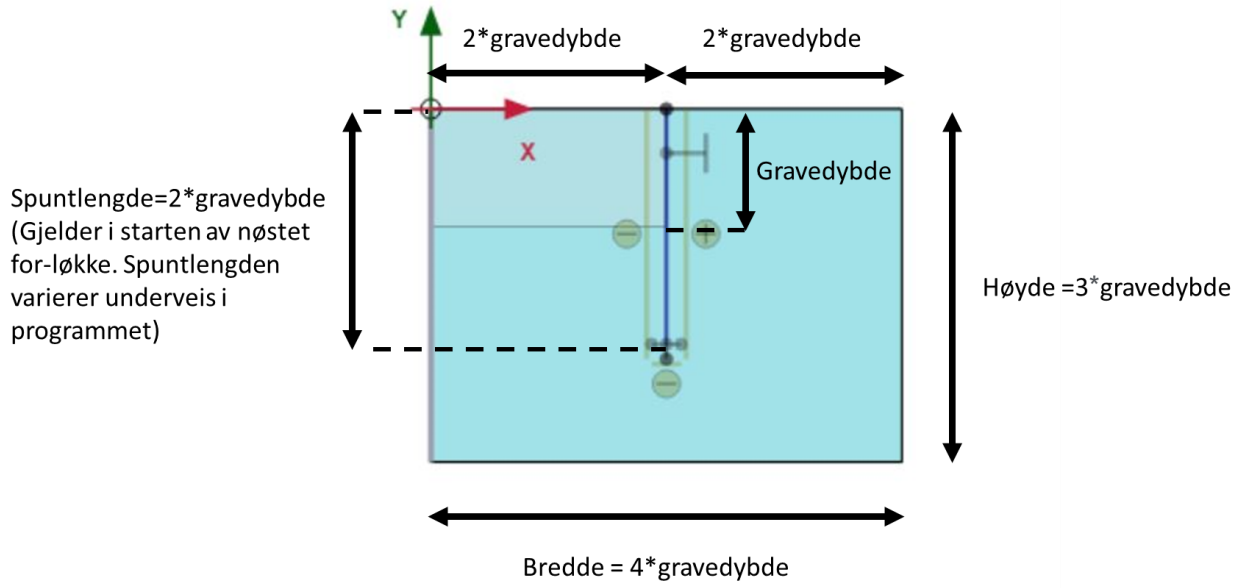
Som man kan se i flytskjemaet er det to while-løkker i scriptet: en while-løkke for analyser med utkraget spuntvegg og en annen for analyser med stag. For hver loop i while-løkken forkortes spuntlengden med én meter. while-løkken kjøres inntil betingelsene ikke er tilfredsstilt.

4.6 Plaxis-modellen som genereres med scriptet

4.6.1 Geometri

I trinn 3 i flytskjemaet opprettes et nytt prosjekt i Plaxis, slik at alle objekter (borehull, materialer, strukturelle elementer og så videre) som er i programmet fra før av blir fjernet. På denne måten starter hver loop i den nøstede for-løkken med en "ren" modell. Deretter blir modellen i figur 4.3 opprettet. Alle lengder i modellen er definert ved hjelp av gravedybden. Dette er en fordel, da man kun trenger å endre gravedybden i *script.py* for å endre geometri for jordvolum og spuntlengde i modellen. Geometrien for jordvolumet i modellen er konstant gjennom en loop i for-løkken.

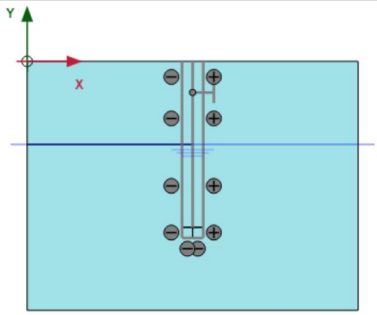
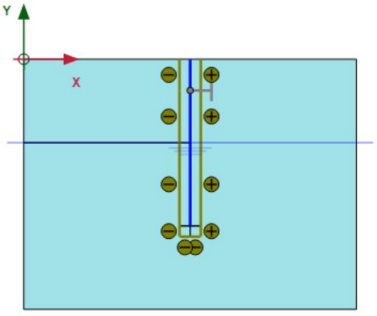
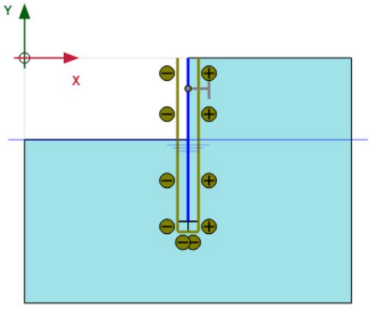
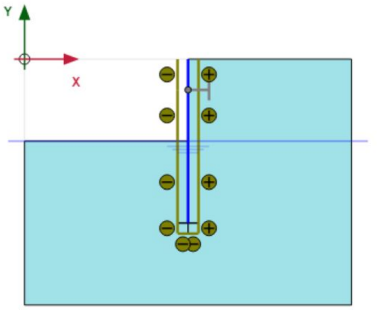
Avstand fra spuntveggen til de vertikale rendene til jordvolumet er satt til å være 2*gravedybden, slik at geometrien skal tillate utvikling av bruddmekanismen i jorda. Høyre og venstre rand er fastholdt sideveis, men jorda kan bevege seg fritt i vertikal retning. Randeffektene vil kunne ha betydning for deformasjoner, men vil ha liten innvirkning på grenselikevekten.



Figur 4.3: Modellen som opprettes i trinn 3 i flytskjemaet. Geometrien for jordvolumet i Plaxis-analysene avhenger av gravedybden.

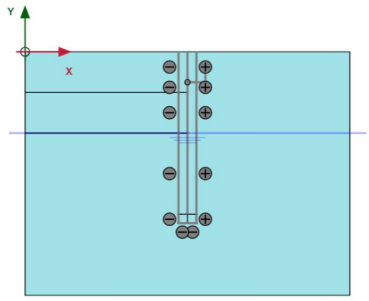
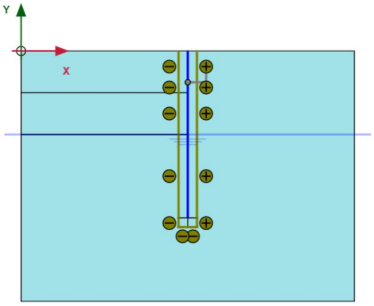
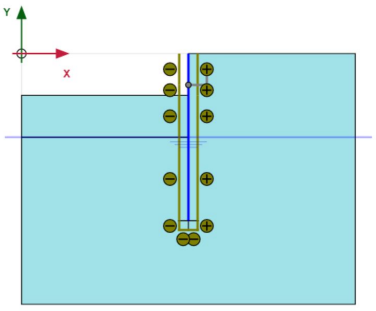
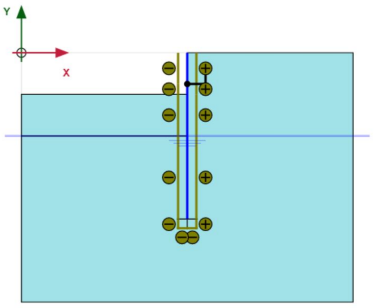
4.6.2 Beregningsfaser

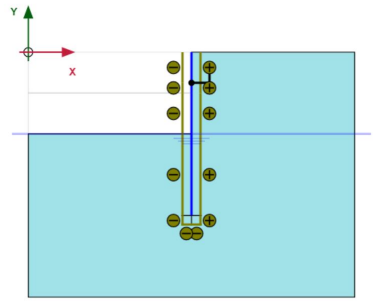
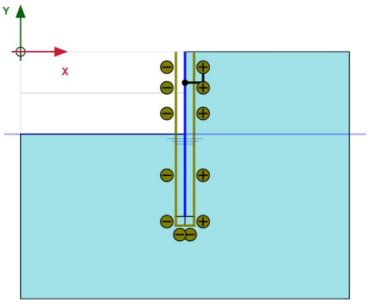
Tabell 4.2 viser beregningsfaser for utkraget spuntvegg. Beregningsfasene opprettes i trinn 3, og brukes for Plaxis-beregninger i trinn 3 og 7 i flytskjemaet.

Modell	Beregningsfase
	**Initial Phase:** Jordvolum er aktivert. Grunnvannstand er satt på samme nivå som bunnen av utgravingen.
	**Phase_1:** Aktiverer plate og interface-elementer. (Calculation type = Plastic)
	**Phase_2:** Deaktiverer polygon (utgraving). (Calculation type = Plastic)
	**Phase_3:** Sikkerhetsanalyse (Calculation type = Safety)

Tabell 4.2: Beregningsfaser for analyser med utkraget spuntvegg

Tabell 4.3 viser beregningsfaser for spuntvegg med forankring. Beregningsfasene med forankring opprettes i trinn 11 i flytskjemaet, og brukes for Plaxis-beregninger i trinn 11 og 15.

Modell	Beregningsfase
	**Initial Phase:** Jordvolum er aktivert. Grunnvannstand er satt på samme nivå som bunnen av utgravingen.
	**Phase_1:** Aktiverer plate og interface-elementer. (Calculation type = Plastic)
	**Phase_2:** Utgraving for installasjon av anker. Polygon deaktiveres. (Calculation type = Plastic)
	**Phase_4:** Aktiverer anker. Forspenning av anker. (Calculation type = Plastic)

	**Phase_4:** Full utgraving. Polygon deaktiveres. (Calculation type = Plastic)
	**Phase_5:** Sikkerhetsanalyse (Calculation type = Safety)

Tabell 4.3: Beregningsfaser for analyser med utkraget spuntvegg

4.6.3 Materialer i modellen

For å kunne teste scriptet i oppgaven var det nødvendig å velge input-verdier for spunt, jord og anker i Plaxis-analysene. Input-parameterne for materialene er presentert i de neste underkapitlene. Dersom materialene i scriptet byttes ut kan det være behov for justeringer i koden for at scriptet skal fungere med de nye inputparameterne.

Jordmaterialer

HS-modellen er brukt til å modellere jordmaterialene i analysene. Tabell 4.4 viser input-parametere for sand med løs, middels og fast pakning som ble brukt til å teste scriptet i oppgaven. Materialene er definert i funksjonen *lag_liste_jordmaterialer*. For å kjøre analyser med andre materialer kan man slette kode for HS-materialene i *lag_liste_jordmaterialer* og opprette nye materialer.

	Benevning	**Sand Løs**	**Sand Middels fast**	**Sand Fast**
Material Model		Hardening Soil	Hardening Soil	Hardening Soil
Drainage type		Drained	Drained	Drained
γ_{unsat}	kN/m^3	16	17	18.2
γ_{sat}	kN/m^3	19.4	19.8	20.3
E_{50}^{ref}	kN/m^2	20 000	30 000	40 000
E_{oed}^{ref}	kN/m^2	20 000	30 000	40 000
E_{ur}^{ref}	kN/m^2	60 000	90 000	120 000
$power\ (m)$	-	0.5	0.5	0.5
c'_{ref}	kN/m^2	1	1	1
ϕ	^{o}	30	35	40
ψ	^{o}	0	5	10
ν_{ur}	-	0.2	0.2	0.2
K_{NC}^0	-	0.5	0.43	0.36
OCR	-	1	1	1

Tabell 4.4: Input-verdier for HS-materialer som er benyttet til testing av scriptet. Materialene er definert i funksjonen *lag_liste_jordmaterialer*. Input-parameterne er basert på *Validation of empirical formulas to derive model parameters for sands* (Brinkgreve & Engin, 2010)

Kohesjon (c_{ref}) var opprinnelig satt til 0, men på grunn av numeriske problemer i Plaxis (Error Code 103) ble kohesjon endret til 1 kN/m^2

Spuntprofiler

Tabell gir en oversikt over spuntprofiler som brukes i Plaxis-analysene. Spuntprofilene er definert i funksjonen *spunt_anker_materiale()*, og brukeren velger spuntprofiler for Plaxis-analysene *script.py*.

Tabell 4.5: Spuntprofiler som er definert i funksjonen *spunt_anker_materiale*. Data for spuntprofiler er hentet fra ArcelorMittal (2020).

Parameter	**Benevning**	**AZ12-700**	**AZ17-700**	**AZ26-700**	**AZ36-700**	**AZ46**
EA	kN/m	2.583E6	2.793E6	3.927E6	4.536E6	6.111E6
EI	kNm^2	39.65E3	76.08E3	125.41E3	188.45E3	231.95E3
w	$kN/m/m$	0.9486	1.0241	1.4421	1.6579	2.2465
$\upsilon(nu)$	-	0.25	0.25	0.25	0.25	0.25
Mp	kNm/m	500.4	672.6	1019.4	1389.9	1790.2
$N_{p,1}$	kN/m	4158.6	4496.7	6322.4	7302.9	9838.6

Løsmassestag

Løsmassestaget i modeller er av typen fixed end anchor. Ankermaterialet som er brukt til testing av scriptet er definert i funksjonen *spunt_anker_materiale()*. Tabell 4.6 viser input-parameterne

for ankeret.

Tabell 4.6

Parameter	**Benevning**	Fixed End Anchor
Elastisitet	-	Elastisk
EA	kN	153.3E3
$L_{spacing}$	m	1
Equivalent length	m	10

Stivhet for ankeret er hentet fra Teknisk datablad for Ischebeck TITAN Injeksjonsstag og Peler for stag 40/20. Siden ankermaterialet er elastisk må brukeren selv kontrollere at N_{max} ikke overskrider kapasiteten til ankeret som er valgt. Scriptet er testet med en staglengde på 10 meter. Staglengden kan endres i den første kodeseksjonen i *script.py*. Stagets forspenning beregnes i funksjonen *calculate_prestress*, og er satt lik resultantkraften fra K0-trykket fra jorda som graves ut når ankeret installeres.

4.7 Elementinndeling (mesh)

Elementinndelingen (mesh) velges i den første kodeseksjonen i *script.py*. Scriptet er testet med alle elementinndelingene i Plaxis, fra veldig grovt til veldig fint.

4.8 Sikkerhetsfaktor

Sikkerhetsfaktoren defineres av brukeren i den første kodeseksjonen i *script.py*. Scriptet er testet med sikkerhetsfaktor 1.25. Sikkerhetsfaktoren inngår ikke i beregningene, men brukes til betingelser i if-setninger og while-løkker.

4.9 Kontroll av resultater

For å kontrollere resultater fra kjøring av scriptet, ble Plaxis Output for tilfeldige modeller inspisert. Verdier i Plaxis Output ble sammenlignet med resultatene i Excel-filene, og det ble gjort en vurdering på hvorvidt bruddmekanismen for modellene var realistiske. Selv om enkelte modeller ble kontrollert, er ikke dette en garanti for at scriptet er feilfritt. Scriptet burde derfor brukes kritisk.

4.10 Resultater

Tillegg D viser hvordan Excel-filene *UtkragetSpuntvegg.xlsx* og *SpuntveggMedStag.xlsx* ser ut. Scriptet er i dette tilfellet kjørt med elastisk platemateriale. Kolonnene som er merket med *Plastic*, som for eksempel *Mmax [kNm/m] spunt, Plastic*, viser verdier som er hentet fra plastiske beregningsfaser. Funksjonen *get_N_M_Utot* henter ut maksimal absoluttverdi for normalkraft, bøyemoment og deformasjon for spuntveggen fra alle de plastiske beregningsfasene. Verdiene som står i tabellen er altså de største verdiene gjennom alle de plastiske fasene. Kolonner som er merket med *Safety* viser maksimale absoluttverdier som er hentet fra beregningsfase med calculation type *Safety*. Kolonnen med navnet *Prosess i flytskjema* viser hvor i flytskjemaet beregningene er utført.

4.11 Feilmeldinger og løsninger

I arbeidet med scriptet dukket det opp feilmeldinger i konsollen i Python og i GUI i Plaxis. Det var til tider utfordrende å finne årsaken til feilmeldingene, da det finnes begrenset med litteratur om plxscripting-biblioteket. I de neste underkapitlene presenteres feilmeldinger som var noe krevende å løse.

4.11.1 Opprette platemateriale med Python

Når et platemateriale skal opprettes med Python-kommandoer er man nødt til å definere G_{ref} i tillegg til parameterne som defineres i GUI i Plaxis. Dersom G_{ref} ikke er definert kan følgende feilmelding dukke opp: *Normal stiffness is zero! Enter proper EA value.* G_{ref} er gitt ved følgende formel (Bentley Systems, 2020):

$$G_{ref} = \frac{E}{2 \cdot (1 + \nu)} \tag{4.1}$$

der E og d er gitt ved følgende formler:

$$E = \frac{EA}{d} \tag{4.2}$$

$$d = \sqrt{\frac{12 \cdot EI}{EA}} \tag{4.3}$$

Materialparameterne for en plate avhenger av hverandre, som man kan se i ligning 4.1-4.3. Dersom materialparameterne i scriptet kommer i konflikt med hverandre vil parameterne i Plaxis bli feil, selv når dette gjelder desimalene. Figur 4.4a og 4.4b viser kode som skal opprette spuntprofilen AZ12-700 i Plaxis. Det som skiller de to kodene er at G_{ref} og d har har færre desimaler i figur 4.4a. Materialet som opprettes når koden i figur 4.4a vil inneholde feil fordi parameterne kommer i konflikt, da desimalene ikke er nøyaktige nok.

I scriptet i oppgaven ble dette løst ved å opprette spuntprofil AZ12-700 med korrekte verdier for EA, EI og ν i GUI i Plaxis, for deretter å bruke *echo*-kommandoen i Command Line (se kapittel 3.4.3) og finne verdiene for d og G_{ref}. d og G_{ref} i figur 4.4b er bestemt på denne måten, og når denne koden kjøres opprettes platematerialet AZ12-700 på korrekt vis.

```
AZ12_700 = g_i.platemat()
AZ12_700.setproperties ("MaterialName", "AZ12-700",
"Elasticity", 1,
"IsIsotropic", True,
"EA", 2583000,
"EA2", 2583000,
"EI", 39648,
"nu", 0.25,
"d",0.4292,
"w", 0.948627,
"Mp", 500.4,
"Np", 4158.6,
"Np2", 4158.6,
"RayleighAlpha", 0,
"RayleighBeta", 0,
"Gref", 2407383.8)
```

(a)

```
AZ12_700 = g_i.platemat()
AZ12_700.setproperties ("MaterialName", "AZ12-700",
"Elasticity", 1,
"IsIsotropic", True,
"EA", 2583000,
"EA2", 2583000,
"EI", 39648,
"nu", 0.25,
"d",0.429179591722649,
"w", 0.948627,
"Mp", 500.4,
"Np", 4158.6,
"Np2", 4158.6,
"RayleighAlpha", 0,
"RayleighBeta", 0,
"Gref", 2407383.80837943)
```

(b)

Figur 4.4: Figur a) viser Python-kode der parameterne G_{ref} og d er beregnet med formlene 4.1-4.3 og rundet av. Ved å bruke koden i figur b) opprettes et platemateriale med korrekte parametere. d og G_{ref} i figur b) er funnet ved å benytte *echo*-funksjonen i GUI i Plaxis. Platematerialet som opprettes i figuren er spuntveggprofilen AZ12-700.

En annen, og kanskje bedre, løsning på problemet er å opprette variabler for parameterne og beregne verdier for G_{ref} og d i Python (Bentley Communities, 2020b). Figur 4.5 viser hvordan dette gjøres for AZ12-700.

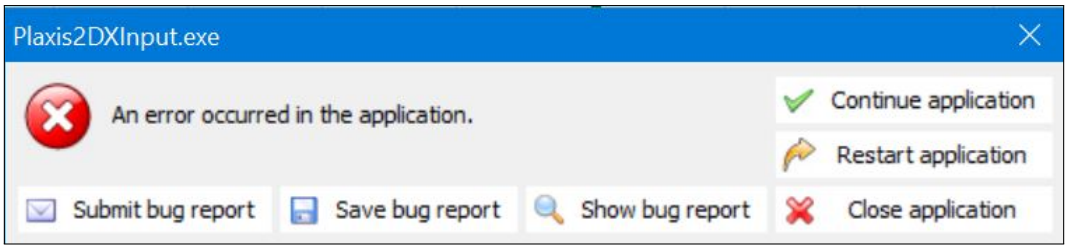
```
EI=39648
EA=2583000
d=math.sqrt(12*EI/EA)
nu=0.25
E=EA/d
Gref=E/(2*(1+nu))


AZ12_700 = g_i.platemat()
AZ12_700.setproperties ("MaterialName", "AZ12-700-3",
"Elasticity", 1,
"IsIsotropic", True,
"EA", EA,
"EA2", EA,
"EI", EI,
"nu", nu,
"d", d,
"w", 0.948627,
"Mp", 500.4,
"Np", 4158.6,
"Np2", 4158.6,
"RayleighAlpha", 0,
"RayleighBeta", 0,
"Gref", Gref
)
```

Figur 4.5

4.11.2 Lukke Output-vinduer

Når et script brukes til å kjøre mange analyser i Plaxis etter hverandre er man nødt til skrive kommandoer som lukker Output-vinduer underveis i kjøring av scriptet. Output-vinduer kan lukkes med kommandoen *s_o.close()*. Figur 4.6 viser meldingen som kan dukke opp i Plaxis Output når mange vinder er åpne samtidig.



Figur 4.6: Melding som kan dukke opp dersom det er for mange åpne vinduer i Plaxis Output.

4.11.3 Feil versjon av Python

Da arbeidet med scriptet startet dukket denne feilmeldingen opp i Python Console: *DLL load failed while importing _Blowfish. Den angitte modulen ble ikke funnet.* Feilmeldingen skyldtes at Python 3.8 ble brukt i stedet for Python 3.7. Per i dag er man nødt til å bruke Python 3.7.4 for å kunne kjøre scripts i Plaxis (Bentley Communities, 2020a).

Kapittel 5

Oppsummering, konklusjon og videre arbeid

De viktigste funnene som har blitt gjort i arbeidet med masteroppgaven er gjengitt i dette kapittelet. Oppgaven oppsummeres ved å svare på forskningsspørsmålene i innledningen.

5.1 Oppsummering

Forskningsspørsmål som ble presentert i kapittel 1.2:

- **Hvilke verktøy i Python kan brukes til å automatisere spuntberegninger i Plaxis?**

  Det er ofte nødvendig å utføre iterative beregninger for å dimensjonere en spuntvegg. Kontrollstrukturer i Python kan brukes til å styre flyten i et program og for å utføre iterative beregninger. if/elif/else-setninger kan brukes til å automatisere beslutningsprosesser, og løkker kan tas i bruk dersom man vil repetere like eller nesten like handlinger. Løkker er mye brukt i optimering og for å løse ligninger iterativt. Kontrollstrukturene som er benyttet i scriptet i denne oppgaven er for-løkker, while-løkker og if/elif/else-setninger.

  Funksjoner i Python gjør det mulig å lese resultater fra Plaxis Output underveis i analyser og lagre resultater i variabler eller datasekvenser som lister, arrays, dictionaries og lignende. På denne måten kan resultater fra spuntberegninger lagres automatisk selv om Plaxis-analyser kjøres rett etter hverandre med et script. I denne oppgaven er resultater spuntberegningene

lagres i dictionaries. Python tilbyr også funksjoner som muliggjør integrasjon av Plaxis og andre programmer, slik at input-data og resultater kan importeres og eksporteres direkte mellom disse. I denne oppgaven eksporteres resultater til Microsoft Excel når scriptet kjøres i Plaxis.

- **Hvordan kan tradisjonell dimensjonering av spuntvegg brytes ned i mindre steg og automatiseres ved hjelp av Python script?**

  Flytskjemaet i tillegg E viser hvordan dimensjonering av en spuntvegg kan brytes ned i mindre steg der if/elif/else-setninger brukes til beslutningsprosesser, og while-løkker brukes til iterative beregninger.

- **Hvilke utfordringer kan oppstå når man automatiserer Plaxis-analyser med Python?**

  Dette forskningsspørsmålet er besvart ut ifra erfaringene som ble gjort i denne masteroppgaven. Andre utfordringer knyttet til Plaxis remote scripting er diskutert i kapittel 3.3.3.

  Det finnes det begrenset med informasjon om plxscripting-biblioteket på nett. Dette gjør det noe krevende å finne løsninger for feilmeldinger som dukker opp i GUI i Plaxis og i konsollen i Python underveis i kjøringen av scriptet, spesielt når man ikke har har mye erfaring med programmering.

  Når flere Plaxis-analyser kjøres rett etter hverandre med et script utelukker dette muligheten til å inspisere Plaxis Output for hver enkelt modell. Dette kan føre til at feil i modellen ikke blir oppdaget.

  Det er utfordrende å bruke et script hvis man ikke har skrevet det selv. Det kan være hensiktsmessig å dele opp koden i flere moduler for å gjøre den mer oversiktlig. Det er også viktig å legge til forklarende kommentarer i koden.

5.2 Konklusjon og videre arbeid

Scriptet som er utarbeidet i denne oppgaven fungerer for utvalgte input-parametere, men scriptet har flere begrensninger. Scriptet kan brukes som utgangspunkt for videre arbeid med automatisering av spuntdimensjonering i Plaxis. Forslag til videre arbeid i scriptet:

- Bruke maksimalt moment som et kriterium i while-løkkene. Bytte til en stivere spuntprofil

når momentkapasiteten til spuntveggen overskrides.

- Bruke iterasjon for å finne optimal plassering av stag på spuntveggen for hver analyse.
- Bruke iterasjon for å bestemme stagtype, forspenning av stag og vinkel for staget.
- Deformasjon kan inkluderes som et kriterium i while-løkkene i scriptet.
- Sette inn flere stagrader dersom det er behov for det.
- I scriptet i denne oppgaven må brukeren legge inn input-verdier i *script.py* og i modulene. Det kan oppstå feil i scriptet når brukere som ikke har skrevet scriptet må gjøre endringer i koden. For å forbedre brukervennligheten i scriptet kunne man ha opprettet en Excel-fil der input-verdier føres inn i regneark som importeres til Plaxis (se kapittel 3.3.1). En annen mulighet er å skrive skrive kode som åpner menyer i Spyder der brukeren blir bedt om å velge input-parametere for analysen.
- Dersom scriptet skal brukes til dimensjonering av spuntvegg kunne det vært en idé å utforme scriptet slik at det gir et forslag til spuntprofil, spuntlengde og eventuell forankring etter at scriptet er kjørt. Det er mulig å lagre Plaxis-modeller underveis i kjøring av et script med kommandoen i g_i.save()(Grecu, 2018), og et alternativ kunne vært å lagre modellen med optimal design for spuntvegg og eventuelle forankringselementer.
- Dersom man ønsker å undersøke effekten av parametervariasjoner i spuntanalysen kan man legge til flere for-løkker i *script.py*.
- Variere grunnvannstanden i analysene og utføre strømningsanalyser.
- Kontrollere at msf i *Safety*-analysene konvergerer for å sikre mer nøyaktige verdier for msf.

Referanser

ArcelorMittal. (2020). *AZ (R) Sections*. Hentet 2021-02-03 fra https://sheetpiling.arcelormittal.com/products-services/production-range/az-sections/

Arne Engen, Baardvik, G., Kalsnes, B., Karlsrud, K., Lande, E.J., Langford, J., ... Veslegard, G. (2016). *Begrensning av skader som følge av grunnarbeider* (Teknisk rapport).

Bane NOR. (2017, mars). *Ski: Se fremdriften med Follobanen i bilder - Bane NOR*. Hentet 2021-01-03 fra https://www.banenor.no/Prosjekter/prosjekter/follobanen/ski/innhold/2017/ski-se-fremdriften-med-follobanen-i-bilder/

Bentley Communities. (2020a, desember). *ImportError: DLL load failed while importing _blowfish: The specified module could not be found - PLAXIS | SOILVISION Forum - PLAXIS | SOILVISION - Bentley Communities*. Hentet 2020-12-24 fra https://communities.bentley.com/products/geotech-analysis/f/plaxis-soilvision-forum/206542/importerror-dll-load-failed-while-importing-_blowfish-the-specified-module-could-not-be-found/623885#623885

Bentley Communities. (2020b). *Unable to assign platemat properties in commands runner - PLAXIS | SOILVISION Forum - PLAXIS | SOILVISION - Bentley Communities*. Hentet 2021-02-04 fra https://communities.bentley.com/products/geotech-analysis/f/plaxis-soilvision-forum/204740/unable-to-assign-platemat-properties-in-commands-runner/615994#615994

Bentley Systems. (2015a). *Identify Python commands from Plaxis command line*. Hentet 2020-06-05 fra https://www.plaxis.com/support/python-scripts/python-commands-reference/ (Library Catalog: www.plaxis.com)

Bentley Systems. (2015b, mai). *Plaxis Remote scripting*. Hentet 2020-03-11 fra https://www.plaxis.com/news/software-update/plaxis-remote-scripting/

Bentley Systems. (2020, mars). *Remote scripting: Error when defining normal stiffness of plate (EA) - PLAXIS | SOILVISION Forum - PLAXIS | SOILVISION - Bentley Communities*. Hentet 2021-01-11 fra https://communities.bentley.com/products/geotech-analysis/f/plaxis-soilvision-forum/194055/remote-scripting-error-when-defining-normal-stiffness-of-plate-ea/575922#575922

Brinkgreve, R. & Engin, E. (2010). Validation of empirical formulas to derive model parameters for sand. I *Numerical Methods in Geotechnical Engineeringe*.

Brinkgreve, R., Kumarswamy, S., Swolfs, W. & Foria, F. (2018a). *PLAXIS 2D Material Models Manual 2018*. Delft, Netherlands: PLAXIS Company.

Brinkgreve, R., Kumarswamy, S., Swolfs, W. & Foria, F. (2018b). *PLAXIS 2D Reference Manual*. Delft, Netherlands: PLAXIS Company.

Cepalia, A. (2021). *Scripts, Modules, Packages, and Libraries - RealPython*. Hentet 2021-01-20 fra https://realpython.com/lessons/scripts-modules-packages-and-libraries/

Choi, J.C., Engin, H.K. & Jostad, H.P. (2016). *L&L: Automation of Plaxis FEAs with python scripting*. (NGI)

Fredriksson, A., Kullingsjö, A., Rynder, A. & Stille, H. (2018). *Sponthandboken 2018*. Pålkommissionen.

Gerhardsen, S.M. & Guldstad, A.W. (2020, mars). *Hva er Python?* Hentet 2020-04-20 fra https://karriere360.no/artikler/hva-er-python/486403 (Library Catalog: karriere360.no)

Grande, L. (1998). Dimensjoneringsprinsipper - hensyn til regler og standardverk. Trondheim.

Grecu, S. (2018). Behaviour of Axially Loaded Bucket Foundations in Sand. , Appendix C. (Aalborg University)

Haugen, F.A. (2019, september). *Python-boka*. Universitet i Sørøst-Norge. Hentet 2020-05-07 fra https://home.usn.no/finnh/books/python_no/book/python_boka_2019_09_22_v01.pdf

Lekanger, K. (2019, november). *Github: Nå har Python gått forbi Java for første gang*. Hentet 2020-05-07 fra https://www.digi.no/artikler/github-na-har-python-gatt-forbi-java-for-forste-gang/478543 (Library Catalog: www.digi.no)

Mykleset, K.E. (2018). Vurdering av hjørneeffekter ved dimensjonering av små spuntgroper. (Publisher: NTNU)

Nordal, S. (2019). *Geotechnical Engineering Advanced Course*. Norwegian University of Science and Technology Geotechnical Engineering Group.

Oracle. (2019). *Control Flow Statements*. Hentet 2020-05-01 fra https://docs.oracle.com/javase/tutorial/java/nutsandbolts/flow.html

Python Software Foundation. (2020a). *About Python*. Hentet 2020-04-21 fra https://www.python.org/about/

Python Software Foundation. (2020b). *Built-in Functions — Python 3.7.7 documentation*. Hentet 2020-05-07 fra https://docs.python.org/3.7/library/functions.html

Rahman, M.S. & Ulker, M.B.C. (2018). *Modeling and Computing for Geotechnical Engineering: An Introduction*. CRC Press. (Google-Books-ID: JIuADwAAQBAJ)

Sindre, G. (2019). *Python: Løkker*. Hentet 2020-05-01 fra file:///C:/Users/Dina%20Erika%20Hansen/Downloads/U38-Python-GS-loops%20(1).pdf (NTNU)

Standard Norge. (1999). *Utførelse av spesielle geotekniske arbeider - Spuntvegger*.

Standard Norge. (2008a). *Eurokode 7: Geoteknisk prosjektering Del 2: Regler basert på grunnundersøkelser og laboratorieprøver NS-EN 1997-2:2007+NA:2008*. Oslo: Standard Norge.

Standard Norge. (2008b). *Eurokode 8: NS-EN 1998-1:2004+NA:2008 Prosjekteriung av konstruksjoner for seismisk påvirkning*. Oslo.

Standard Norge. (2016a). *Eurokode 0: Grunnlag for prosjektering av konstruksjoner*. Oslo.

Standard Norge. (2016b). *Eurokode 7: Geoteknisk prosjektering Del 1: Allmenne regler NS-EN 1997-1:2004+A1:2013+A1:2013+NA:2016*. Oslo.

Statens vegvesen. (2010a). Håndbok 016 Geoteknikk i vegbygging. , 624.

Statens vegvesen. (2010b). *Spuntfilm*. Statens Vegvesen. Hentet 2021-01-03 fra https://www.vegvesen.no/fag/teknologi/geofag/geoteknikk/spuntfilm

Statens vegvesen. (2018). *Håndbok V220: Geoteknikk i vegbygging*.

Systems, B. (2020). *Knowledge Base Articles Archive*. Hentet 2020-04-26 fra https://www.plaxis.com/support/

van der Sloot, M. (2019). *PLAXIS Command Line and Scripting for Automated Model Definition*. Hentet 2020-04-27 fra https://learn.bentley.com/app/VideoPlayer/LinkToIndividualCourse?LearningPathID=113626&CourseId=129064&MediaID=5018734 (Bentley Systems)

van Rossum, G. & Warsaw, B. (2013). *PEP 8 – Style Guide for Python Code*. Hentet 2021-01-25 fra https://www.python.org/dev/peps/pep-0008/

Tillegg A

Akronymer

API Application Programming Interface

FDM Finite Difference Method

FEM Finite Element Method

GUI Graphical User Interface

HS Hardening Soil

Tillegg B

Konfigurasjon av remote scripting server i Plaxis

Plaxis remote scripiting muliggjør kommunikasjon mellom Python og Plaxis. For at de to programmen skal kunne kommunisere må det settes opp en server i Plaxis og koble Python til serveren.

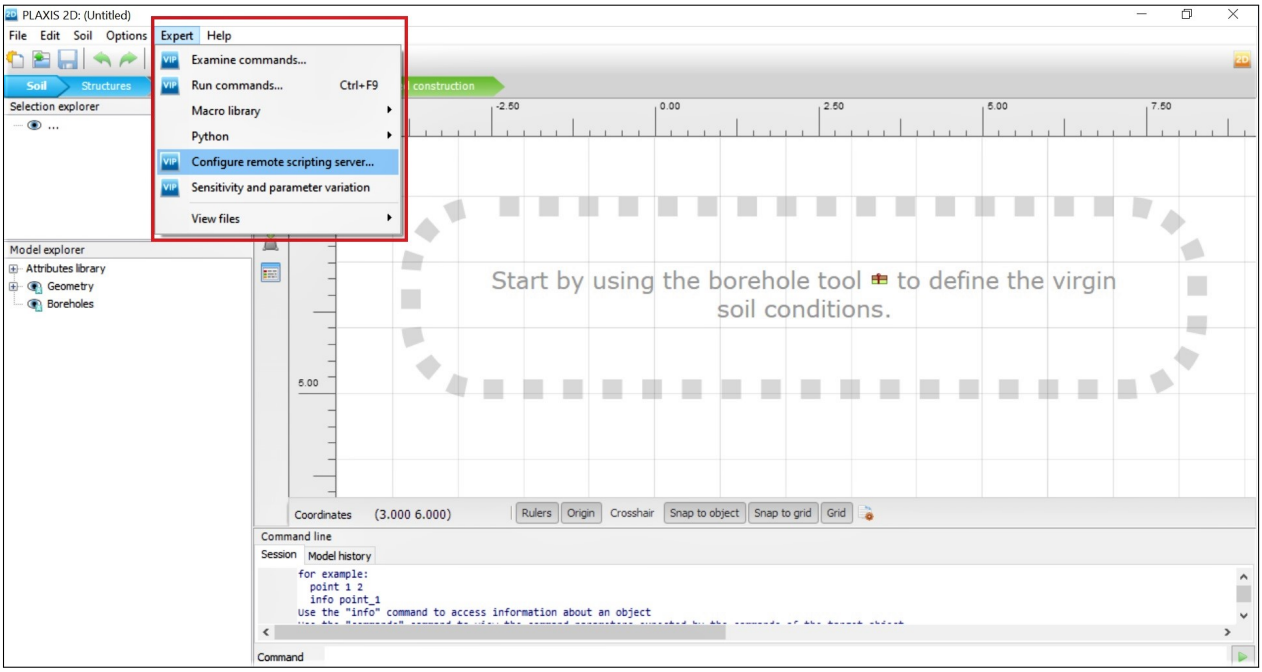
B.1 Forutsetninger for remote scripting i Plaxis

For å kunne benytte remote scripting i Plaxis må følgende betingelser være oppfylt:

- Plaxis VIP lisens.
- Plaxis 2015 eller nyere versjoner av Plaxis.
- En kodeeditor der Python-koden skrives. Eksempler på teksteditorer er Scitep, Spyder, Pycharm og så videre.
- Grunnleggende ferdigheter i Python er anbefalt.
- Tilgang til internett.
- Brannmuren må ikke hindre at Plaxis har tilgang til internett eller at andre programmer kommuniserer med Plaxis.

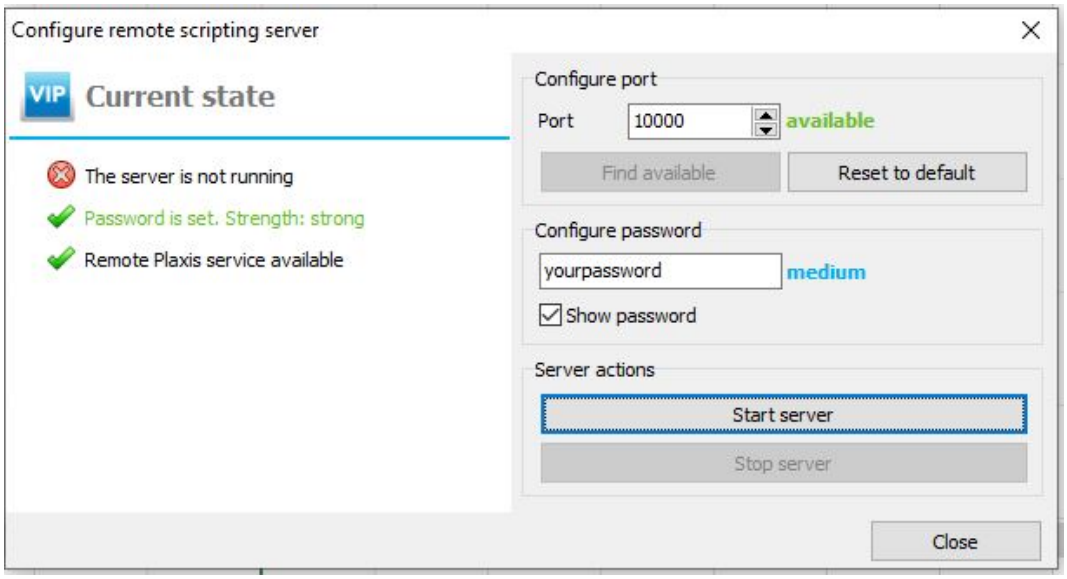
B.2 Konfigurasjon av remote scripting server i Plaxis

For å kunne kjøre et script i Plaxis må det settes opp en remote scripting server. Denne serveren muliggjør kommunikasjon mellom Python og Plaxis. For å sette opp Remote scripting server i Plaxis må man gå inn i menyen *Expert* og velge *Configure remote scripting server*, som vist i figur B.1.



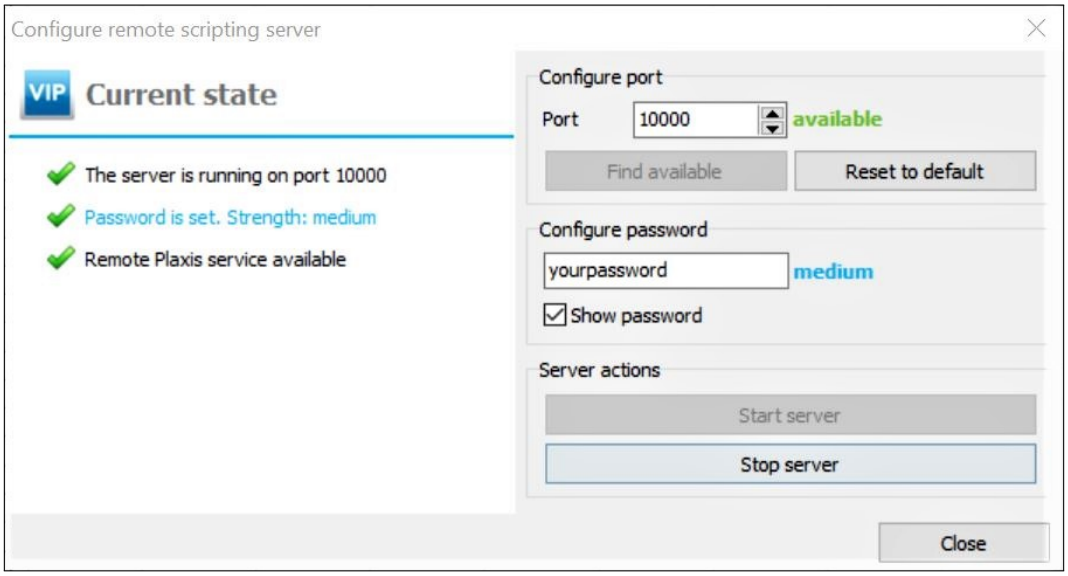
Figur B.1: Velg Expert>Configure remote scripting server.

Det vil nå åpnes et nytt vindu, som vist i Figur B.2. I dette vinduet velges en ledig local port og et passord. Passordet er nødvendig for å opprette en sikker kobling mellom Python og Plaxis.



Figur B.2: Konfigurasjon av remote scripting server. Bildet gjelder for Plaxis 2D 2017 og Plaxis 3D 2017, samt nyere versjoner.

Velg *Start server* for å aktivere remote scripting server i Plaxis. Dersom vinduet ser ut som bildet i figur B.3 er aktiveringen av serveren vellykket, og Python kan nå kobles til Plaxis.



Figur B.3: Remote scripting server i Plaxis er aktivert.

Tillegg C

Boilerplate code

I programmering er en boilerplate code en seksjon med kode som gjenbrukes flere steder, og kan ses på som en slags standard. Hensikten med en boilerplate code er å spare tid på å skrive ned kode som er brukt flere ganger tidligere, slik at man kommer raskt i gang med kodingen.

I starten av alle scripts som skal kjøres i Plaxis er det nødvendig å ha med en seksjon med kode som kalles Boilerplate code. Boilerplate code sørger for at Python kobles til serveren som er opprettet i Plaxis. Figur C.1 viser et eksempel på en boilerplate code. Boilerplate code kan ha små variasjoner i ulike script, men består hovedsakelig av de samme kommandoene. I masteroppgaven *Behaviour of Axially Loaded Bucket Foundations in Sand* (Grecu, 2018) er boilerplate code bygd opp litt forskjellig sammenlignet med eksempelet i denne oppgaven, men har den samme funksjonen.

```
1 #Definerer system path for scripting libraries:
2 import sys
3 sys.path.append(r"C:\Program Files\Plaxis\PLAXIS 2D\python\Lib\site-packages")
4 from plxscripting.easy import*
5
6 #Definerer local port number (portnummer):
7 localport_input=10000
8 localport_output=10001
9
10 #Kobler til remote scripting server i PLAXIS:
11 s_i, g_i = new_server('localhost', localport_input, password = 'yourpassword')
12 s_o, g_o = new_server('localhost', localport_output, password = 'yourpassword')
```

Figur C.1: Boilerplate code

Linje 2 og 3 i figuren ovenfor har som funksjon å definere programstien (system path) til biblio-

teket *plxscripting*. Ved å definere programstien til dette biblioteket i scriptet, "forteller" man Python hvor denne mappen ligger, slik at moduler i *plxscripting* kan importeres og tas i bruk i Python scriptet. Programstien angir altså plasseringen av plxscripting på datamaskinen, se figur C.2. Plxscripting lastes automatisk ned når Plaxis lastes ned.

Figur C.2: Lokasjon for *plxscripting* på datamaskinen

På linje 4 i figur C.1 importeres modulen *easy* som ligger i plxscripting-biblioteket. Ved å bruke kommandoen *import ** importeres alle metodene i *easy*.

Ved å benytte funksjonen *new_server* på linje 11 og 12 kobles scriptet til remote scripting server i Plaxis slik at Input og Output i Plaxis kan kontrolleres gjennom kommandoer i scriptet. Funksjonen *new_server* er importert fra modulen *easy*. Input og Output må ha ulike portnummere (localport_input og localport_output), og i denne oppgaven er disse satt til å være 10001 og 10000. Det er viktig at portnummeret for Input samsvarer med feltet i *Configure remote scripting server*, se figur C.1. Dette gjelder også passordet.

I koden på linje 11 og 12 defineres fire variabler som benyttes i kommandoene i scriptet. Variabler som kontrollerer Input:

- **s_i**: kontrollerer kommandoer relatert til kontroll av filer, benyttes for eksempel for å åpne en fil.
- **g_i**: kontrollerer kommandoer relatert til selve modellen i Plaxis, for eksempel å opprette et borehole, generere mesh, lage materialer og så videre.

Variablene som kontrollerer Output:

- **s_o**: kontrollerer kommandoer relatert til av filer, for eksempel å lukke en fil.
- **g_o**: kontrollerer kommandoer relatert til selve modellen i Plaxis, for eksempel hente ut resultater, lage plots og så videre.

Det er viktig å lagre scriptet som en .py-fil slik at den blir identifisert som en Python-fil.

Tillegg D

Resultater

Tabellene på neste side viser utklipp fra Excel-filene *UtkragetSpuntvegg.xlsx* og *SpuntveggMedStag.xlsx*

Resultater – utkraget spuntvegg

Prosess i flytskjema	Gravedybde [m]	Jordmaterialer	Spuntprofil	Mesh	Spuntlengde [m]	ErrorCode, Plastic	Mmax [kNm/m] spunt, Plastic	Nmax [kN/m] spunt, Plastic	Utot [m] spunt, Plastic	Msf, Safety	Mmax [kNm/m] spunt, Safety	Nmax [kN/m] spunt, Safety	Utot [m] spunt, Safety
5	-4	LooseSand	AZ12700	very coarse	8	0	85,34239591	-16,21227654	0,086709998	1,083898333	108,9434595	-17,59199592	0,17777424
5	-5	LooseSand	AZ12700	very coarse	10	0	171,5764956	-23,83504864	0,184138387	1,068896539	215,8111664	-25,8908748	0,322662093
5	-6	LooseSand	AZ12700	very coarse	12	0	303,0796352	-33,13920857	0,375389401	1,048825539	360,405655	-34,88717021	0,568968737
8	-7	LooseSand	AZ12700	very coarse	14	103	399,857017	-41,55795885	0,492552491	<1	-	-	-
8	-8	LooseSand	AZ12700	very coarse	16	103	371,3749284	-52,96774805	0,426722741	<1	-	-	-
8	-9	LooseSand	AZ12700	very coarse	18	103	407,183599	-64,99517754	0,48564131	<1	-	-	-
8	-10	LooseSand	AZ12700	very coarse	20	103	182,9664558	-80,23860525	0,18537312	<1	-	-	-

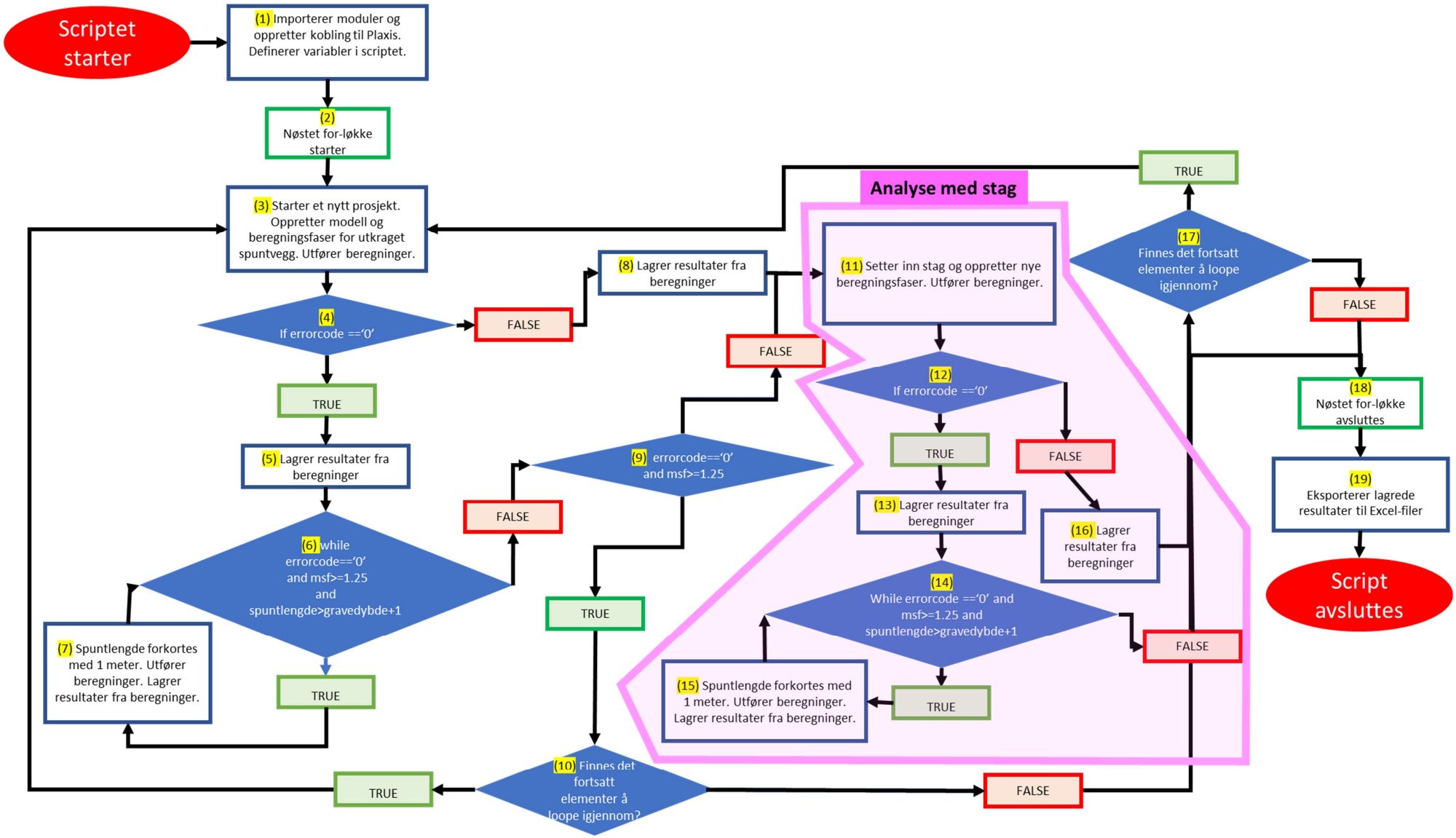
Resultater – forankret spuntvegg

Prosess i flytskjema	Gravedybde [m]	Jordmaterialer	Spuntprofil	Mesh	Spuntlengde [m]	Forspenning stag	ErrorCode, Plastic	Mmax [kNm/m] spunt, Plastic	Nmax [kN/m] spunt, Plastic	Nmax anchor [kN/m], Plastic	Utot [m] spunt, Plastic	Msf, Safety	Mmax [kNm/m] spunt, Safety	Nmax [kN/m] spunt, Safety	Nmax anchor [kN/m], Safety	Utot [m] spunt, Safety
13	-4	LooseSand	AZ12700	very coarse	8	32	0	-23,41306885	-16,71654455	47,37874069	0,003794586	2,20071318	-118,9621588	-11,77266417	142,5328912	13,17612439
15	-4	LooseSand	AZ12700	very coarse	7	32	0	-23,59816655	-15,72803032	47,89536303	0,003831601	1,876230693	-74,12999638	-10,45309568	119,763111	67,17685968
15	-4	LooseSand	AZ12700	very coarse	6	32	0	-23,03604445	-12,83784178	48,90972477	0,003922459	1,537173723	-41,82951089	-10,14430783	98,70386851	49,84913731
15	-4	LooseSand	AZ12700	very coarse	5	32	0	15,92228417	-7,096090136	54,6096182	0,005346856	1,197377257	25,42608294	-5,499082283	81,43337803	35,34331263
13	-5	LooseSand	AZ12700	very coarse	10	32	0	-44,39206533	-29,15168578	60,91531366	0,006319748	2,109946913	-250,6018793	-17,71139523	206,5548701	6,323430402
15	-5	LooseSand	AZ12700	very coarse	9	32	0	-45,60234424	-28,21002403	62,22142428	0,006292213	1,850890465	-179,2302927	-18,2356437	179,2976455	35,99656158
15	-5	LooseSand	AZ12700	very coarse	8	32	0	-47,7316314	-25,22299153	64,03364565	0,006105045	1,581509893	-129,4323443	-17,91629535	150,634464	39,61363106
15	-5	LooseSand	AZ12700	very coarse	7	32	0	-46,81595363	-19,82407441	66,58946052	0,005997434	1,311560777	-72,72631181	-12,28601866	126,2884389	73,50651749
15	-5	LooseSand	AZ12700	very coarse	6	32	101	-30,57417466	-7,274541792	32	0,041012091	<1	-	-	-	-
13	-6	LooseSand	AZ12700	very coarse	12	32	0	-69,6461174	-42,90638525	79,59744902	0,011738886	2,07059165	-424,1217948	-16,12590016	278,5428493	0,384123291
15	-6	LooseSand	AZ12700	very coarse	11	32	0	-70,55365187	-42,33921157	80,91068627	0,011794371	1,845687753	-345,5947849	-21,04703629	249,6837386	2,936301452
15	-6	LooseSand	AZ12700	very coarse	10	32	0	-73,20022992	-40,3678829	83,82426739	0,012027191	1,62713176	-261,9415642	-22,36352439	216,9651607	19,78135802
15	-6	LooseSand	AZ12700	very coarse	9	32	0	-77,60097172	-36,39463888	86,70664625	0,012080213	1,389835207	-191,3509396	-22,83312215	184,5459774	42,95122607
15	-6	LooseSand	AZ12700	very coarse	8	32	0	-76,76206656	-26,70422928	93,39544801	0,013223107	1,162137985	-117,5572948	-20,03863112	157,2799534	72,85346007
13	-7	LooseSand	AZ12700	very coarse	14	32	0	-101,0844884	-55,18179864	103,2818919	0,020885249	1,961007472	-473,3698893	-51,5931553	263,2626029	0,181202239
15	-7	LooseSand	AZ12700	very coarse	13	32	0	-101,3815449	-54,75578408	104,0811579	0,020890417	1,85314365	-527,9853239	-19,64990203	312,9077844	0,319009729
15	-7	LooseSand	AZ12700	very coarse	12	32	0	-103,0244314	-54,08028375	106,351036	0,02122371	1,663298144	-436,994505	-21,3354661	293,605691	0,508398586
15	-7	LooseSand	AZ12700	very coarse	11	32	0	-108,8708733	-52,81560923	110,8727766	0,022313354	1,452967278	-335,0649601	-29,0041712	255,4627141	7,774658811
15	-7	LooseSand	AZ12700	very coarse	10	32	0	-115,5385079	-49,16991745	114,2337048	0,0226802	1,261313905	-262,5583732	-29,68742156	223,2590137	54,68574492
15	-7	LooseSand	AZ12700	very coarse	9	32	101	-124,3172428	-26,90588208	32	0,037421166	<1	-	-	-	-
13	-8	LooseSand	AZ12700	very coarse	16	32	0	-141,5357012	-68,48187732	131,4067453	0,035533226	1,705539038	-510,0500261	-73,55345175	277,6408477	0,216236884
15	-8	LooseSand	AZ12700	very coarse	15	32	0	-141,9563724	-68,2530571	132,2082675	0,035641149	1,826754039	-657,1291497	-58,30492259	322,6414073	0,304041271
15	-8	LooseSand	AZ12700	very coarse	14	32	0	-143,038204	-67,63775883	134,6882459	0,036136221	1,677545891	-608,0032664	-24,20469725	334,1263833	0,311217161
15	-8	LooseSand	AZ12700	very coarse	13	32	0	-147,8618382	-67,22434022	139,0259457	0,037818035	1,513640699	-567,0979722	-22,54828324	342,9603866	0,639746136
15	-8	LooseSand	AZ12700	very coarse	12	32	0	-157,3809701	-66,06620914	143,4582633	0,039990664	1,329107358	-467,0594293	-36,05790471	304,6005944	3,514866084
15	-8	LooseSand	AZ12700	very coarse	11	32	0	-167,5395901	-61,66757547	148,9514737	0,041215413	1,148792354	-356,4656273	-33,08487721	263,6772959	59,75481573
13	-9	LooseSand	AZ12700	very coarse	18	32	0	-193,8973493	-83,67168508	163,460567	0,058871981	1,560360808	-565,2816338	-93,41449539	293,604814	0,258987897
15	-9	LooseSand	AZ12700	very coarse	17	32	0	-193,7872354	-83,47304865	163,5624717	0,058689621	1,590261323	-649,5152385	-90,57586835	320,3071586	0,321337189
15	-9	LooseSand	AZ12700	very coarse	16	32	0	-194,4670938	-83,0717934	165,1296047	0,058993235	1,606884283	-699,3291275	-84,60620505	329,401764	0,347324775
15	-9	LooseSand	AZ12700	very coarse	15	32	0	-197,2519869	-82,86946633	168,7203906	0,06061204	1,505526514	-617,6226877	-76,64116557	314,7780884	0,285806116
15	-9	LooseSand	AZ12700	very coarse	14	32	0	-207,3695545	-82,56524028	175,0198461	0,064977099	1,393533624	-664,7773375	-28,71009765	376,4026613	0,530215285
15	-9	LooseSand	AZ12700	very coarse	13	32	0	-224,0304939	-81,19819422	180,9749053	0,069758185	1,233187026	-582,8231319	-39,97194187	348,8528537	2,527779194
13	-10	LooseSand	AZ12700	very coarse	20	32	0	-256,1413393	-100,6320844	197,3752237	0,093243709	1,477133047	-667,363424	-112,0565794	321,3755032	0,33730143
15	-10	LooseSand	AZ12700	very coarse	19	32	0	-257,0422362	-100,7955406	198,0515962	0,093581138	1,463657805	-698,3015454	-113,8864972	330,5031052	0,372848781
15	-10	LooseSand	AZ12700	very coarse	18	32	0	-254,8863637	-100,0986181	198,2591658	0,092748083	1,394108341	-649,280675	-111,8834827	321,6152526	0,336418208
15	-10	LooseSand	AZ12700	very coarse	17	32	0	-258,4765519	-100,2159854	201,2688186	0,094635687	1,406608469	-719,1009399	-106,2617608	345,0967229	0,388981764
15	-10	LooseSand	AZ12700	very coarse	16	32	0	-266,3927324	-100,0536952	207,1199252	0,099844202	1,39288982	-749,2580087	-92,01332365	355,1949797	0,400765663
15	-10	LooseSand	AZ12700	very coarse	15	32	0	-285,9847139	-100,0596675	215,9043413	0,109877805	1,304351212	-757,5510275	-36,77521411	398,2959038	0,525333613
15	-10	LooseSand	AZ12700	very coarse	14	32	0	-323,941033	-94,19408244	230,8596031	0,123968864	1,153503319	-736,1682765	-58,37149462	401,0676184	4,973081051

Tillegg E

Flytskjema

Flytskjemaet på neste side viser programflyten for scriptet som er utarbeidet i oppgaven.


Scriptet starter
(1) Importerer moduler og oppretter kobling til Plaxis. Definerer variabler i scriptet.
(2) Nøstet for-løkke starter
(3) Starter et nytt prosjekt. Oppretter modell og beregningsfaser for utkraget spuntvegg. Utfører beregninger.
(4) If errorcode ==‘0’
FALSE
TRUE
(5) Lagrer resultater fra beregninger
(6) while errorcode==‘0’ and msf>=1.25 and spuntlengde>gravedybde+1
FALSE
TRUE
(7) Spuntlengde forkortes med 1 meter. Utfører beregninger. Lagrer resultater fra beregninger.
(8) Lagrer resultater fra beregninger
(9) errorcode==‘0’ and msf>=1.25
FALSE
TRUE
(10) Finnes det fortsatt elementer å loope igjennom?
TRUE
FALSE
Analyse med stag
(11) Setter inn stag og oppretter nye beregningsfaser. Utfører beregninger.
(12) If errorcode ==‘0’
TRUE
FALSE
(13) Lagrer resultater fra beregninger
(14) While errorcode ==‘0’ and msf>=1.25 and spuntlengde>gravedybde+1
TRUE
FALSE
(15) Spuntlengde forkortes med 1 meter. Utfører beregninger. Lagrer resultater fra beregninger.
(16) Lagrer resultater fra beregninger
(17) Finnes det fortsatt elementer å loope igjennom?
TRUE
FALSE
(18) Nøstet for-løkke avsluttes
(19) Eksporterer lagrede resultater til Excel-filer
Script avsluttes