Delft University of Technology

Faculty of Civil Engineering and Geosciences
CIEM5060-09 Master Thesis

A study on the effect of soil thickness variation of overconsolidated clays on the secondary consolidation of the soil underneath immersed tunnel elements

Specified on the Fehmarnbelt-tunnel

Author:
J.M.J. van Beek 4480783

Tuesday 10th June, 2025

Preface

This thesis explores secondary consolidation phenomena beneath immersed tunnel elements in overconsolidated clay layers, specifically focusing on the Fehmarnbelt Tunnel. The whole model used will be deconstructed and explained in detail before proceeding to the final results.

This subject sparked my interest because I always liked structures as a child. In the third grade of VWO I knew for sure that Civil Engineering would be the perfect study for me. Being busy with little "puzzles" to solve the stability of structures has always been interesting to me. During my study years in Delft, I made a side-track to Applied Earth Sciences, deepening my knowledge about the earth and its amazing qualities. The combination between these two interests is: The tunnel. I really liked to be busy investigating the tunnel as the structure itself, the interaction between the tunnel and the soil, and the research on the reaction of the soil over time.

This would not have been possible without RHDHV and the guidance I received within the company from ir. C.M.P. 't Hart. Not only did he guide me through this thesis, but also guided me during my internship, that was prior to this thesis. Thanks to Marcel, I really learned a lot about the field in which I did research and about the topics I researched. Furthermore, I would like to thank my thesis committee, Dr. ir. W. Broere and Dr. P. Mares Nasarre for their patience. guidance and positive criticism on my thesis. Last but not least, I would like to thank all my friends who helped me through the difficult parts of this thesis, by studying together with me and proofreading a large part of this thesis.

Abstract

This thesis presents a study on the impact of soil thickness variability on secondary consolidation of overconsolidated glacial soils beneath immersed tunnel elements, specifically focusing on the Fehmarnbelt Tunnel. The research employs a combination of analytical models, including the Timoshenko beam on Kerr foundation model (TBKF model), the Conte and Troncone method (2006) and the Feng et al. (2020) method, to analyze the initial, primary, and secondary consolidation phases, respectively. The entire tunnel is subdivided into specific zones that represent different combinations of soil types and tunnel configurations to make the model less complex but still grasp the influence on the complete tunnel. This subdivision is shown in Figure 1.

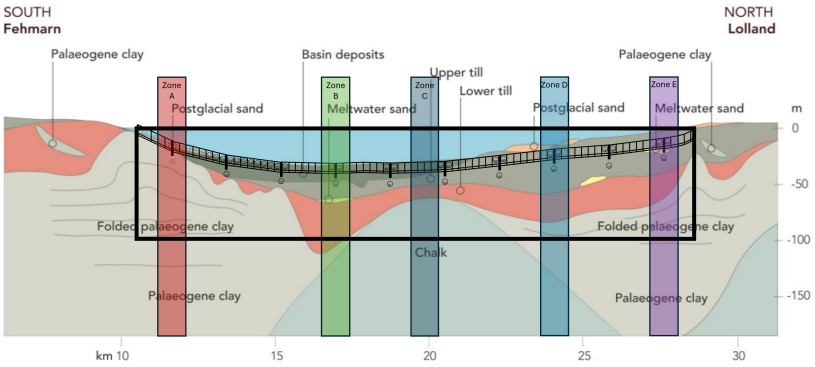


Figure 1: Zones of interest Fehmarnbelt tunnel (Yumpu.com 2011)

The composition of the soil columns of these zones is given, but the thickness is not. The thicknesses of different soil columns are generated by a Monte Carlo simulation following a doubly truncated lognormal distribution for the sampling of the soil thicknesses. This distribution is limited by the lower bound of 0.1 m in thickness and an upper bound of the remaining height in the soil column, which is especially important in the lowest soil layer in the soil column. This generation of different soil columns is then fed to the complete model to generate outcomes.

The key findings are as follows.

Initial Deformation: The initial deformation of the interaction between the soil and the structure is influenced by the composition of the soil and the type of tunnel element (regular or special). The variability in soil thickness has a minimal impact on initial deformation, concluding form the range of initial deformation values of 0 to -2.2 millimeters for all zones. This means that the initial deformation does not contribute significantly to the overall deformation profile.

Primary Consolidation: Primary consolidation is influenced by the length of the drainage path, the coefficient of consolidation, and the oedometer stiffness of the soil. Primary consolidation is significantly affected by the variability of the randomly generated soil thickness, especially during the phase where soil is removed from the soil column. However, the phase in which load is added to the soil column shows a decrease in variability over time. Zone D and Zone E, with the most cohesive soil layers and only permeable boundaries at the top and bottom of the soil column, show the widest range in primary consolidation values at 2190 days. The primary consolidation phase continues until the point where all excess pore pressure is dissipated, leading to the onset of secondary consolidation.

Secondary Consolidation: Secondary consolidation is dominated by the equivalent time parameter,

the end of primary consolidation, and the compression, swell, and creep index of the soil. Secondary consolidation is highly sensitive to variations in the randomly generated thickness of the soil layers. The analysis highlights the importance of accurate soil thickness estimation, particularly in zone B due to the presence of the basin deposits layer, the removal of this disturbing layer could help to improve the certainty in the results of zone B. Overall, variations in the randomly generated thickness of cohesive soil layers present within a soil column with only cohesive layers were found to significantly influence the secondary consolidation process. This highlights the importance of accurately determining both the soil properties and the layer thicknesses. The study reveals that secondary consolidation can range from heaving to settling, with a standard deviation of 1 meter in randomly generated soil layer thickness that causes the range of results from 54 millimeters of heaving to 122 millimeters of settling over all zones.

Impact of Soil Layer Thickness: The research underscores the critical role of soil layer thickness in predicting long-term deformation and ensuring the stability and serviceability of immersed tunnels. The findings indicate that variations in randomly generated soil thickness have a significant impact on the secondary consolidation process, emphasizing the need for accurate estimation of soil parameters and thickness. The study concludes that even a change of 0.5 meters in the standard deviation of randomly generated soil thicknesses can greatly increase the range in the resulting settlement or heave. In settlement ranging from 48 mm to 98 mm for regular elements in Zone B, or from 43 mm of heave to 34 mm of settlement for special elements in Zone D.

Recommendations for future research in this case study: The thesis recommends investigating the effects of cyclic loading, such as tidal loads and back siltation, to provide a more realistic representation of deformation values. Furthermore, a parameter sensitivity analysis and model limitations analysis are required to understand the impact of errors in soil parameter estimation on the results. Future research should also explore the interaction between individual tunnel parts and the effects of tunnel elements rotating relative to each other.

The upper limit of the doubly truncated lognormal distribution can be neglected to investigate whether the threshold that occurs when increasing the standard deviation still exists. The standard deviation was chosen to be the same for every soil type; in further research, it can be chosen to vary this standard deviation for every soil type to investigate the uncertainty and risk of a specific soil type with greater precision.

In conclusion, this study contributes to a better understanding of the behavior of the soil beneath immersed tunnels and informs more accurate predictions and designs in future projects. By addressing soil regions and layers that are highly susceptible to a wide range of variation in secondary consolidation, engineers can take measures to improve the durability and safety of tunnel structures, leading to more resilient infrastructure.

Symbols

β:	Pore-water compressibility	$[-]$
$\Delta\epsilon$:	Change in strain	$[-]$
$\Delta\sigma$:	Change in stress	$[kPa]$
$\Delta\sigma^{'}_{av}$:	Average increase in effective pressure on the clay layer caused by the construction of the foundation	$[kPa]$
Δe :	Change in void ratio	$[-]$
ΔL :	Change in length	$[m]$
Δt:	Change in time	$[s]$
ε:	Strain	$[-]$
η:	Parameter accounting for compressibility of the soil and pore fluid	$[-]$
γ_w:	Unit weight of water	$[kN/m^3]$
κ:	Curvature of the neutral axis	$[-]$
ν_s:	Poisson's ratio	$[-]$
n:	Porosity	$[-]$
ρ:	Density of the fluid	$[kN/m^3]$
σ:	The external force applied on the soil distributed over the cross-sectional area	$[kPa]$
σ':	Effective pressure	$[kN/m^2]$
σ_0:	Average load over the period	$[kPa]$
σ'_0:	Initial stress state	$[kPa]$
σ'_p or $\sigma^{'}_c$:	Preconsolidation pressure	$[kPa]$
$\phi(x)$:	Rotation angle of the cross section	$[rad]$
ψ:	Rotation of the beam	$[rad]$
ω:	Circular frequency	$[rad/s]$
A and B:	Load amplitudes	$[kPa]$
B:	Width of the tunnel element	$[m]$
c:	Compression stiffness of the upper layer in Kerr foundation	$[kN/m]$
$C_{\alpha e}$:	Coefficient of secondary compression	$[-]$
C_c:	Compression index	$[-]$
C_e:	Swelling index	$[-]$
C_v:	Coefficient of consolidation	$[m^2/s]$
C_1 - C_4:	Integration constants for the TBKF model	$[-]$
e:	Void ratio	$[-]$
e_0:	Initial void ratio	$[-]$
e_{av}:	Average void ratio during consolidation	$[-]$
E_{eq}:	Equivalent Oedometer stiffness	$[kPa]$
E_{oed}:	Oedometer stiffness	$[kPa]$
E_s:	Elastic modulus of the foundation	$[kPa]$
EI:	Bending stiffness of the element	$[kNm^2]$
F:	The transfer matrix that depends on the shape of f_i	$[-]$
F_b:	Buoyant force	$[kN]$
F_{ex}:	The applied external force	$[kN]$
f_s:	The non-uniform distribution coefficient of the shear stress in the section	$[-]$
G:	Shear stiffness of the shear layer	$[kN/m^2]$
g:	Gravitational constant	$[m/s^2]$
G_s:	Shear stiffness of the soil	$[kPa]$
GA:	Shear stiffness of the element	$[kNm^2]$

H:	Thickness of the compressible soil layer	$[m]$
H_c:	Thickness of the clay layer	$[m]$
H_d:	The length of the drainage path	$[m]$
h_i:	Respective height of soil layer	$[m]$
H_s:	Thickness of the foundation	$[m]$
j:	The loading step numbering	$[-]$
k:	The layer numbering	$[-]$
k_h:	Hydraulic conductivity	$[m/s]$
k_s:	Compression stiffness of the lower layer in Kerr foundation	$[kN/m]$
k_w:	The coefficient of permeability	$[m/s]$
L_0:	Original length	$[m]$
M:	$[(2m+1)\pi]/2$	$[-]$
m:	An integer = 1,2,...	$[-]$
m_v:	Coefficient of volume compressibility	$[1/kPa]$
$M(x)$:	Moment in beam	$[kNm]$
OCR:	Overconsolidation ratio of the soil	$[-]$
$q(x)$:	Uniform load applied to beam	$[kPa]$
$S_{consolidation,j}$:	The consolidation under j-th loading	$[m]$
$S_{creep,dj,k}$:	The delayed creep consolidation due to the coupling of the excess pore water pressure	$[m]$
$S_{creep,fj,k}$:	The creep consolidation with respect to the final j-th effective stress	$[m]$
$S_{creep,j}$:	Creep consolidation under the j-th loading	$[m]$
S_s:	Secondary consolidation	$[m]$
S_{totalB}:	The total consolidation based on Hypothesis B	$[m]$
$s_k(t)$:	Deformation due to each harmonic component	$[m]$
T:	Period of the load	$[s]$
t:	Time	$[s]$
t_0:	Creep parameter in units of time (1 day in this research Wei-Qiang Feng et al. 2020)	$[day]$
t_1,t_2:	Times at the begin and end of secondary consolidation period	$[days]$
t_{cj}:	Construction period of load	$[days]$
$t_{ej,k}$:	Equivalent time for a soil	$[-]$
t_{EOP}:	The time when all excess pore pressures are dissipated	$[days]$
t_j:	Duration up until next load or till the end of consolidation	$[days]$
T_v:	Non-dimensional time factor	$[-]$
U:	The degree of consolidation	[%]
$u_k(z,t)$:	Pore-water pressure at each harmonic component	$[kPa]$
$\bar{u}(z,t-\tau)$:	The solution when the loading rate is kept at unity	$[kPa]$
V_G:	Shear force of the shear layer	$[kN]$
V_s:	Volume of solids	$[m^3]$
V_v:	Volume of pores	$[m^3]$
V_w:	Volume of water replaced	$[m^3]$
v_z:	Rate of water flow across a unit area of soil in the z direction	$[m^2/s]$
$V(x)$:	Shear force in beam	$[kN]$
$w(x)$:	Deformation of beam	$[m]$
w_k:	Deformation of the shear layer	$[m]$
x:	Distance from left end of beam	$[m]$
y_0:	The vector of initial parameters	$[-]$
y_p:	The vector that reflects the influence of the external forces	$[-]$
z:	Depth	$[m]$
$\frac{dw}{dx}-\phi(x)$:	Shear angle of the cross-section due to shear deformation	$[rad]$

Contents

1 Introduction

The first immersed tunnel, made up of several segments that were sunk to the bottom of the river, was built in 1869 (2010). Since then the immersed tunnel has developed significantly. Nowadays the segments are constructed in a dry dock and towed on the water to the location where they are submerged and placed upon their foundation layer which was prepared in a pre-dredged gully at the bottom of the river/sea. During the last years, the design and techniques to construct an immersed tunnel (IMT) have developed greatly. However, there are still challenges, such as the accumulation of differential settlements, the long-term serviceability of the tunnel, and the specific design to withstand seismic activity in the long term. The dangers of these challenges are the development of cracks or the movement of parts of the tunnel upward, causing additional forces on the connecting parts of the segments (2022).

One of the challenges is secondary soil consolidation (creep), the time-dependent deformation of the soil under sustained load, which can affect the structural performance of the tunnel over time, especially when subjected to cyclic load or long-term pressure (Zhang and Broere 2019). In current research, no answer can be found on the question of where this creep comes from, only some theories exist. (Szavits-Nossan 2015) There are, however, models that could predict this secondary consolidation based on Terzaghi's theory. Secondary soil consolidation poses a challenge, especially in combination with uncertainty in soil thickness. This uncertainty comes from the heterogeneity of soil over a large area, measurement errors, interpolation, and extrapolation errors. All types of error are explained in more detail in (Uzielli et al. 2006), (Dan et al. 2023), and (Ding et al. 2024), respectively. This uncertainty in soil thickness could contribute to a larger secondary consolidation, which could have an impact on the structural performance of the tunnel over time and should be taken into account in the designing phase of the immersed tunnel.

1.1. Research context

The focus of this research will lie on the challenge of differential settlement and specifically the secondary consolidation (creep) that takes place over time. In the design phase of several IMT-projects, secondary consolidation is defined making use of a deterministic approach to predict its effect over the course of a longer time span (e.g. 50 years) (Egeli and Kartaltepe 2012). This design approach is necessary, because tunnels on extremely soft and plastic soils undergo more uneven settlements than suspected during the design phase according to (Heijden 2023) and (Gavin et al. 2019). A representative example of this statement is the Kil Tunnel investigated by (Gavin et al. 2019). The settlement of the tunnel presented in this research is shown in Figure 1.1.

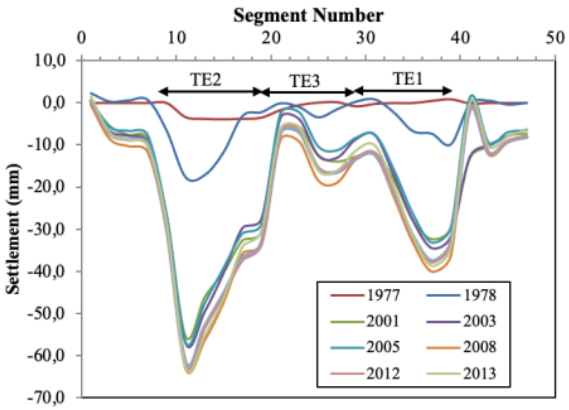


Figure 1.1: Settlements of the Kil-tunnel over time by (Gavin et al. 2019)

Currently more data is available on cases like the Kil Tunnel and the effect of creep can be measured (Grantz 2001). It is known to have an impact on the (differential) settlement, which is suggested to be caused by the variation in the thickness and the type of foundation. In addition, soil variability could worsen or better (depending on the covariance length and the length of the element) differential settlements('t Hart, Morales-Nápoles, and Jonkman 2024).

Overall, large differential settlements could cause serious cracks or deformations of the elements in the IMT and research on the described effect could improve the design made for these types of tunnels and limit the costs of soil / foundation / tunnel improvement that need to be undertaken when differential settlements exceed the serviceable limit state. The Fehmarnbelt Tunnel is used here as a case study to serve as a practical framework in which research can be carried out. The Fehmarnbelt Tunnel is an IMT that is constructed between Germany and Denmark and will be (when finished) the largest IMT worldwide.

The soil beneath the Fehmarnbelt tunnel consists of highly plastic and soft soils (Kammer et al. 2012) that need a long time to consolidate, which could cause large secondary consolidation and differential settlements throughout the length of the tunnel (Zhang and Broere 2019). This could pose a risk to the serviceability of the tunnel in the future and must be covered correctly in the designing and construction phase of the tunnel. In addition to this problem, a large variability in soil over the length of the tunnel or a large measurement error could increase the risk (Noor and Daud 2016). The question remains as to how large these discrepancies caused by measurement errors or soil variability may be.

1.2. Research problem

Research has been conducted on the impact of spatial variability of subsoil stiffness on immersed tunnels by (Wu 2017), on the bearing capacity of shallow foundations by (Daryani and Mohamad 2015), and on the settlement in the longitudinal direction of the IMT by (Y. Wang et al. 2023) and (Tang et al. 2023). (Heijden 2023) highlighted the risks that (differential) settlements pose to the serviceability of immersed tunnels. This will be covered in Section 2.4 of Chapter 2. Studies by (Olsen, Kasper, and Wit 2022), (Grantz 2001), (Łotysz 2010), (Di et al. 2016), (G. Wei, Qiu, and X. J. Wei 2012), and (Gavin et al. 2019) have improved our understanding of the long-term settlement of existing immersed tunnels in soft soil deposits. The main takeaways from this research are covered in Section 2.4.2 of Chapter 2. Examples of elastic analysis of soil-foundation interaction are provided by (Avramidis and Morfidis 2006), (Worku 2013), (Selvadurai 1979), (Morfidis 2007) and (Hamza 2016) and are discussed in further detail in Section 4.4 in Chapter 4. Primary and secondary consolidation of multilayered soil under multistage ramp loading is covered by Feng et al. (2017), and Conte and Troncone (2006) and provides a perfect framework for a model to define secondary consolidation of overconsolidated clay layers and is described in Chapter 4.

With knowledge of current long-term settlement from relevant cases combined with understanding of the design considerations on IMT foundations, failure mechanisms, soil variability, and soil consolidation

models, the knowledge gap is addressed. This general approach is necessary because long-term deformations could cause severe damage to the tunnel structure. Significant long-term deformations and differential settlements could lead to cracking of the concrete, resulting in tunnel leakage. The research mentioned above serves as a basis for quantifying the effect of variations in soil layer thickness on the secondary consolidation of the soil column under the tunnel element. Currently, this understanding is lacking. All these research topics are summarized in figure 1.2, leading to the research gap and the main subject of this thesis.

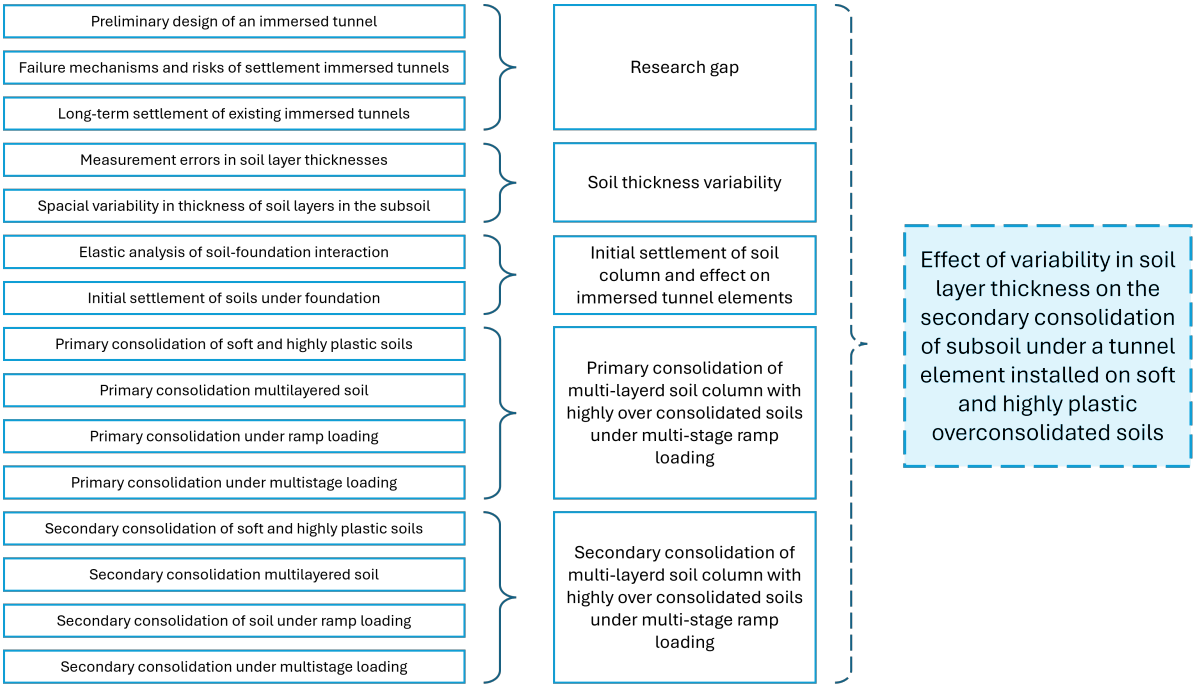


Figure 1.2: Research topics thesis

1.3. Goal and aim

This research aims to investigate the impact of variations in soil thickness and depth within a defined soil column on secondary consolidation of soil beneath immersed tunnel elements in the longitudinal direction. The study will focus on long-term deformations that develop under normal environmental conditions, excluding factors such as earthquake activity, siltation during and after tunnel installation, and tunnel mechanical failures. A simplified model of the tunnel and soil layers will serve as the framework for this study. Within this framework, simulations will be conducted that incorporate variations in soil thickness to define and examine the effects on secondary consolidation.

The focus of this research will be on the Fehmarnbelt Tunnel, which serves as the primary example case. The final product of this study will provide a deeper understanding of the impact of variations in soil layer thickness and depth on secondary consolidation. In addition, a model capable of capturing these effects in terms of the total long-term deformation of all elements in the longitudinal direction will be developed.

1.4. Research questions

The objective of this research provides a framework for the study to be carried out. To ensure a structured approach and facilitate the research process, this objective is divided into several subobjectives. The principal research question guiding this study is: "What is the effect of soil thickness variability on the secondary consolidation along the alignment of the Fehmarnbelt Immersed Tunnel Project, and what method can be employed to characterize and quantify this?"

This overarching research question is further subdivided into specific questions addressing particular

phenomena and effects as follows:

- What is the effect of the soil-structure interaction and how does this translate to the initial deformation under the immersed tunnel elements?
- What is the effect and total magnitude of the variation in soil thickness on the primary consolidation of highly overconsolidated multilayered soft soils under multistage ramp loading?
- What is the effect and total magnitude of the variation in soil thickness on secondary consolidation of highly overconsolidated multilayered soft soils under multistage ramp loading?
- What will happen to the range of variation for the total deformation of the soil beneath the tunnel due to the change in standard deviation of the soil layer thickness?

1.5. Research structure

This thesis started with an introduction to the project and an overview of the topics and structure present in this thesis. The other chapters are as follows.

1. Literature study on soil deformation processes
2. Specifics of the Fehmarnbelt tunnel
3. Methodology model
4. Results of the analytical models for one simulation
5. Variability results for the entire tunnel configuration
6. Conclusion and recommendations
7. List of equations

The second chapter provides a comprehensive review of the literature relevant to the task at hand. Covering an extensive explanation of various parameters used in soil analysis and offers a detailed account of the processes involved in soil deformation and consolidation.

The third chapter provides an overview of the specifics of the Fehmarnbelt Tunnel. It dives into the construction process, tunnel dimensions, loading phases, geological conditions, and the zones in which the project will be subdivided to make the analysis of the ground response easier comprehensible.

The fourth chapter outlines the methodology used to analyze the impact of soil thickness variability on secondary consolidation of soil beneath immersed tunnel elements. The models and techniques used in this study are crucial for understanding the complex interactions between soil layers and tunnel structures. Each section provides a detailed explanation of the models, their technical aspects, advantages, and limitations, ensuring a comprehensive approach to soil deformation analysis.

Chapter five presents the results obtained from analytical models applied to a single simulation of soil layer thicknesses. The findings provide insight into the initial deformation, primary consolidation, and secondary consolidation of the soil beneath the elements of the Fehmarnbelt Tunnel. Each section details the results for specific zones, highlighting the impact of soil composition and thickness variability on the overall deformation process.

Chapter six presents the results of the variability analysis conducted using 500 different simulations of soil layer thicknesses. The findings highlight the impact of soil thickness variability on initial deformation, primary consolidation, and secondary consolidation of the soil beneath the Fehmarnbelt Tunnel elements. Each section provides a detailed analysis of the variability in deformation in different zones, emphasizing the importance of considering soil variability in geotechnical design and construction.

The last chapter summarizes the key findings of the study, highlighting the impact of soil thickness variability on the secondary consolidation of the soil beneath the elements of the Fehmarnbelt Tunnel. It provides a comprehensive overview of the conclusions drawn from the analysis and offers recommendations for future research and practical applications. The insights gained from this study contribute to a better understanding of the behavior of the soil under immersed tunnels and inform more accurate predictions and designs in future projects.

2

Literature study on soil deformation processes

Understanding the mechanisms of soil deformation is crucial for the design and construction of immersed tunnels. This chapter provides a comprehensive review of the existing literature on soil deformation processes, focusing on the geotechnical aspects of soil loading and unloading, the behavior of different soil types under various conditions, and the consolidation phases that significantly impact the stability and longevity of tunnel structures.

2.1. Geotechnical aspects of soil loading and unloading

This section delves into the fundamental properties of soil, including its composition and the behavior of different types of soil under mechanical loading and unloading and explores the characteristics of granular and cohesive soils, highlighting their distinct responses to external forces. The latter part includes the primary and secondary consolidation phases, which are critical to understanding the long-term performance of the soil beneath immersed tunnel elements.

2.1.1. The definition of soil

Soil, as a natural material, consists of solid particles, water, and air. The proportion of these components varies, influencing the soil's physical properties. This subsection defines soil and introduces key concepts such as void ratio and porosity, which are essential for analyzing soil behavior under load.

As stated by (Testbook 2024): *"Soils, as they exist in nature, consist of solid particles (mineral grains, rock fragments) with water and air in the voids between the particles."* This means that the entire volume of most soils is made up of solids (now referred to as the soil skeleton), water, and air. A representative volume of a soil is shown in figure 2.1. These phases of the soil within the soil can interchange their respective volumes subjected to external factors such as freezing, loading, or drying of the soil.

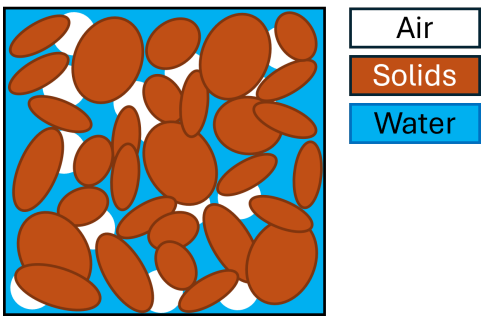


Figure 2.1: Representative volume of soil

The relationship between the volume voids (air and water) and the volume of the soil skeleton is called the void ratio. This soil property is used in further research in this thesis. It is calculated using equation 2.1 and related to porosity (ratio of the volume of voids over the **total volume** of the soil) of the soil as shown in equation 2.2.

$$e = V_v / V_s \tag{2.1}$$

$$n = \frac{e}{(1+e)} \tag{2.2}$$

where

e: Void ratio [-]
V_s: Volume of solids [m^3]
V_v: Volume of pores [m^3]
n: Porosity [-]

The initial void ratio (e_0) is a more specific ratio that describes the state of the soil before any loading or consolidation has occurred. This ratio is used in the calculation of several consolidation parameters, which are described in further detail in Section 2.1.6 and Section 2.1.7.

Soils can be subdivided into categories according to the size of their solid particles. Ranging from small to large soil particles: clays, silts, sands, gravels, and all combinations of these soil types. Clays contain a lot of smaller soil particles, giving them some special characteristics. These characteristics are derived from (Ural and Zoveidavianpoor 2018).

- Clays exhibit plasticity, which means that they can be reformed in shape when wet.
- Clays can hold up a lot of water due to the smaller particles and thus larger soil surface area.
- Clays have a low permeability, so water moves slowly within the cohesive soil.
- Clays are highly compressible, which means they can undergo significant volume changes when subjected to loads
- Clays can exhibit significant shrinkage or swelling.

These qualities of clay can be the foundation for many engineering problems, which will be discussed further in the following chapters. Especially when these clays are loaded or unloaded, they cause primary and secondary consolidation of a soil column, as explained in Section 2.1.2.

2.1.2. Loading and unloading of soil

Different types of soil exhibit unique responses to loading and unloading. Granular soils, characterized by low compressibility and high permeability, react differently compared to cohesive soils, which are highly compressible and exhibit significant secondary consolidation. This subsection explains these differences and their implications for soil deformation. For the sake of explanation, this section will discuss granular soils (sands) and cohesive soils (clays) and their reaction to loading.

Granular soils have low compressibility, which means that they undergo small volume changes under loading. Furthermore, there is much friction between the particles, which leads to a high shear strength. The permeability of these soils is very high because these soils have large pore openings and water can easily flow through them (Bell 1992). The opposite holds for cohesive soils, and they have low permeability. This is due to the fact that cohesive soils have flattened particles and a larger surface area.

In figure 2.2, the loading sequence of the granular soil is shown. The 'q' mentioned in the figure represents the uniform distributed loading or unloading to which the soil column is subjected. In the initial settlement phase, some of the air and water escapes the soil skeleton and the soil exhibits **shrinking** (Das 2016).

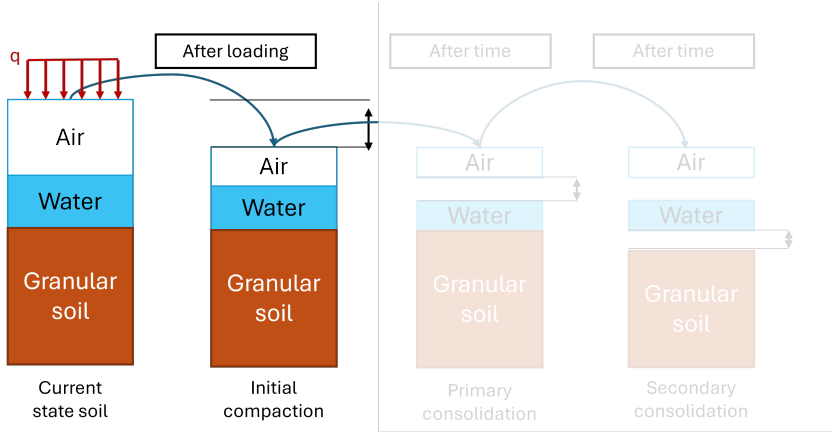


Figure 2.2: Loading sequence of granular soil

The consolidation phase does not apply to granular soils. Due to their high permeability, pore pressures will not build up in the soil, because water can flow out easily (Das 2016).

Bell (1992) states that the secondary consolidation (creep) of granular soils is negligibly small compared to the secondary consolidation of cohesive soils. This is the case because of the larger particle size in the granular soils and the lack of electrochemical interactions of these particles. These characteristics prevent large particle rearrangements or gradual adjustment of soil structure over time.

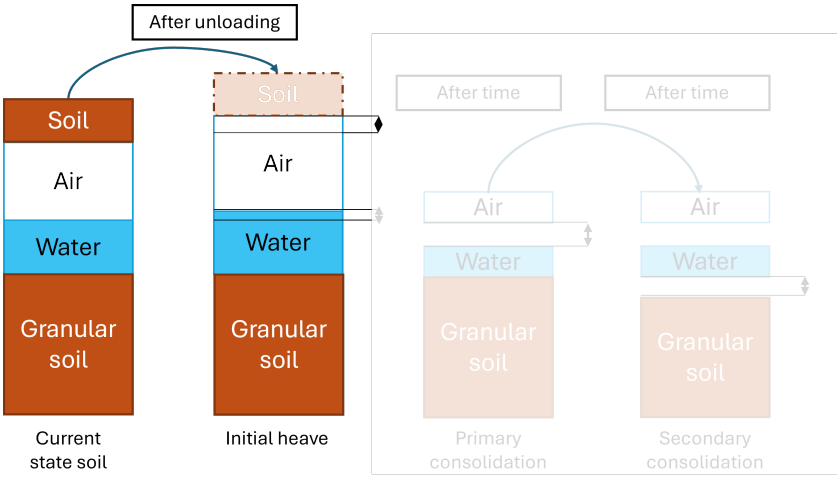


Figure 2.3: Unloading sequence of granular soil

The unloading phase is similar to the loading phase; the difference is that in the loading phase, water and air are pushed out of the soil skeleton, and during unloading there is room for air and water to flow back in. This is visualized in figure 2.3. Again, no consolidation phases are relevant or present.

Cohesive soils have high compressibility, large interparticle bonding, low permeability, and lower shear strength than granular soils. The loading of cohesive soil is shown in figure 2.4. The high compressibility will lead to a larger volume change when subjected to loading of the soil structure. The low permeability will cause the pore pressures to build up, so over time this water wants to escape the soil structure. This causes the soil to shrink even after the loading is applied and up to the point where the excess pore pressures will be 0 again. This phase is called the primary consolidation of the soil and will occur over time. The third phase is called secondary consolidation. In this phase, the soil deforms over time under constant stress. This deformation is caused by the rearrangement of soil particles and the gradual adjustment of the soil skeleton (Bell 1992). These phases will be discussed in further detail in Section 2.1.6 and Section 2.1.7.

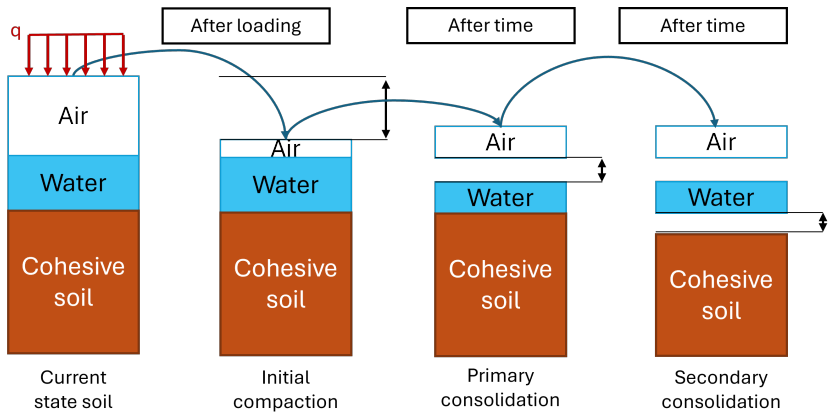


Figure 2.4: Loading sequence of granular soil

The unloading of the soil follows the same principle as with a granular soil. Primary and secondary consolidation does not take place here. For primary consolidation, this is the case, because during unloading, the applied load on the soil is reduced. Primary consolidation is driven by an increase in load, which causes an increase in pore water pressure. When the load is reduced, there is no additional pore water pressure to dissipate (Nova and Hueckel 1981).

Creep is driven by a constant level of applied stress. During unloading, the stress on the top of the soil is reduced, removing the driving force of creep. Without sustained stress, the mechanisms that cause creep, such as particle rearrangement and viscous flow, are no longer active. Furthermore, when the load is removed, the soil tends to recover elastically. This means that the deformation that occurred under load is partially reversed immediately, rather than continuing to deform over time (Shen, Z. Zhou, and Ma 2023). The total sequence is shown in figure 2.5.

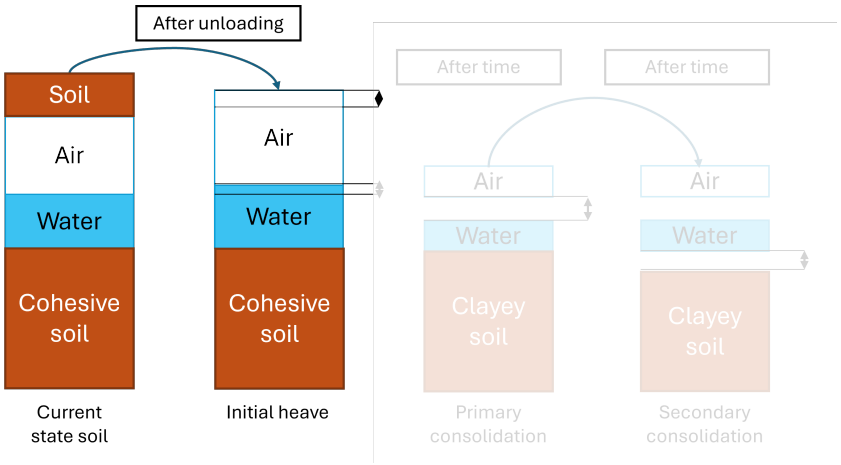


Figure 2.5: Unloading sequence of cohesive soil

2.1.3. Glacial soils

The Fehmarnbelt area features glacial soils (more information on the soils is available in Chapter 3), which have undergone significant compaction due to historical ice pressure. These soils exhibit unique properties, such as high overconsolidation ratios and variable permeability, which must be considered in geotechnical analysis.

The Fehmarnbelt area was stacked with ice up to 11,700 years ago when the Last Ice Age ended. This led to the creation of glacial soil layers in this area. These glacial soils have special characteristics that must be taken into account. Glacial soils consist of highly overconsolidated and highly heterogeneous clays and many soils originating from the deposition of meltwater sediments. The meltwater sediment

or glacier sediment soils do not pose a great risk in terms of primary or secondary consolidation, except if they are made up of a lot of cohesive material.

Glacial soils that are made up of cohesive material will pose difficulties in terms of primary and secondary consolidation. The special characteristics to be taken into account are from (Savage, Morrissey, and Baum 2000) and as follows:

- Glacial soils are highly overconsolidated, because they experienced high stresses in the past. This leads to a high OCR value. (explained in section 2.1.4)
- Glacial soils are often very compacted, because the large volume of ice that was present on top of them. This compaction leads to a high soil density.
- Glacial soils have a large variation in permeability. Glacial meltwater deposits often contain coarser material, leading to higher permeability, and glacial tills (cohesive soil) for example have a very low permeability.
- These soils are typically less compressible compared to other soil types due to their dense nature.
- Their dense state also contributes to high soil stiffness.

In addition, the heterogeneous nature of glacial soils can pose challenges in geotechnical engineering. Variability in composition and properties requires careful site investigation and soil testing to ensure stability and performance (Savage, Morrissey, and Baum 2000).

2.1.4. Overconsolidation of soils

This subsection explains the concept of the overconsolidation ratio (OCR) and its importance in predicting soil behavior under load.

"Overconsolidation is the condition under which a soil is in when it experiences a current stress lower than a stress it has experienced in the past" (Das 2016). **Stress** is the external force applied on the soil distributed over the cross-sectional area and is defined as shown in equation 2.3. **Strain** is the deformation of a soil due to an applied force and represents a relative change in shape or size. Mathematically, strain is defined as shown in equation 2.4 (Das 2016).

$$\sigma = \frac{F_{ex}}{A} \tag{2.3}$$

$$\varepsilon = \frac{\Delta L}{L_0} \tag{2.4}$$

where

σ: The external force applied on the soil distributed over the cross-sectional area $[kPa]$
F_{ex}: The applied external force $[kN]$
A: The cross-sectional area over which the force is distributed $[m^2]$
ε: Strain $[-]$
ΔL : Change in length $[m]$
L_0: Original length $[m]$

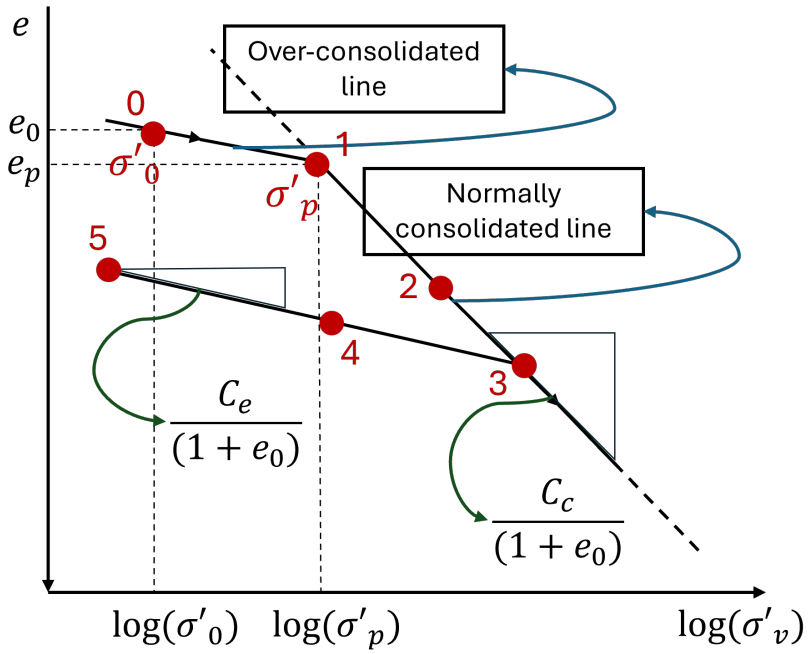


Figure 2.6: Graph with void ratio versus stress and the stress paths of a cohesive soil under multi-stage loading

Figure 2.6 shows a graph that depicts the principle of overconsolidation. Point 0 in the figure resembles the initial stress state of the soil. After enough stress is experienced, it will reach point 1, which resembles the preconsolidation pressure, from this point the plastic deformation will occur. The plastic deformation will be explained in more detail in Section 2.1.5. So in every stress state on the line from point 0 to point 1 the soil will be in an overconsolidated state. The ratio of overconsolidated stress to current stress state is called the **overconsolidation ratio** and is an important soil parameter used in this research. It is calculated as shown in equation 2.5.

$$OCR = \frac{\sigma_p^{'}}{\sigma_0^{'}} \tag{2.5}$$

where

OCR:	Overconsolidation ratio of the soil $[-]$
σ'_p:	Preconsolidation pressure $[kPa]$
σ'_0:	Initial stress state $[kPa]$

2.1.5. Plastic behavior of cohesive soils

Cohesive soils exhibit plastic deformation beyond their elastic limit. Understanding this behavior is crucial for predicting long-term soil performance. This subsection discusses the stress-strain relationship and the role of oedometer stiffness in soil analysis.

In figure 2.7 the stress versus strain for a cohesive soil is shown. The soil behaves itself elastic from point 0 to 1, and only after crossing the preconsolidation pressure, will it plastically deform. Plastic deformation refers to the permanent change in the shape of soil when subjected to stress beyond its elastic limit. Unlike elastic deformation, which is reversible, plastic deformation remains even after the stress is removed. After plastic deformation has occurred and the load has been removed, the soil now has a new preconsolidation pressure (point 3) and has hardened, because it needs a higher pressure than before to reach the elastic boundary again (Das 2016).

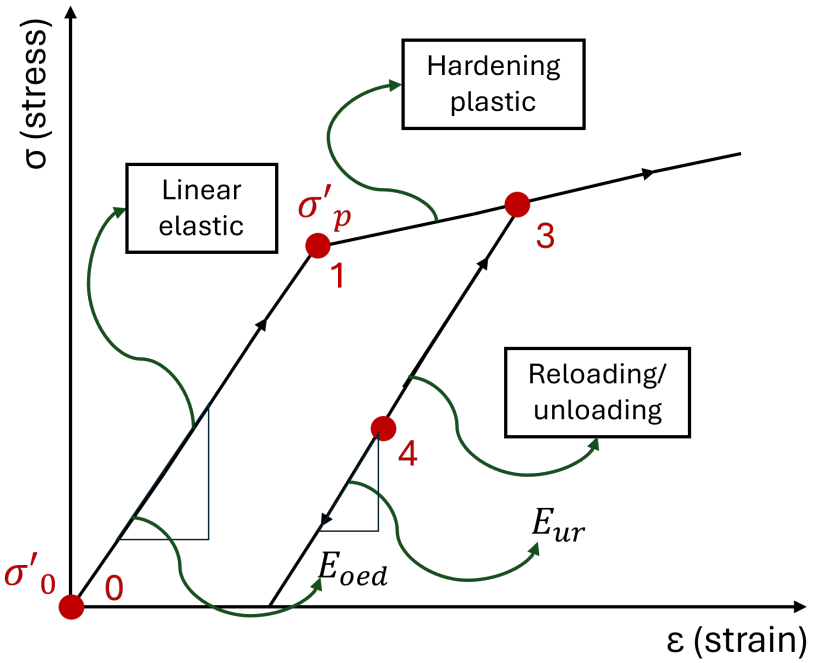


Figure 2.7: Stress versus strain for a cohesive soil under multi-stage loading

The oedometer stiffness (mentioned in Figure 2.7) of a soil is determined in the laboratory and defines the ratio of the applied vertical stress to the resulting vertical strain in a soil sample during an oedometer test. The oedometer stiffness reflects the resistance of the soil to deformation under one-dimensional loading conditions. It is defined as shown in equation 2.6 and will be used in further detail in this report.

$$E_{oed} = \frac{\Delta\sigma}{\Delta\epsilon} \tag{2.6}$$

where

E_{oed}: Oedometer stiffness $[kPa]$
$\Delta\sigma$: Change in stress $[kPa]$
$\Delta\epsilon$: Change in strain $[-]$

2.1.6. Primary consolidation of soil

Primary consolidation involves the dissipation of excess pore water pressure, leading to a decrease in soil volume. This subsection introduces Terzaghi's one-dimensional consolidation equation and explains the factors that influence primary consolidation.

Primary consolidation is the process by which a soil undergoes a change in volume over time when a load was applied to the soil (sample). This load is carried by the pore water, creating excess pore water pressure, which drives the dissipation of the pore water and leads to a decrease in soil volume. This progress is described and analyzed by (Terzaghi, Peck, and Mesri 1996) and mathematically described with *Terzaghi's one-dimensional consolidation equation* in equation 2.7. As mentioned, this process is considered one-dimensional, meaning that deformation occurs only in the vertical direction, while lateral deformation is negligible. This assumption is justified by considering the soil to be laterally infinite, which implies negligible pore water dissipation in the horizontal directions.

$$\frac{\delta u}{\delta t} = C_v * \frac{\delta^2 u}{\delta z^2} \tag{2.7}$$

where

u: Excess pore water pressure $[kPa]$
t: Time $[s]$
C_v: Coefficient of consolidation $[m^2/s]$
z: Depth $[m]$

The C_v in this equation measures the speed with which the consolidation in the soil occurs and is defined as shown in Equation 2.8 from (Terzaghi, Peck, and Mesri 1996). This parameter is also an important parameter in the further research in this thesis.

$$C_v = \frac{k_h}{\gamma_w * m_v} \tag{2.8}$$

where

C_v: Coefficient of consolidation $[m^2/s]$
k_h: Hydraulic conductivity $[m/s]$
γ_w: Unit weight of water $[kN/m^3]$
m_v: Coefficient of volume compressibility $[1/kPa]$

The hydraulic conductivity (k) in this equation represents the rate at which water can move through the soil. In a three-dimensional soil volume, hydraulic conductivity varies in different directions, with distinct values for each axis. For the purposes of this thesis, the focus will be on vertical hydraulic conductivity, which is the key parameter under consideration. Whenever hydraulic conductivity is mentioned, it refers specifically to the vertical direction. The m_v *(Coefficient of volume compressibility)* is calculated by $1/E_{oed}$ and quantifies how much a volume of soil changes under applied load.

The primary consolidation phase continues until the point where all excess pore pressure is dissipated. Terzaghi (1996) also came up with a parameter to define the degree of consolidation, which is shown in Equation 2.9.

$$U = \frac{2}{\sqrt{\pi}} \sqrt{\frac{C_v t}{H_d^2}} \tag{2.9}$$

where H_d is the length of the drainage path, the shortest length a water particle must travel to exit the soil. C_v is the consolidation coefficient, t is the time and U is the degree of consolidation. When the degree of consolidation reaches 100 % the primary consolidation is finished and the secondary consolidation phase (Section 2.1.7) will start from that stress and strain level (Terzaghi, Peck, and Mesri 1996).

2.1.7. Secondary consolidation of soil

Secondary consolidation, or creep deformation, occurs over time under sustained load. This subsection explores the mechanisms driving secondary consolidation and its impact on soil deformation.

The secondary consolidation explained here at the beginning is based on Hypothesis A (R. Olson 1989) and is used to better understand the principle. In Section 2.1.9 both hypotheses will be explained in further detail.

Secondary consolidation is driven by rearrangement and adjustment of soil particles under sustained load (Das 2016). The rate at which this occurs decreases with time but can continue for a long period of time, especially in cohesive soils (Terzaghi, Peck, and Mesri 1996). The important takeaway here is that secondary consolidation depends only on time, not stress or strain. However, it starts at a certain level of stress/strain that will differ for different soil parameters due to a variation in the primary consolidation and thus the end point of that consolidation phase. A broad overview of all three

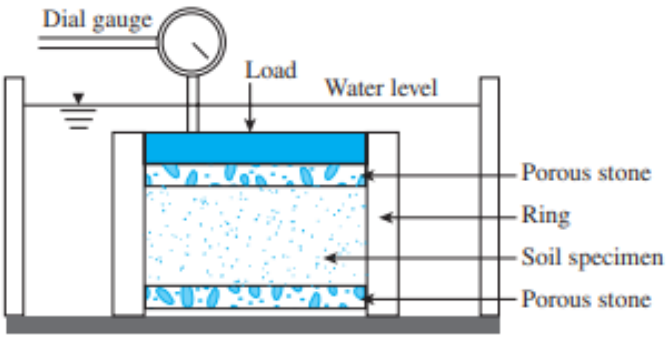


Figure 2.8: Test setup for consolidation test (Das 2016)

deformation processes in cohesive soil is given in Section 2.1.9. Again, cohesive soils exhibit significant secondary consolidation in contrast to granular soils that show minimal effects (Kaczmarek and Dobak 2024). Secondary consolidation is expressed as shown in Equation 2.10 from (Terzaghi, Peck, and Mesri 1996).

$$S_s = C_{\alpha e} * H * log\left(\frac{t_2}{t_1}\right) \tag{2.10}$$

where

S_s: Secondary consolidation $[m]$	
$C_{\alpha e}$:	Coefficient of secondary compression $[-]$
H:	Thickness of the compressible soil layer $[m]$
t_1,t_2:	Times at the begin and end of secondary consolidation period $[s]$

The parameter $C_{\alpha e}$ (the secondary compression coefficient) is determined in a laboratory with, for example, an oedometer test and quantifies the rate of secondary consolidation (Das 2016).

2.1.8. Determination of consolidation indices

Consolidation indices, such as the compression index and the swelling index, are determined by laboratory tests. This subsection explains the methods used to obtain these indices and their relevance in soil analysis.

The swelling index (C_e), the compression index (C_c), and the secondary compression index ($C_{\alpha e}$) are determined from laboratory data, as stated above. This laboratory data is the result of an *oedometer test*. The oedometer test setup is shown in Figure 2.8.

The results of the oedometer test are shown in Figures 2.9 and 2.10. In Figure 2.9, the change in the void ratio is plotted against the log of the corresponding vertical stress applied to the soil sample. The slopes of the lines correspond to the compression index and the swelling index, the empirical relation for these indices is depicted in equation 2.11 and 2.12. In figure 2.10, the change in void ratio is plotted against the logarithm of time. Here, the slope of the line corresponds to the secondary compression index which is defined as shown in Equation 2.13 (Das 2016).

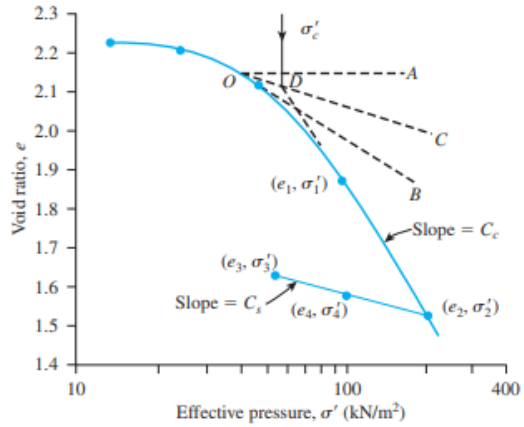


Figure 2.9: Void ratio versus effective pressure curve for a soft clay (Das 2016)

$$C_c = \frac{e_1 - e_2}{log\sigma_2^{'} - log\sigma_1^{'}} = \frac{e_1 - e_2}{log\left(\frac{\sigma_2^{'}}{\sigma_1^{'}}\right)} \quad \text{(Das 2016)} \tag{2.11}$$

$$C_e = \frac{e_3 - e_4}{log\left(\frac{\sigma_4^{'}}{\sigma_3^{'}}\right)} \quad \text{(Das 2016)} \tag{2.12}$$

$$C_{\alpha e} = \frac{\Delta e}{logt_2 - logt_1} = \frac{\Delta e}{log\frac{t_2}{t_1}} \quad \text{(Das 2016)} \tag{2.13}$$

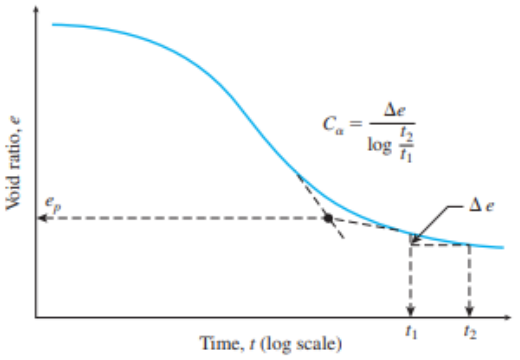


Figure 2.10: Variation of e with logarithmic time under given incremental load (Das 2016)

where

C_e:	Swelling index [−]
C_c:	Compression index [−]
e_1, e_2, e_3, e_4:	Void ratios [−]
t_1:	Time for completion of primary consolidation settlement $[years]$
t_2:	Time at which the secondary consolidation needs to be known $[years]$
Δe	Change in void ratio at the end of primary consolidation [−]
σ'	Effective pressure $[kN/m^2]$

2.1.9. The complete consolidation process

The complete consolidation process includes initial deformation, primary consolidation, and secondary consolidation. This subsection provides an overview of these phases and their cumulative effect on soil deformation.

The complete consolidation process is visualized on a time versus settlement graph in Figure 2.11. This figure shows that the *final total deformation* is an addition of the initial deformation, the primary consolidation, and the secondary consolidation. Each of these parts have their own analytic calculation model, which will be described in more detail in Chapter 4.

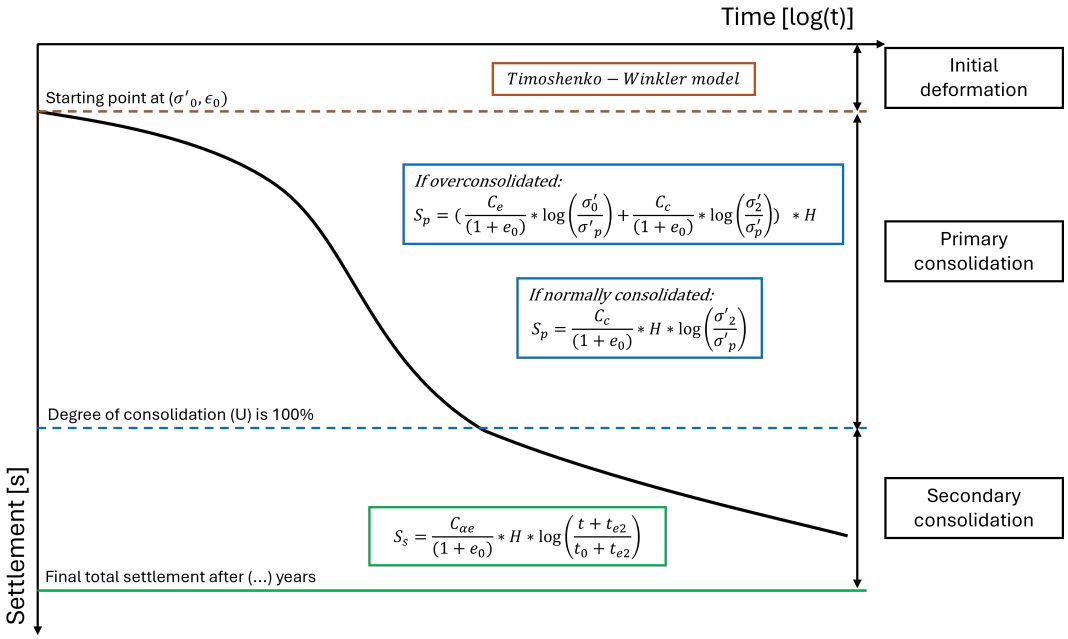


Figure 2.11: Complete consolidation process for cohesive soil

The factors C_c, C_e, $C_{\alpha e}$ are the dimensionless factors described in section 2.1.7 and used to quantify the rate of deformation/consolidation. There exist two hypotheses about the start of secondary consolidation; hypothesis A and hypothesis B.

Hypothesis A states: *The secondary consolidation, or creep compression strain due to viscous effects, begins only after the end of primary consolidation (EOP).* Research that gives a theory and explanation behind this is from: (R. Olson 1989), (Martins 1985), and (Joseph 2014).

Hypothesis B states: *The secondary consolidation (creep compression) begins during the primary consolidation phase and continues thereafter.* Sources supporting this hypothesis are: (Takeda et al. 2013), (W.-Q. Feng and J.-H. Yin 2017) and (Hawlader, Muhunthan, and Imai 2003).

Degago (2011) points out that many laboratory and field studies have been carried out on both hypotheses and their results are very mixed. Some support hypothesis A and others support hypothesis B. This ongoing debate has kept the topic contentious among researchers, highlighting the need for further research to reach a consensus. The research carried out in (Degago et al. 2011) involved the isotache method (which will be described in more detail in Chapter 4) to determine the time-dependent compressibility of clays. The conclusion of the research pointed out that the secondary consolidation of clays agrees well with hypothesis B. So, this hypothesis will be used in this thesis.

2.2. Multi-stage loading of multi-layered cohesive soil columns under foundation

This section examines the impact of multistage loading on cohesive soil columns, including the challenges posed by varying soil parameters and the interaction between soil layers. It introduces the Timoshenko beam Kerr foundation model and discusses its application in geotechnical analysis.

2.2.1. Loading of multilayered soil columns

Multilayered soil columns exhibit complex behavior under load due to variations in soil parameters. This subsection explains the methods used to calculate the equivalent soil stiffness and the implications for soil deformation.

A multilayered soil column brings new challenges to the table. The soil itself will have varying soil parameters throughout its depth. In addition, excess pore pressures will build up over a greater height in

the soil, because some soil layers will be trapped in between cohesive layers that have a low permeability.

Initial compaction will be modeled with the Timoshenko beam Kerr foundation model (from now on: TBKF model). More information about this model can be found in Chapter 4. The strength in this model is defined by three parameter: G , k and c and these depend on ν and E_{oed} (explained in sections 2.1.1 and 2.1.5 respectively). These parameters are used to determine the amount of deformation that will occur initially. The strength of the soil will vary throughout the depth of the column due to the different layers of soil present. To counteract this problem, it is chosen to use a mean strength value for the entire column as proposed in (Pantelidis 2019).

Pantelidis (2019) researched the maximum relative error of many methods that could be used to determine the equivalent soil stiffness of a multilayered soil column. The result of this research is shown in figure 2.12. The unsafe side mentioned here refers to the overestimation of the soil stiffness/strength parameters. Many of these methods are unsatisfactory for this research, because they are based on thin soil layers (Odemark's method(1949), Barros' method (1966), Sridharan et al.'s method (1990), Hirai and Kamei's method (2004) , Hirai's method (2008) and Abu-Farsakh and Chen's method (2012)) decreasing strength with increasing depth (Odemark's method(1949)), or on a regular interval (Barden's method (1962)) in soil layering. However, the soil layers present in the Fehmarnbelt area increase in strength with increasing depth and are very irregular in interval and thickness. Pantelidis (2019) states that the most commonly used method in the academic world is the method of Egorov & Nichiporovich in (1961) suggested by Bowles (1996).

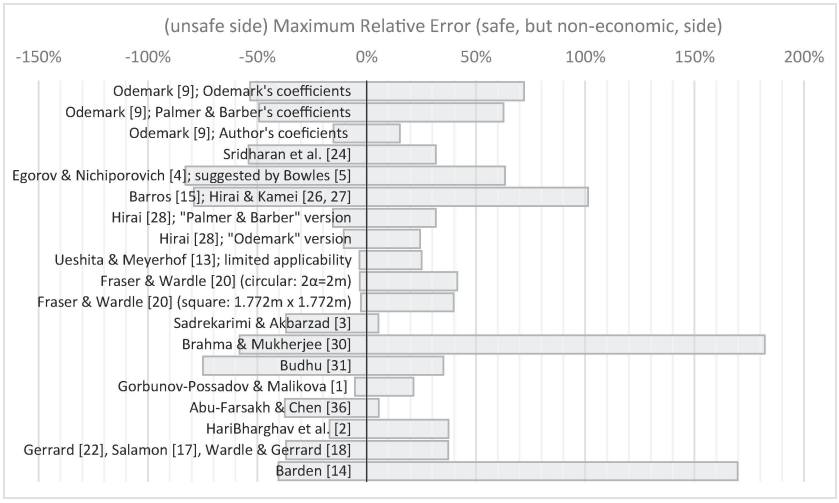


Figure 2.12: Results comparison methods equivalent soil stiffness (Pantelidis 2019). The minus sign indicates the unsafe side and not minimum error in this figure

Figure 2.12 clearly shows that the Ueshita & Meyerhof (1967) and Fraser & Wardle (1976) methods are the relatively safest methods to use, but are not clearly defined ((Ueshita and Meyerhof 1967)) or are a whole study on their own. So for the sake of this thesis the more easy and more commonly used method of Egorov & Nichiporovich (1961) is used. This choice is based on the fact that the initial settlement will be a smaller part of the total settlement. The proof for this statement will be shown in the results (Chapter 5) of the models. The takeaway from the research of Pantelidis (2019) is that the equivalent strength parameter could be overestimated due to the method chosen here. The final result of this research will be a quantitative determination of the effect of soil thickness variability on secondary consolidation, and overshooting the stiffness parameter will not endanger the conclusion of this research, so it is safe to use the method suggested by Egorov & Nichiporovich (1961).

The method chosen gives equation 2.14 as a way to calculate the equivalent soil strength parameters. Poisson's ratio in equation 2.15 is a parameter that describes the relationship between longitudinal and lateral strain of a material and is used to convert the strength modulus to a stiffness modulus as elaborated in Chapter 4. It is a parameter that is determined in the laboratory.

$$E_{eq} = \frac{\sum_{i-1}^{n} h_i E_{oed,i}}{\sum_{i-1}^{n} h_i} \tag{2.14}$$

$$\nu_s = \frac{\sum_{i-1}^{n} h_i \nu_{s,i}}{\sum_{i-1}^{n} h_i} \tag{2.15}$$

where

E_{eq}: Equivalent Oedometer stiffness $[kPa]$
E_{oed}: Oedometer stiffness $[kPa]$
ν_s: Poisson's ratio $[-]$
h_i: Respective height of soil layer $[m]$

2.2.2. Multistage loading of soil

Multistage loading involves applying loads incrementally, allowing better control and monitoring of soil settlement. This subsection discusses the effects of multistage loading on primary and secondary consolidation.

Multistage loading is often used in geotechnical engineering to gradually increase the load on the soil, allowing better control and monitoring of settlement (Z.-J. Chen, Wei-Qiang Feng, and Jian-Hua Yin 2021). For example, if soil layers need to be placed to strengthen the foundation, it is often done in several phases. All deformation phases of the soil column below react differently on multi-stage loading. In figure 2.13 several steps of loading are shown. From point 0 to point 3, three loading steps are applied and from point 3 to point 5, two unloading steps are applied to the soil sample or column (Jian-Hua Yin and Wei-Qiang Feng 2017).

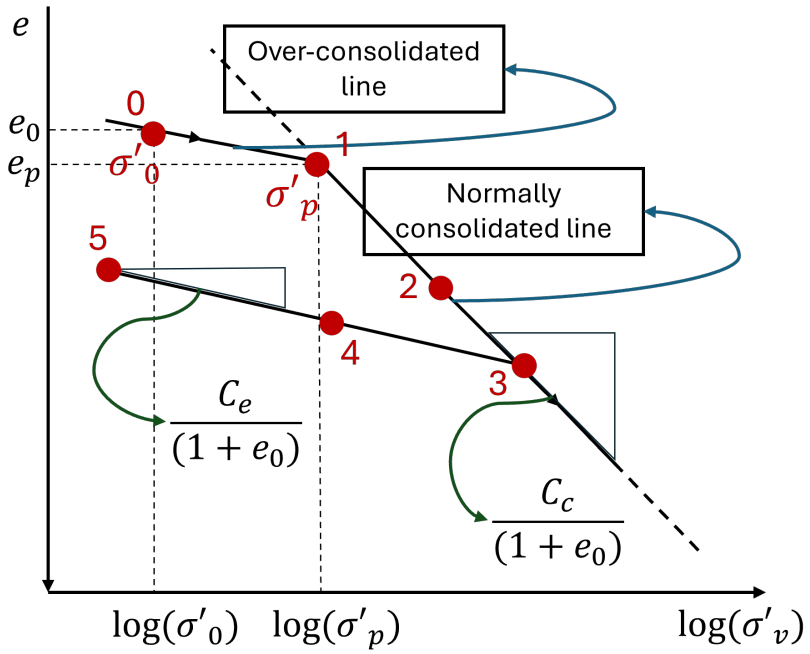


Figure 2.13: Graph with void ratio versus stress and the stress paths of a cohesive soil under multi-stage loading

The initial deformation in multistage soil loading is quite easy to approach. The initial deformations are cumulative over all loading stages. However, multistage loading has a large effect on primary consolidation and, to a lesser extent, on secondary consolidation.

The primary consolidation is affected by multistage loading. Multistage loading is often used to spread the stress more evenly across the soil layers, give the soil time to lose its excess pore pressures, and

make the soil grains gain strength. The following things need to be taken into account for the primary consolidation under multi-stage loading:

- The dissipation of excess pore pressure will occur during a period when no additional load is present, causing the soil to consolidate in between loading periods.
- The overall settlement will be less because consolidation will occur in between the stages of loading occurs.
- The preconsolidation pressure will increase due to the load and its corresponding secondary consolidation (following hypothesis B), so the incremental load will increase the strength of the soil and leads only to elastic deformations in this research, an example is shown in Figure 2.14.

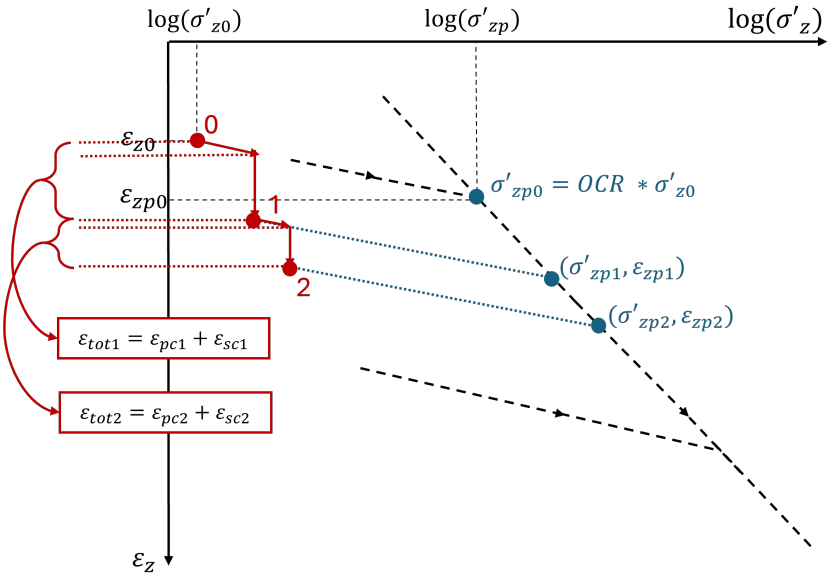


Figure 2.14: Incremental loading path of cohesive soil with primary and secondary consolidation

Figure 2.14 shows the loading path of a cohesive soil and its primary and secondary consolidation. The straight red arrows down indicate secondary consolidation, because no additional load is present and the strain continues. The arrows that follow the blue/black lines indicate the primary consolidation as shown in Figure 2.11 as indicated by the red dots. The total strain is a combination of primary consolidation and secondary consolidation that occurs during the loading phases. The dotted blue lines indicate the stress-strain path a soil would take after a certain loading phase if loaded again. As can be concluded from the figure, the preconsolidation pressure is shown to grow with each loading step (Wei-Qiang Feng et al. 2020). The details of the model in Figure 2.14 will be discussed and explained in further detail in Chapter 4.

The secondary consolidation is affected by multistage loading, but in lesser extent than primary consolidation.

- With each loading step, the soil gains strength (as described with primary consolidation), and this can reduce secondary consolidation.
- The secondary consolidation is governed by time parameters and the End of Primary consolidation (EOP) time; with smaller loading steps, this EOP time parameter will be lower thus starting the secondary consolidation process earlier. More details on this parameter can be found in Chapter 4.
- The cumulative primary consolidation will be less as described in the previous part, which means that the cumulative secondary consolidation will also be smaller, because the starting point of the secondary consolidation is lower (W.-Q. Feng and J.-H. Yin 2017).

These points will be discussed and explained in more detail in Chapter 4 where the model used to determine the quantity is explained.

2.3. Variability in the soil profiles

Understanding the variability in soil profiles is essential for accurate geotechnical analysis. This section covers the measurement errors and spatial variability of soil parameters, highlighting their impact on soil deformation predictions.

2.3.1. Measurement errors in soil layer thicknesses

Measurement errors can significantly affect soil analysis. This subsection discusses the sources of these errors and their implications for geotechnical investigations.

The soil investigation of the soil in the Fehmarnbelt area involved the following techniques (Yumpu.com 2011):

- Seismic surveys
- Boring campaigns
- Geophysical borehole testing
- Advanced laboratory testing
- Large scale testing
- Seismicity

The latter two are not important in the scope of this research, because seismicity is very low according to the soil investigation carried out by Arup (Yumpu.com 2011) and the large-scale testing focuses only on the folded Paleogene clay qualities and does not focus on the parameters needed in this research. Geophysical borehole testing only provided insight into the correlation of the upper layers in the borehole and did not contribute much to the final estimation on layer thicknesses.

The results of the seismic survey, drilling campaigns, and advanced laboratory tests play a significant role in the scope of this research. Advanced laboratory testing provided the soil parameters needed in the models described in this chapter and in Chapter 4. The resulting parameters from the laboratory tests of (Yumpu.com 2011) can be found in Section 3.4.1. The variability of these parameters is outside the scope of this thesis and will not be taken into account.

Measurement errors in seismic surveys and drilling campaigns occur due to errors by the instrumentation itself, due to human errors during the investigation, or by interpolating results from different places. The error could have a large effect on the initial deformation, primary consolidation, and secondary consolidation. The quantity of this effect will be discussed in Chapter 6. Human errors are not predictable and can only be taken into account as a small deviation from the mean value. The interpolation and instrumentation errors play an important role in this thesis. Soil layers can vary from 5 % to 10 % for seismic surveys according to (2005) and from 10 % to 15 % for drilling campaigns according to (1999). This will be the range used for the standard deviation adopted in this thesis for the soil thicknesses of the Fehmarnbelt area. Additional information on soil profile, zones of interest, and thicknesses will be covered in Chapter 3.

2.3.2. Spacial variability in thickness of soil layers in the subsoil

Soil layer thickness varies not only with depth but also longitudinally. This subsection explores the distribution of soil thickness and its impact on deformation predictions.

Soil thickness variations could cause a large difference in the outcome of the predictions on the deformation of the layers present. To capture this range in deformation prediction, a distribution that describes the variation in soil thickness is needed to start the analysis. An example of such a distribution is given in figure 2.15 from (Yan et al. 2021). (Yan et al. 2021), (X. Wang 2021) and (Yamashita et al. 2024) all give indications on the type of distribution to use for the thicknesses of the soil layers.

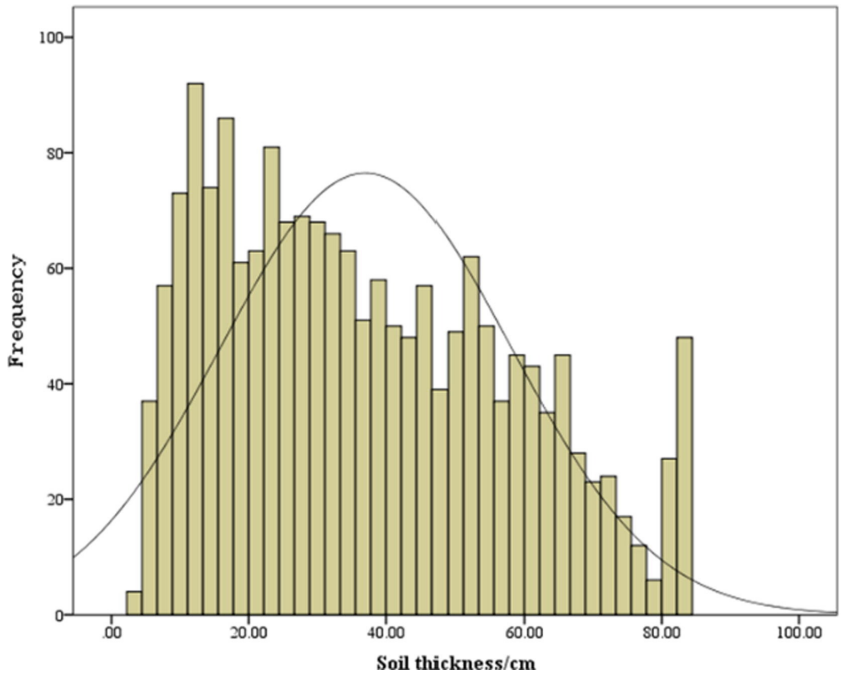


Figure 2.15: Distribution of thicknesses of soil layers by (Yan et al. 2021)

According to (Yamashita et al. 2024) and (X. Wang 2021), the soil thickness distribution is often lognormal. (Yan et al. 2021) further explains that while the soil layer thickness distribution is between lognormal and normal distributions, it deviates significantly from a normal distribution, making the lognormal distribution more appropriate for sampling variations in soil layer thickness.

The total depth of the soil column is restricted between the top of the tunnel and a depth of 100 meters, as shown in Figure 2.16. This 100-meter boundary is a ballpark number chosen based on literature (Hassan 2017), suggesting that the loads acting at the top of the column will influence soil deformation up to twice the width of the tunnel part (for rectangular footings). Given a width of approximately 50 meters, the total depth of influence is set at 100 meters. The tunnel height and the height of the water column are included in this total height, so around 80 meters of soil will be the restricted height of the soil column.

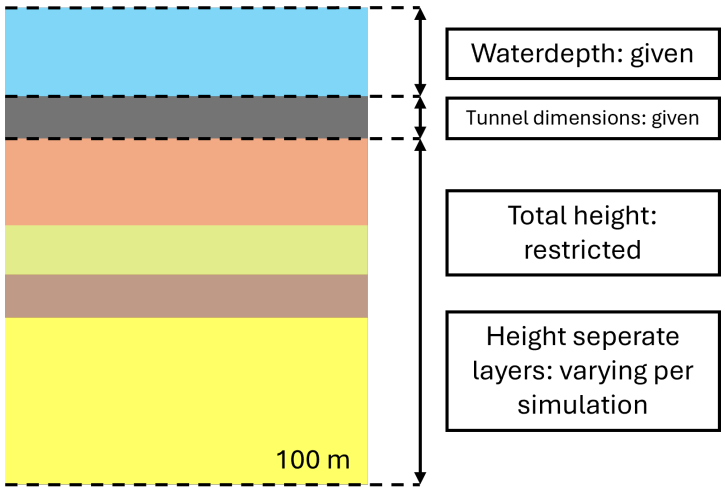


Figure 2.16: Soil column configuration for variation analysis

The random generated layer thicknesses are at least 0.1 meters thick, and their total height is restricted

to the height of the original soil column. This means that the lower layers have an upper boundary of the remaining height that is left in the soil column.

2.4. Risks and failure mechanisms of the tunnel

Immersed tunnels are susceptible to various risks due to soil deformation. This section outlines the failure mechanisms and risks associated with differential settlement, soil-structure interaction, and long-term deformation.

2.4.1. The failure mechanisms and the risks immersed tunnel due to soil deformation

Differential settlement and the interaction between soil and structure can lead to tunnel failure. This subsection discusses these mechanisms and their implications for tunnel stability

Immersed tunnels are susceptible to various failure mechanisms and risks due to soil deformation. Understanding these mechanisms is crucial to ensure the long-term stability and safety of these structures. The five main causes of failure are: differential settlement, differential soil-structure interaction, time-dependent loads (tidal/seismic), groundwater fluctuations, and long-term deformations. Groundwater fluctuations and time-dependent loads, such as tidal or seismic loads, will not be discussed in this thesis, because they are outside the scope of this thesis, as mentioned in Chapter 1. Differential deformations, soil-structure interaction, and long-term deformation will be covered in the following way:

- **Differential deformations**: the differential deformations on the longitudinal axis of the tunnel will be visualized by the difference between special and regular elements that will have different dimensions and thus act differently on the soil beneath.
- **Soil-structure interaction**: the interaction between the soil and structure is governed by the spring stiffness modulus of the soil and will vary over the longitudinal axis due to a variation in the soil type.
- **Long-term deformations**: the end of primary consolidation will vary greatly with the varying soil thicknesses and soil parameters.

All of these mechanisms lead to the same type of failure; cracking of the concrete of the tunnel and leaking at the connected parts of the tunnel. A regularly behaving immersed tunnel is visualized in figure 2.17. The tunnel shown consists of five elements placed behind each other. In the zoomed-in part, the segments (and their connection to each other) of a regular element are shown.

When deformations in certain places in the subsoil become greater than in other locations, the tunnel itself will start to move and deform too, leading to a tunnel visualized in figure 2.18. The connected areas (shear keys) will start to break and the water will start to leak, compromising the performance of the tunnel.

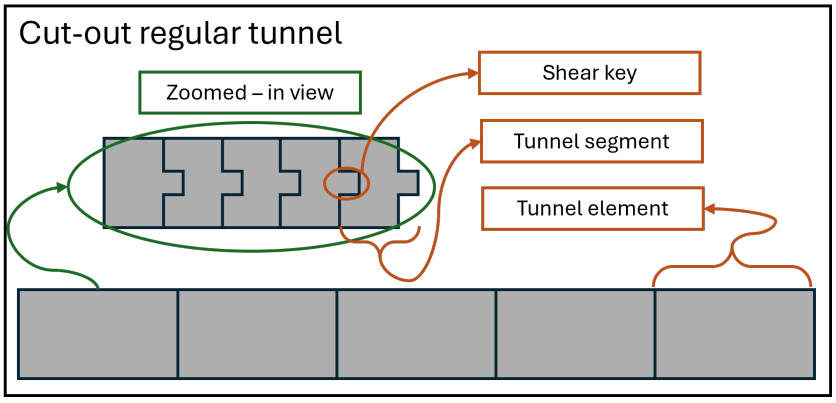


Figure 2.17: Cut-out of a regularly behaving tunnel

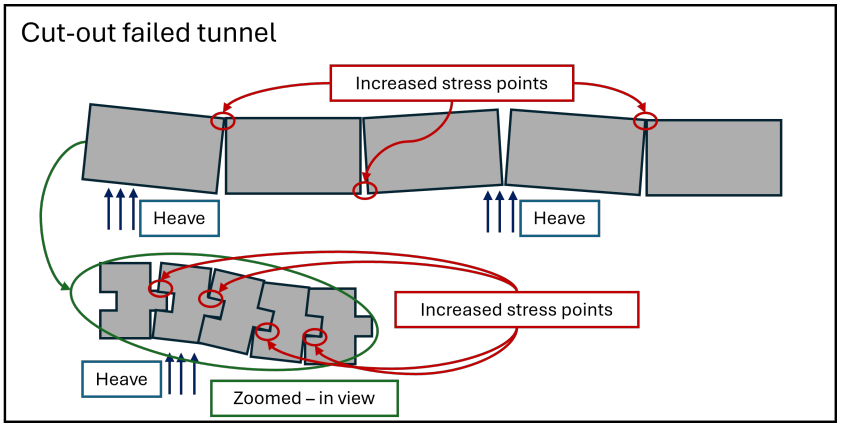


Figure 2.18: Cut-out of a deformed tunnel

2.4.2. Long-term deformation of existing immersed tunnels

Long-term deformation plays an important role (as mentioned in the previous section) in the future performance of existing tunnels. Case studies of existing immersed tunnels provide insights into long-term deformation patterns. This subsection reviews these studies and their relevance to the Fehmarnbelt Tunnel project. Current case studies for these effects are the Yonjiang Immersed Tunnel and the Hong Kong-Zhuhai-Macao Bridge.

The Yonjiang Immersed Tunnel

After 11 years of operation, the Yongjiang Tunnel exhibited a differential settlement of approximately 182 millimeters. The cause of this differential settlement was the uneven distribution of soil layers, the back-silting, and the tidal loads. The results of the differential settlements were cracking of the concrete and leakage in the place where the tunnel segments are joined together. The settlement profile of (H. Zhou et al. 2025) and (Kou et al. 2024) is shown in Figure 2.19 and Figure 2.20 respectively. These figures describe the settlement that occurred over time in the respective tunnels. Figure 2.20 shows the northern part of the Yongjiang tunnel and E1, E2, etc. represent the elements of the tunnel.

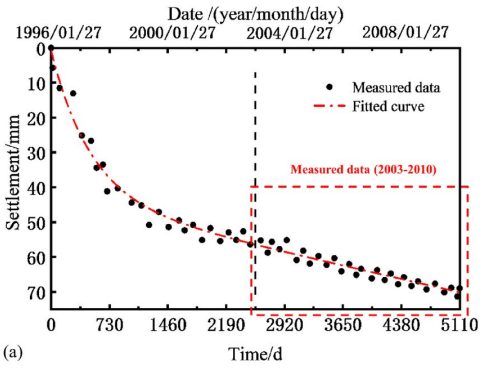


Figure 2.19: Settlement of Yongjiang tunnel from (H. Zhou et al. 2025)

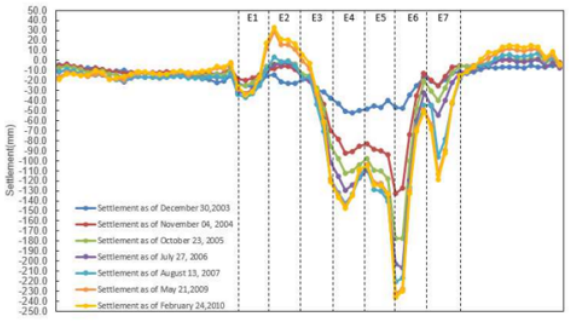


Figure 2.20: Cumulative settlement curve of northern part Yongjiang tunnel (mm) from (Kou et al. 2024)

The Hong Kong-Zhuhai-Macao Bridge

As said with the Yongjiang Immersed Tunnel, the driving force for differential deformation is the variability in stiffness of the soil, the differential loading conditions (back-silting and tidal forces) and the variations in the foundation (gravel) layer that supports the tunnel elements. As mentioned earlier, variability in mean soil stiffness, due to variation in soil thickness, will be the main focus in the initial deformation part of this research. An overview of the deformation is presented in figure 2.21. The main takeaway from this research is the probabilistic design methods used to address the uncertainties in soil stiffness (Y.-N. Wang, L.-C. Wang, and Zhao 2023).

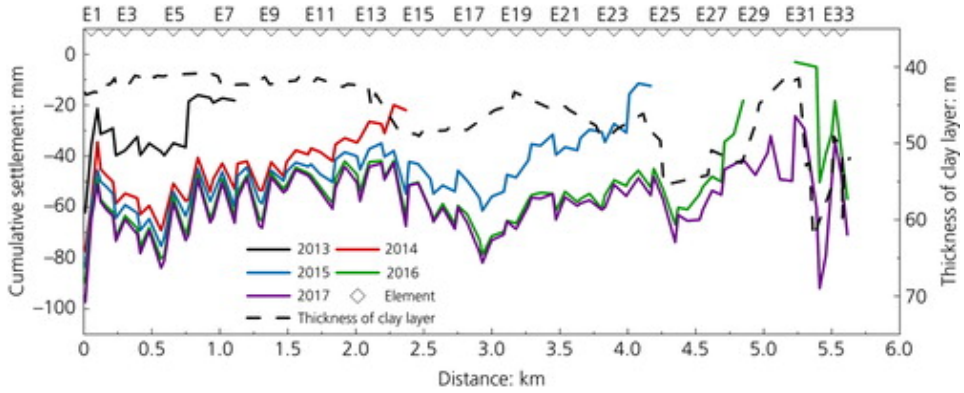


Figure 2.21: Settlement of the Hong Kong - Zhuhai-Macao Tunnel from (Y.-N. Wang, L.-C. Wang, and Zhao 2023)

3

Specifics of the Fehmarnbelt tunnel

The Fehmarnbelt Tunnel represents a significant engineering achievement, connecting Denmark and Germany through an 18-kilometer immersed tunnel. This chapter provides a detailed overview of the tunnel's design, construction process, geological conditions, and the specific zones that will be analyzed in this study. Understanding these specifics is crucial for assessing the impact of soil thickness variability on secondary consolidation of the soil beneath the tunnel elements.

3.1. Introduction Fehmarnbelt Project

The Fehmarnbelt Tunnel is set to become the longest immersed tunnel in the world, spanning the Fehmarnbelt strait between Rødbyhavn in Denmark and the island of Fehmarn in Germany. This section introduces the project, highlighting its significance and the challenges posed by the unique geological conditions of the area.

The tunnel consists of a two-lane highway in both directions and two railway tracks. The tunnel is made up of 79 elements and 10 special elements. The special elements are used to store equipment for the tunnel itself. The subsoil present at the site is a difficult soil in which to construct an IMT, this is because it consists mostly of glacial clay material. The foundation of the tunnel is made up of gravel beds that will be laid down in a pre-dug trench. The trench is constructed by removing 15 million m^3 of sand, which will have a large impact on the state of the soil (Kammer et al. 2012).

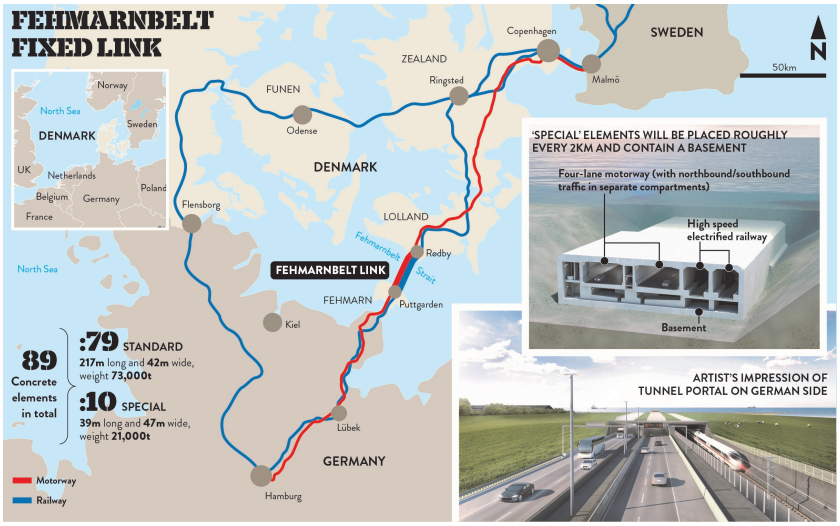


Figure 3.1: Fehmarnbelt fixed link (Hakimian 2024)

In figure 3.1 the connection that is constructed is shown. The blue section on the red route is the IMT that needs to be placed. The red line itself is the complete infrastructure that is constructed to replace the blue route which is significantly longer. The tunnel is expected to be completed in 2029, making it the largest immersed tunnel in the world. Tunnels that come close to this length are listed in table 3.1, including their length, construction period, and difficulties during design/construction. This table is used in the research as a reference for the design choices and methods that were used to overcome certain difficulties considering the ground conditions in these cases.

Table 3.1: Case studies

Name	Country	Length [m]	Ground profile	Difficulties	Source
Marmaray tunnel	Turkey	13600	Geological rock basin filled up with sediments from river. So hard material on the sides and sand and clay in the middle.	• Seismic activity in area • Corrosive surroundings • Many historical buildings present on the surface • High groundwater level	(Sakaeda 2005)
Shenzhen–Zhongshan Bridge	China	6700	Fully and highly weathered residual soil deposits to its sides and mucky soil in the middle	• Over-large burial layer risk • Differential settlements • High water level • Widest immersed tunnel in the world • Complex geological conditions	(Fu et al. 2020)
Transbay tube	USA	5700	Pleistocene sand and gravels overlain by muck from the bay	• Liquefiable backfills • Seismic activity • Founded on primarily loose material	(Commission 2015)
Lingding-Tonggu Channel Tunnel	China	5664	Medium to coarse sand as base layer filled up with mucky soil to ground level	• Different foundations used along length • Large back-silting • Large differential settlements	(Hu et al. 2018)
Drogdentunnelen	Sweden	3510	(Sub)- arctic soil	• Subartic conditions • Backfill of rock, • High accuracy in gravel foundation	(PPIAF 2020)

3.2. Tunnel dimensions

The tunnel consists of 79 regular elements and 10 special elements, each with specific dimensions and functions. This section provides detailed measurements of these elements, including their length, width, and height, and explains their roles within the overall tunnel structure.

The tunnel elements are illustrated in Figure 3.2 for the regular element and Figure 3.3 for the special element. Each regular element follows standard dimensions: 217 meters in length, 42.2 meters in width, and 8.9 meters in height. In particular, the tunnel configuration consists of the following components:

- Road Tubes (Two in total)
- Escape Tube (Positioned centrally)
- Rail Tubes (Two in total)

In addition, the design incorporates 10 special elements dedicated to housing essential service systems. These special elements measure 39 meters in length, 45 meters in width, and 13.1 meters in height, as depicted in Figure 3.3.

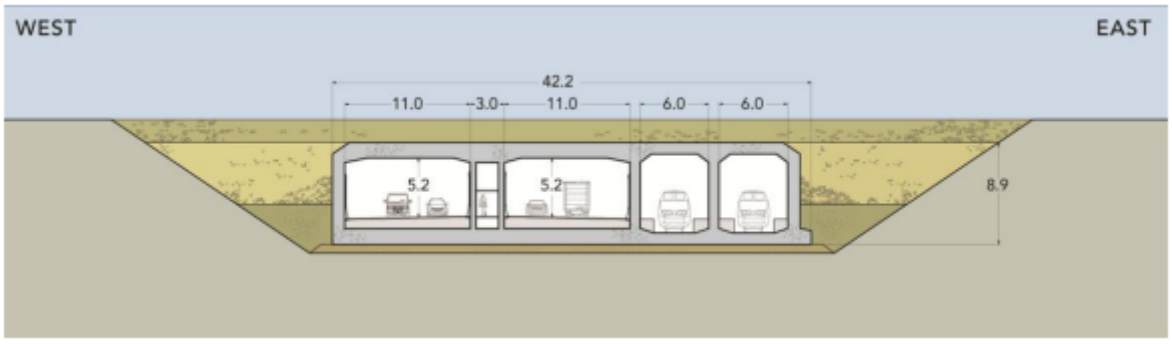


Figure 3.2: Size regular element Fehmarnbelt tunnel (Andersen, Iversen, and Putten 2012)

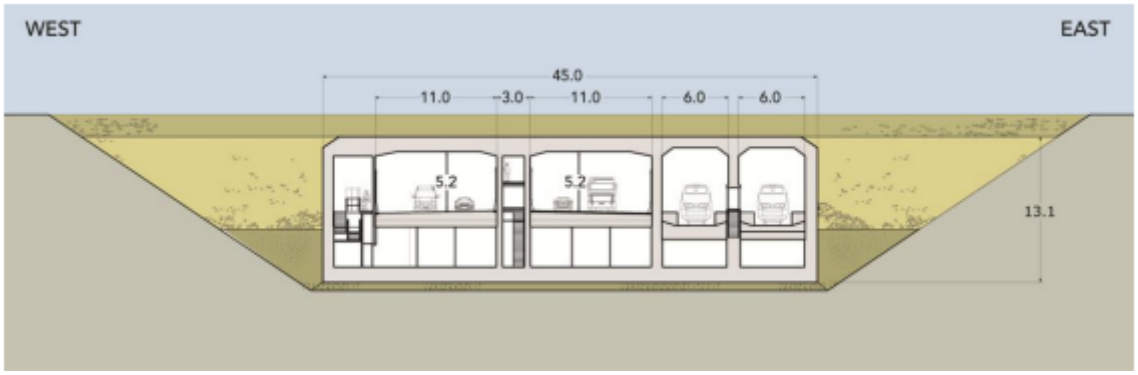


Figure 3.3: Size special element Fehmarnbelt tunnel (Andersen, Iversen, and Putten 2012)

The overall structure, which spans its entire length, exhibits the arrangement shown in figure 3.4. In this representation, the bold black lines denote the special elements, while the thinner lines represent the standard elements (Kammer et al. 2012).

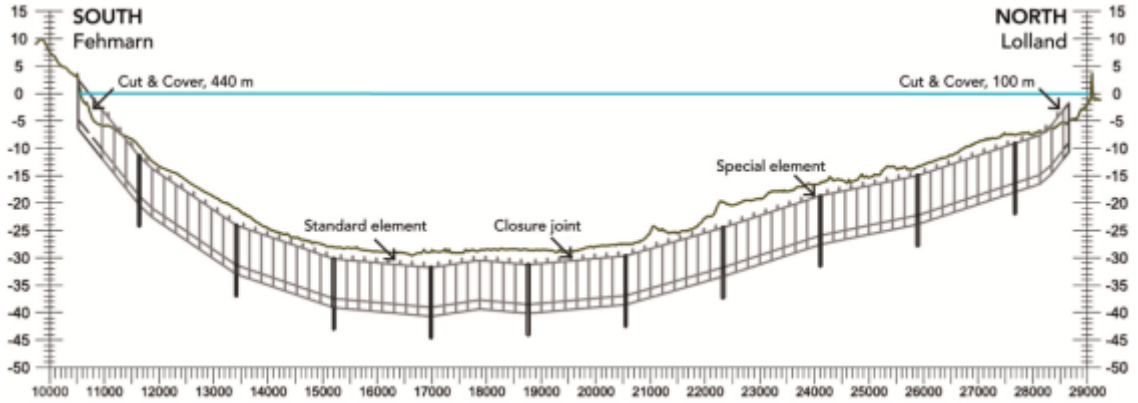


Figure 3.4: Full profile elements Fehmarnbelt tunnel (Kammer et al. 2012)

The other dimensions of the tunnel are discussed in Chapter 4 in the buoyancy calculation part for the calculation of the total concrete in each element and the buoyant force it will produce.

3.3. Loading phases

The construction of the Fehmarnbelt Tunnel involves several loading and unloading phases, each impacting the soil beneath the tunnel. This section outlines these phases, detailing the steps involved in trenching, placement of the gravel layer, immersion of the tunnel element, and filling the trench. The section also discusses the time and magnitude of loading for each phase.

3.3.1. Placement tunnel element

The process of placing tunnel elements involves precise steps to ensure stability and alignment. This subsection describes the preparation of the gravel bed, the immersion of the tunnel elements using pontoons, and the subsequent filling of the trench.

The steps involved in placing the Immersed Tunnel (IMT) for the Fehmarnbelt Fixed Link project are as follows:

1. Gravel Bed Preparation and trenching:
    - The IMT will rest on a gravel bed with a uniform thickness of 0.75 meters. However, the thickness of this gravel bed can vary along the length of the tunnel due to differences in the stiffness of the soil and the depth of the trench under each element.

- The initial step involves excavating the soil to create a trench. The height of the trench will differ at various locations due to the inherent variability of the stiffness of the soil and the limitations of the trenching equipment. (backhoe) Shown in figure 3.5

2. Gravel Layer Placement:
   - A barge equipped with a fall pipe is used to precisely deposit the gravel layer. Shown in figure 3.5.
   - This layer serves as a stable foundation for the elements of the tunnel.
3. Tunnel Element Immersion:
   - The actual tunnel elements are then positioned using immersion pontoons. Shown in figure 3.6
   - The immersion process ensures accurate alignment and placement.
4. Filling of the trench:
   - After the elements of the tunnel have been installed, the trench is filled following these steps:
     - Locking Fill: Used to secure the elements in position.
     - General Fill: Fills the sides of the trench.
     - Protection Layer: Shields the structure from potential damage caused by ships passing overhead.

This final configuration ensures stability, functionality, and protection for the Fehmarnbelt Fixed Link, as depicted in Figure 3.7.

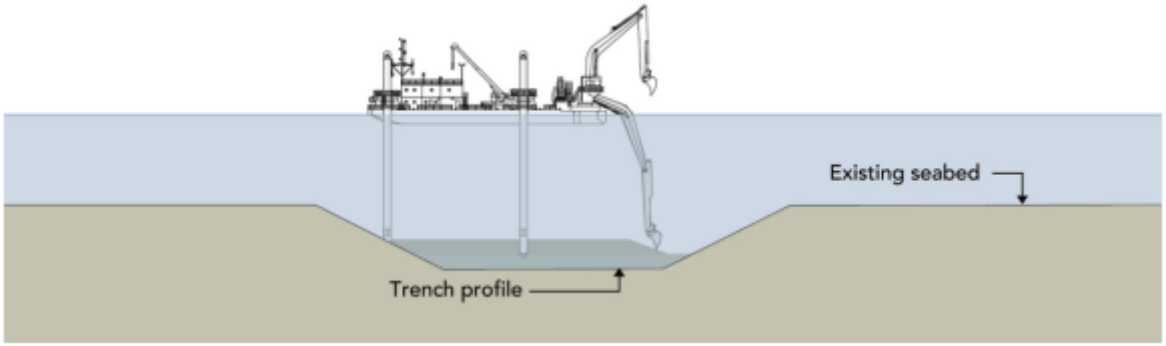


Figure 3.5: Trenching before placing tunnel part (Andersen, Iversen, and Putten 2012)

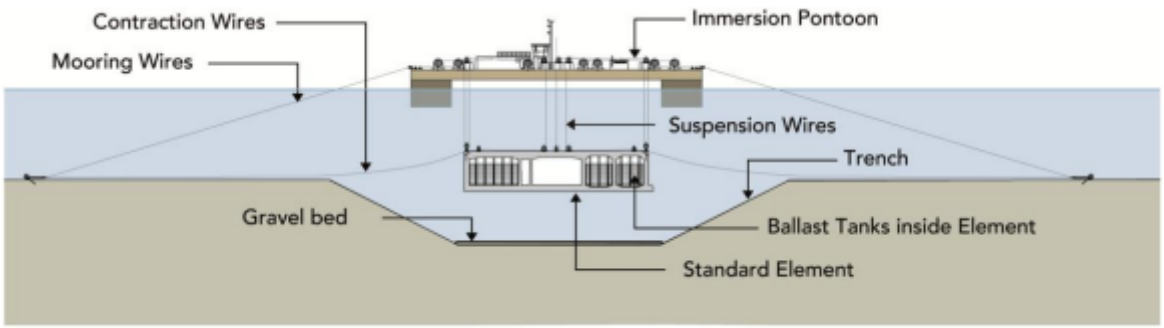


Figure 3.6: Placing tunnel element on foundation (Andersen, Iversen, and Putten 2012)

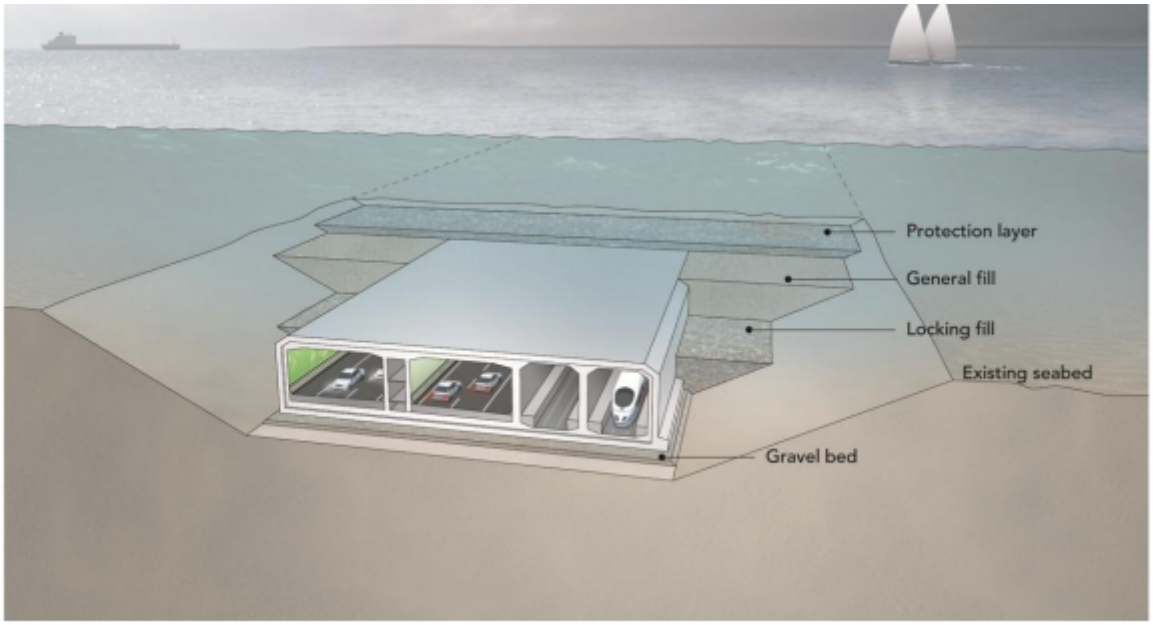


Figure 3.7: Filling up of trench after placement of tunnel part (Andersen, Iversen, and Putten 2012)

3.3.2. Loading time

Each loading phase has a specific duration, which affects the soil consolidation process. This subsection provides a timeline for the construction phases, highlighting the importance of timing in soil deformation analysis.

There are five main loading/unloading phases in the construction progress of the immersed tunnel, as concluded from the previous section (3.3.1). An example of a load versus time graph is shown in Figure 3.8. The t_c in this figure is the duration of a construction period. In other words: the time it takes for the complete load to be placed on the soil. The t represents the total time that a phase lasts, which is until the next phase begins.

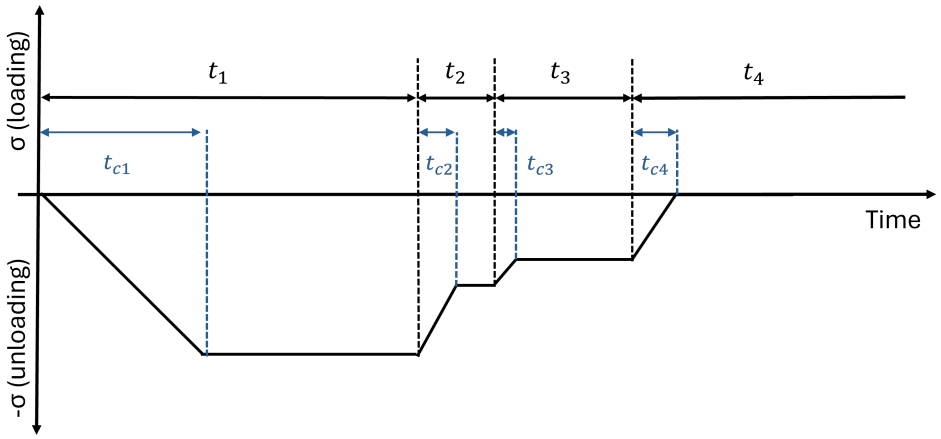


Figure 3.8: Example of loading times and magnitudes for 4 (un)loading stages

The corresponding times for these construction and total phases are chosen as follows:

Table 3.2: Times for element placement phases

(Un)loading phase	Construction time in [days]	Total time in [days]
Trenching of all elements	1095	1150
Placement of total foundation layer	50	150
Placement of one tunnel element	1	150
Placement of total protection layer [1]	50	150
Placement of total backfill	50	No new phase, so no end-time

[1] the protection layer will only be present on the tunnel parts that lie beneath the part of the channel where the ships sail.

The times mentioned in table 3.2 are approximate and should be used as a general guideline. The duration of secondary consolidation is significantly longer, typically ranging from around 30 to 100 years, compared to the number of days indicated here.

3.3.3. Magnitude of loading/unloading

The magnitude of loading and unloading varies across different phases of construction. This subsection quantifies these magnitudes, explaining their impact on soil consolidation.

The five main loading/unloading phases have the following magnitudes:

Table 3.3: Construction magnitudes for element placement phases

(Un)loading phase	Magnitudes in [kPa]
Trenching	Varies per zone [1]
Placement of total foundation layer	21 [2]
Placement of special element	34.79 [3]
Placement of regular element	23.93 [4]
Placement of protection layer	39 [5]
Placement of total backfill	Varies per zone [6]

[1] The trench height times the unit weight of the soil(s) that are removed in that zone.
[2] The unit weight of a gravel layer defined by the standard (NEN 1997) multiplied by the thickness of 1 meter of the foundation layer.
[3] The load of the special element consists of the load of the element and the ballast concrete inside the element to keep it in place.
[4] The load of the regular element consists of the load of the element and the ballast concrete inside the element to keep it in place.
[5] The unit weight of a denser gravel/rock layer defined by the standard (NEN 1997) multiplied by the thickness of 1.5 meters of the protection layer.
[6] The load of soil(s) removed on top of the tunnel and placed back.

This results in the following load profiles over time for all zones (Figure 3.9). The loading profile itself is constructed with the help of Fourier series, which is explained in more detail in Section 4.5.1.

3.4. Geological conditions

The geological conditions of the Fehmarnbelt area are complex, featuring a variety of soil types and formations. This section provides an overview of the geological profile, including post-glacial deposits, glacial tills, Paleogene clay, and Cretaceous chalk. Understanding these conditions is essential for accurate soil analysis.

In figure 3.10 the placement of the tunnel in the existing subsoil is shown. The small squares in the tunnel represent the regular elements, and the longer and thicker lines represent the special elements present in the tunnel structure. In figure 3.11 a top view of the tunnel placement in existing soil

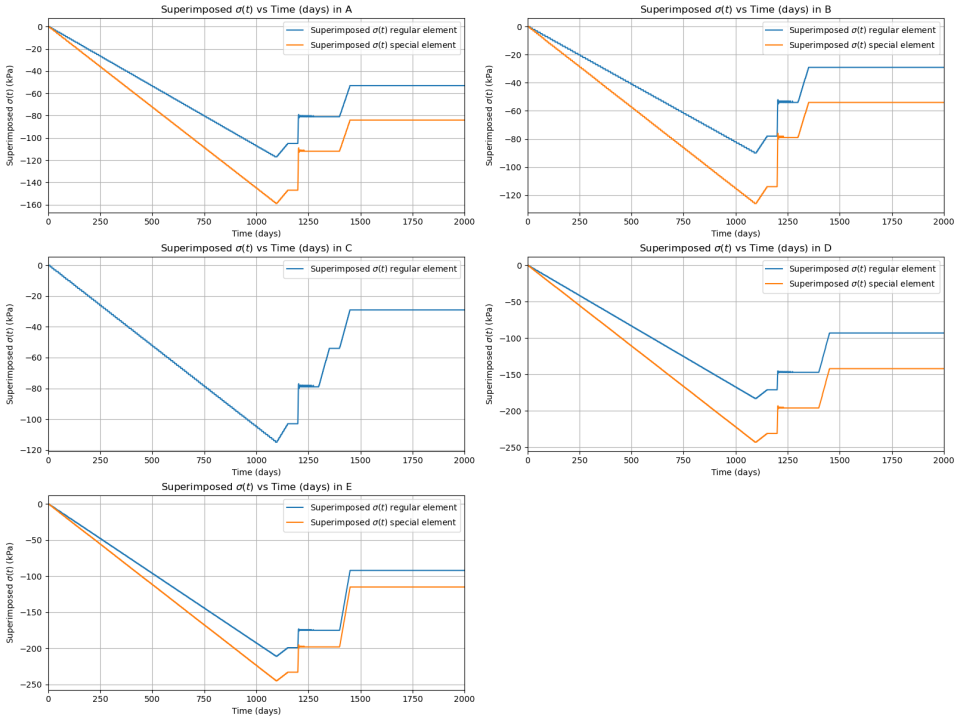


Figure 3.9: Loading profile over time for all the zones

conditions is shown to give an overview of the complete geological condition. The complete profile consists of four main soil units:

- Post-glacial and late glacial deposits. Accumulated after the last ice age.
- Glacial tills and meltwater deposits. Formed by glacial action.
- Paleogene clay. A type of clay with very high plasticity.
- Cretaceous chalk. A distinctive rock layer.

The Post glacial and Late glacial deposits primarily occur in the deeper section of the Fehmarnbelt, specifically in the central-south part of the alignment. These deposits originate from a former lake. Outside of this geological basin (a geological basin is a large rock formation that collects sediment eroded from continents and precipitation), there are only some relatively young marine deposits. Within the geological basin area, the dominant soil composition consists of soft silts and clays, although localized sand bodies are also present. These deposits reflect the dynamic history of the Baltic region during the Post-glacial and Late glacial periods, with the environment changing multiple times between lake and marine conditions(Kammer et al. 2012).

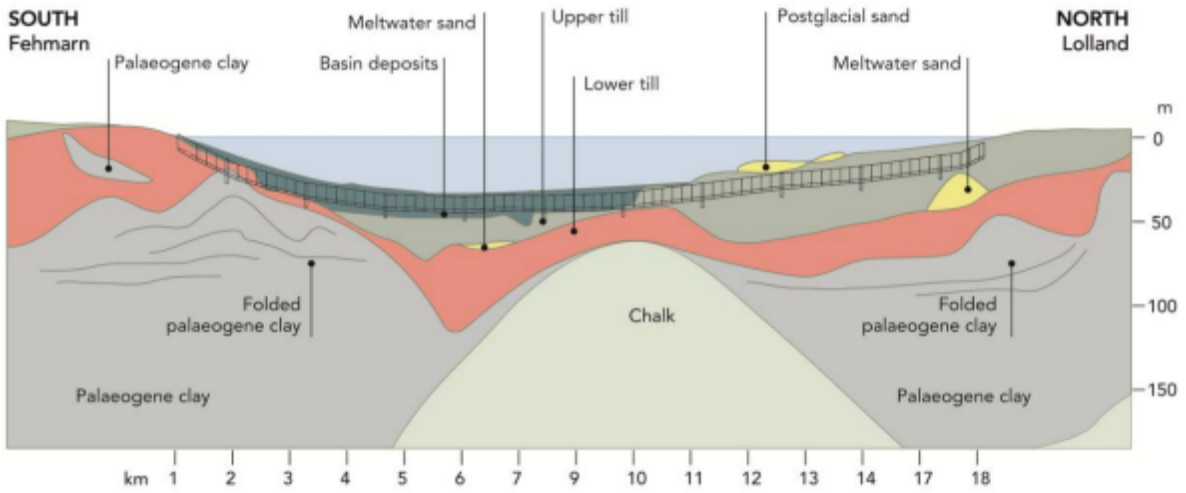


Figure 3.10: Geological profile Fehmarnbelt (Yumpu.com 2011)

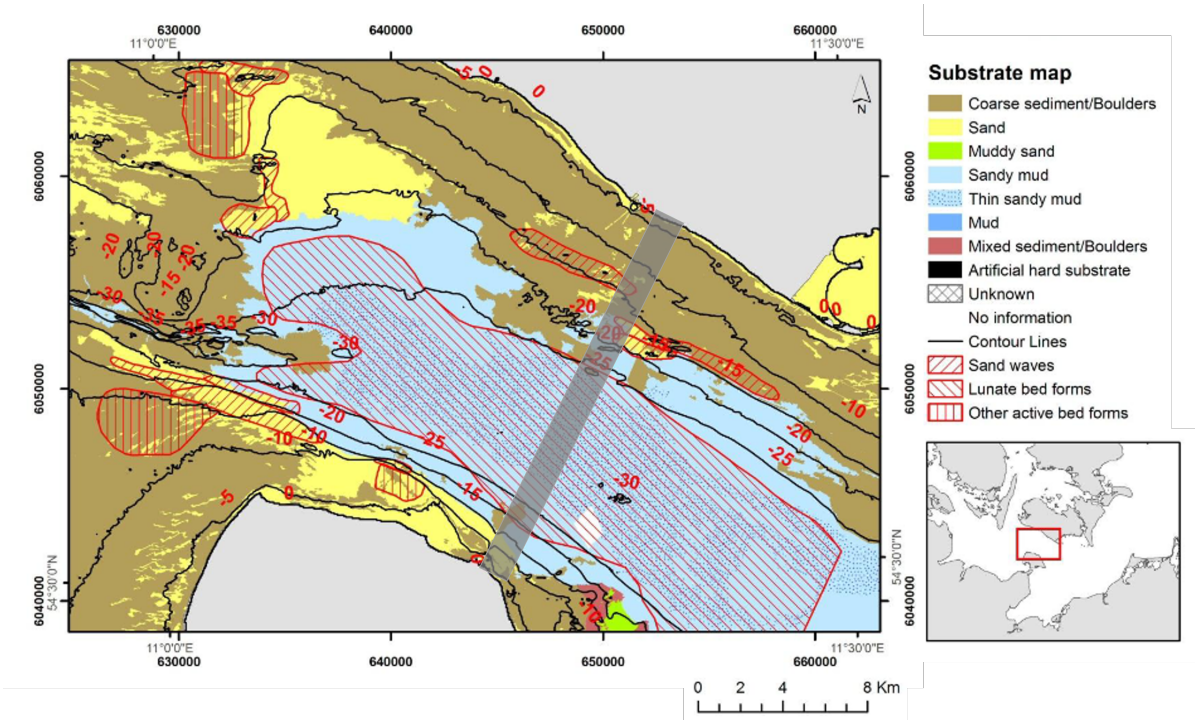


Figure 3.11: Top view placement tunnel geological situation from (FEHY 2013)

During the drilling campaign, geotechnical investigations revealed the presence of several glacial deposits. These include two to five tills from different glacial episodes and two to three meltwater deposits. In particular:

- The lower till typically exhibits medium to high plasticity.
- Gyttja, silts, clays and sands are present within the deposit.
- In contrast, the upper till is characterized by low plasticity and remarkable hardness, potentially even having some degree of cementation (Kammer et al. 2012).

The Paleogene clay exhibits high to very high plasticity and frequently contains a notable proportion of the clay mineral group called smectite. This composition gives the clay certain properties that are similar to those of bentonite. Furthermore, microfossil studies have identified deposits from five geological

formations, listed in order of age: Lillebælt, Røsnæs, Ølst, Holmehus, and Æbeløl (Krogsbøll, Hededal, and Foged 2012).

Significant and unexpected findings emerged from the investigations: within the upper 10-30 meters, the soil has undergone substantial disturbance and folding due to ice pressure during the Quaternary period. Beneath this layer lies Cretaceous chalk, characterized by its fine-grained texture and occasional moderate flint content (Krogsbøll, Hededal, and Foged 2012).

According to Figure 3.10, the chalk layer has been encountered by the boreholes in a relatively small area near the center of the Fehmarnbelt. Specifically, within this central part, the surface of the chalk within a dome-shaped structure lies approximately 20 meters below the seabed. However, beyond this central area, the geophysical profiles reveal a rapid decline in the chalk surface to depths exceeding 200 meters. The intriguing reason for the elevated position of the chalk lies in the presence of a thick salt layer more than 1 kilometer below the region. Portions of this salt layer have been lifted, forming a *'pillow'* beneath the area, gradually lifting the overlying deposits. Remarkably, this upward movement is likely still ongoing. Observations from the geological model and studies of similar structures in neighboring regions suggest that the heave rate is probably less than 0.1 millimeter per year and therefore will not cause structural problems (Kammer et al. 2012).

As part of the investigations, a desk study was conducted to assess seismic activity in the Fehmarnbelt area. The conclusion drawn from this study is that the Fehmarnbelt region is in an extremely low seismicity zone. Consequently, the risk posed by earthquakes to fixed link structures is minimal (*Fehmarn Belt fixed link* 2024). Due to this fact, seismic activity will not be considered in this research.

3.4.1. Soil characteristics

Each type of soil in the Fehmarnbelt area has distinct geotechnical properties. This subsection details these characteristics, including overconsolidation ratios, unit weights, and consolidation indices, which are critical for modeling soil behavior.

Yumpu (Yumpu.com 2011) performed research on the soil present in the Fehmarnbelt link and classified the soil layers, including their most important soil parameters derived from CPT and laboratory results. The overview of those geotechnical classification properties is given in table 3.4 and 3.5. The table given here has the information that was present from the project.

Table 3.4: Geotechnical characteristics of sub-soils present Fehmarnbelt (Yumpu.com 2011)

Name	Color	OCR [-]	γ_{sat} $[kN/m]$	η $[-]$	E_{oed} $[kPa]$	m_v $[1/kPa]$
Water	lightblue	-	9.81	-	-	-
Basin Deposits	dimgrey	1.4	17.75	1	22430	4.46E-05
Upper till	darkgrey	15	23	1	600000	1.67E-06
Meltwater sand	lemonchiffon	1.1	18	1	60000	1.67E-05
Lower till	lightcoral	7.75	22.6	1	83000	1.2E-05
Palaeogene clay	lightgrey	2.5	18.93	1	71500	1.4E-05
Chalk	gainsboro	8	18.7	1	18700	5.35E-05
Postglacial Sand	orange	1	18.3	1	40000	0.000025

Table 3.5: Geotechnical characteristics of sub-soils present Fehmarnbelt (Yumpu.com 2011)

Name	e_0 $[-]$	C_e $[-]$	C_c $[-]$	C_{ae} $[-]$	C_v $[m2/day]$	v^1 $[-]$
Water	-	-	-	-	-	-
Basin Deposits	0.8	0.0021	0.0547	0.0029	0.432877	0.45
Upper till	0.27	0.0027	0.04	0.0026	0.873973	0.2
Meltwater sand	0.4	0.0001	0.0001	0.029	0	0.5
Lower till	0.3	0.0067	0.051	0.0026	0.435616	0.2
Palaeogene clay	1.035	0.063	0.15	0.002	0.479452	0.2
Chalk	0.89	0.0001	0.0001	0.0002	0	0.5
Postglacial Sand	0.76	0	0	0.029	0	0.5

[1] Poisson ratio

3.5. Trench depth

The depth of the trench varies along the length of the tunnel, influencing the response of the soil to loading. This section explains the trench depth requirements for regular and special elements and discusses the impact of trench depth variability on soil consolidation.

The variation of this trench depth will be ignored in this thesis. The variation in depth of the trench will be leveled with the placement of the gravel foundation layer. The only effect is that this gravel layer will have a variable thickness over its width/length. The effect of the variation in this foundation layer (~ 1 meter soil) will be insignificantly small compared to the variation in the thickness of the soil layers. The definition of the depth of the trench for a regular and special element is shown in Figure 3.12. This leads to a trench depth of 11.4 meters for the regular element and a trench depth of 15.96 meters for the special element. The trench will gradually decrease from a regular element to a special element, this gradual decrease in soil thickness and soil load is not taken into account in this research. The trenching will cause an unloading of the soil and the heave (upward deformation of the soil) will start to accumulate.

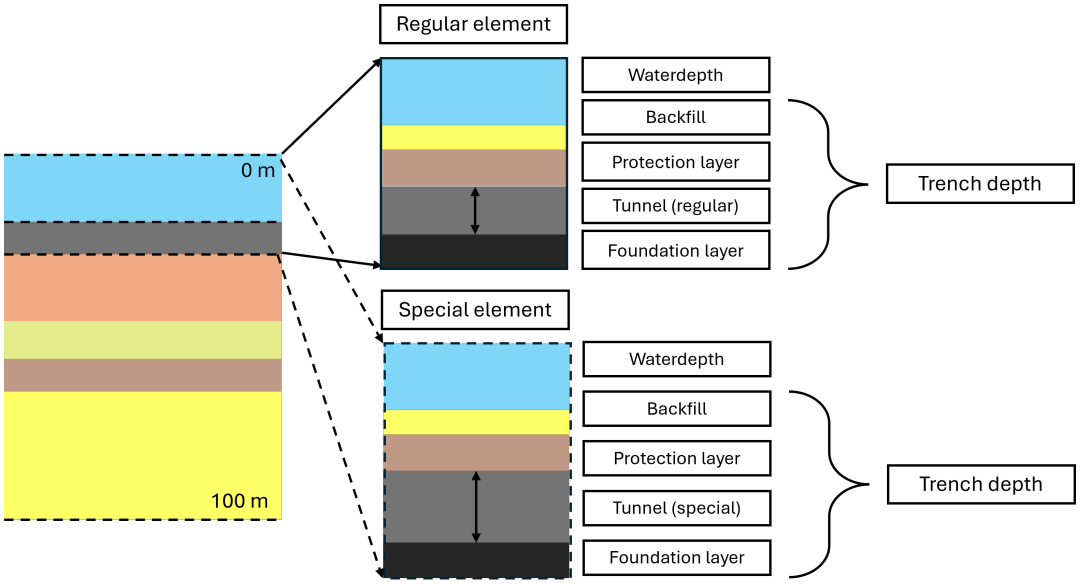


Figure 3.12: Trench depth visualization for special and regular element

3.6. Zone subdivision

To manage the complexity of soil analysis, the tunnel profile is subdivided into specific zones, each with unique soil compositions and conditions. This section introduces these zones, explaining their significance and the rationale behind their selection.

The effect of the variation in the thickness of the soil layer can be analyzed throughout the entire length of the tunnel, but this calculation procedure would cost a lot of time and calculation power to establish. The solution to save time and computational power is the subdivision of the total soil profile into zones of interest. All of these zones of interest contain special conditions that give them significance for the total analysis. The zones are placed as shown in figure 3.13. Together, they form a representative configuration for the whole profile.

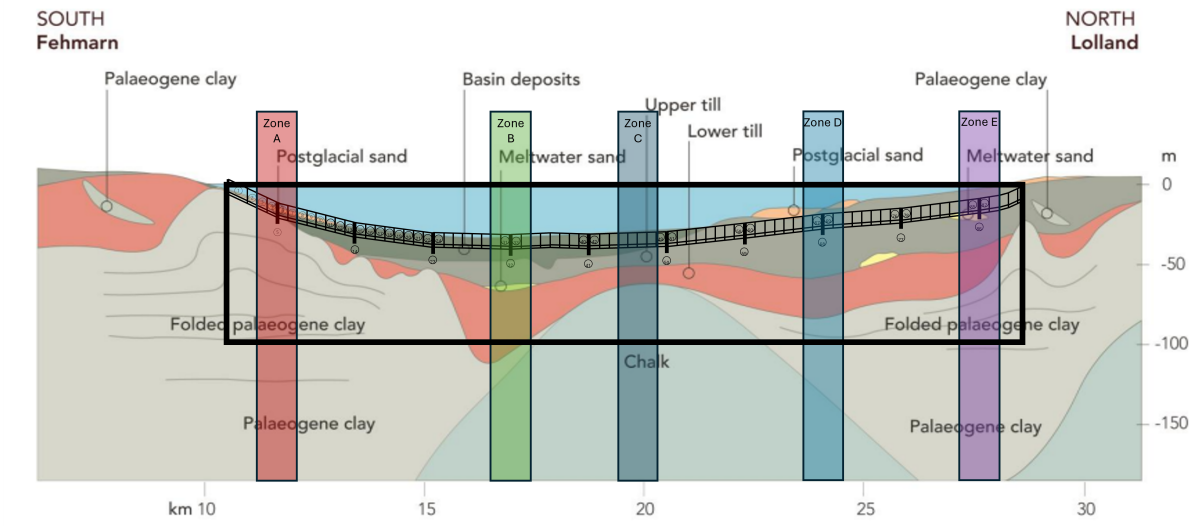


Figure 3.13: Chosen zones of interest to represent whole profile (original figure from (Yumpu.com 2011))

In this division, the important points to note are as follows.

Zone A consists of a single soil type, making it a stable baseline for analysis. Uniformity in the thickness and parameters of the soil layers ensures consistent behavior, providing a reliable reference point. The presence of only cohesive layers results in a longer drainage path, which impacts the consolidation process.

Zone B features a protection layer on top of the tunnel to protect the structure from potential damage caused by ships. This zone includes a basin deposit layer, which influences the soil's consolidation behavior. Drainage within the soil profile is possible, resulting in a smaller drainage path. This drainage path is described in further detail in Chapter 4. In addition, soil types differ between regular and special elements, adding complexity to the analysis.

Zone C also has a protection layer to prevent damage from ships. Unlike other zones, it contains only regular tunnel elements. In this zone, there is a significant volume of chalk present, which affects the consolidation properties of the soil and the overall behavior.

Zone D is characterized by the presence of only cohesive layers, leading to a longer drainage path. In this zone, a large amount of soil is excavated, which affects the consolidation process. Additionally, a substantial amount of backfill is used, further influencing the soil's behavior.

Zone E includes a meltwater sand layer that is present only in the regular element and removed for the special element. Similarly to zone D, a large amount of soil is excavated and a significant amount of backfill is used. This zone is notable for its high diversity of soil types, which adds complexity to the consolidation analysis.

A complete overview of the soil layers present in each zone is given in the graphs in Figure 3.14.

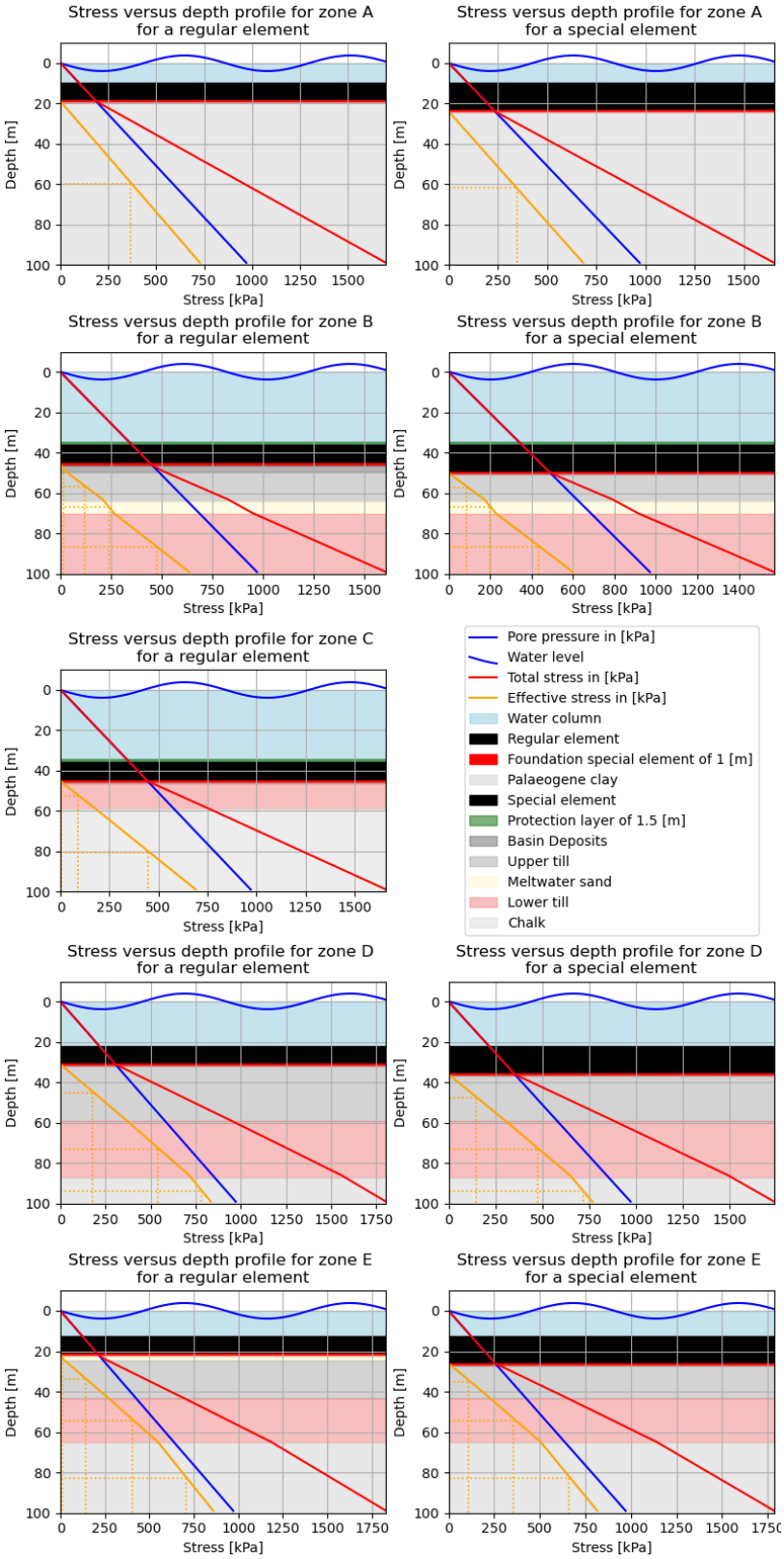


Figure 3.14: Stress distribution with depth for regular and special elements in every zone

4

Methodology

This chapter covers the explanation and reasoning of all the models used in the process of defining the variability in secondary consolidation. The models covered in this chapter are: a buoyancy calculation for the loads of the elements in section 4.2, the current stress state of the soil profile in section 4.3, the Timoshenko Beam on Kerr Foundation - model (TBKF-model) in section 4.4, the Conte & Troncone model to define the primary consolidation (Conte and Troncone 2006) in section 4.5, the Feng et al. model to define the secondary consolidation (Wei-Qiang Feng et al. 2020) in section 4.6 and the Monte Carlo simulation to simulate different layer thicknesses following a mean and standard deviation in section 4.8.

4.1. Overview calculation model

The calculation model integrates various analytical methods to assess soil deformation under the Fehmarnbelt Tunnel. This section provides an overview of the steps involved in the model, from the initial determination of the soil stress state to the final results of the consolidation. It highlights the importance of each step in achieving accurate and reliable predictions.

The complete overview of the steps taken in the model is shown in Figure 4.1. As can be seen, the input of the model is the varying soil thicknesses and the output is the effect that has on the long-term settlement of the soil underneath the immersed tunnel in the longitudinal direction.

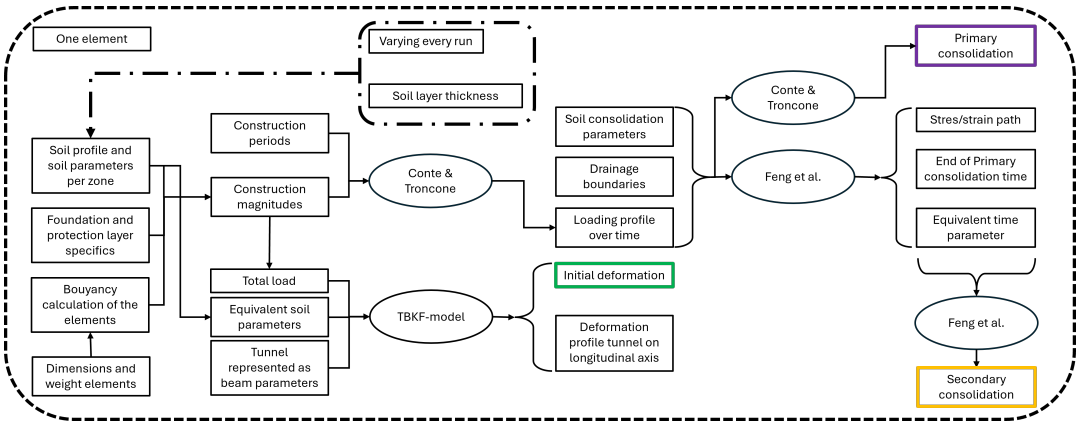


Figure 4.1: Overview of all calculation steps in model

4.2. Buoyancy calculation for loads elements

Buoyancy plays an important role in the stability of immersed tunnel elements. This section explains the principles of buoyancy and its application in calculating the forces exerted by the tunnel elements

on the soil. It includes detailed formulas and results for both regular and special elements, emphasizing the importance of buoyancy in soil deformation analysis.

*Buoyancy is a force exerted by a fluid (such as water or air) that opposes the weight of an object immersed in it. This force is what causes objects to move upward or sink in a fluid (*Buoyancy | History, Science, & Applications | Britannica *2025).* The upward force of the fluid is equal to the weight of the fluid displaced by the object. The same holds for an immersed tunnel, the force acting on the tunnel part is equal to the volume of the water replaced by the tunnel volume. The tunnel is weighted with ballast concrete to counteract this force and to keep it at the bottom of the water body where it is placed. The buoyant force is calculated as shown in Equation 4.1.

$$F_b = \rho * V_w * g \tag{4.1}$$

where

F_b: Buoyant force $[kN]$
g: Gravitational constant $[m/s^2]$
ρ: Density of the fluid $[kN/m^3]$
V_w: Volume of water replaced $[m^3]$

The buoyancy force for the tunnel is calculated by calculating the total volume of the tunnel and multiplying this by the unit weight of the water (which is 9.81 kN/m^3). The resulting force that the tunnel will exert on the soil column is the buoyancy force subtracted by the weight of the tunnel and the ballast concrete. The results of these calculations are shown in Table 4.1.

Table 4.1: Buoyancy calculations results for special and regular element IMT

Parameter	**Regular element**	**Special element**
Weight of structural concrete $[kN/m^3]$	24	24
Length of element $[m]$	217	39
Total volume of concrete $[m^3]$	29937.27	8862.38
Total buoyancy volume $[m^3]$	81250.43	23804.03
Buoyancy force $[kN]$	-840942	-246372
Structural concrete force $[kN]$	718494	212697
Resulting force $[kN]$	72908	11188

4.3. Current stress state soil profile

Understanding the current stress state of the soil is essential for accurate consolidation predictions. This section describes the methods used to determine the initial stress state of the soil profile, including the impact of soil thickness and composition on stress distribution.

4.4. Timoshenko beam on the Kerr foundation model

The Timoshenko beam and Kerr foundation models are integral to the analysis of the interaction between soil and structure. This section provides a detailed explanation of these models, including their technical aspects, advantages, and limitations. The chapter discusses the coupling of the models and their application in predicting soil deformation under tunnel elements.

The Timoshenko beam model describes the forces and deformations of a beam subjected to a load, and the Kerr foundation model is a three - parameter model used to define the interaction between a beam and the soil layers underneath.These two models combined will be called the TBKF model in the rest of this thesis.

4.4.1. Technical aspects of the TBKF model

Technical aspects Timoshenko beam model

The Timoshenko beam model is shown in figure 4.2, where M is the moment present in the structure, V represents the shear force, $q(x)$ is the distributed load present and w is the deformation of the beam itself.

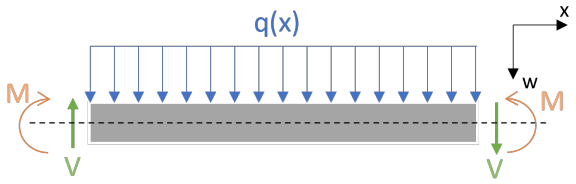


Figure 4.2: Timoshenko beam model

The governing equations for this model are the equations 4.2 and 4.3. 4.2 calculates the shear force in a Timoshenko beam and 4.3 calculates the moment present in a Timoshenko beam, subjected to a certain load.

$$V(x) = \frac{dM(x)}{dx} \tag{4.2}$$

$$\frac{d^2M(x)}{dx^2} = -q(x) \tag{4.3}$$

where

$V(x)$:	Shear force in beam $[kN]$
$M(x)$:	Moment in beam $[kNm]$
x:	Distance from left end of beam $[m]$
$q(x)$:	Uniform load applied to beam $[kPa]$

A smaller part of this beam is represented in Figure 4.3. The bending moment and shear force of this small element are expressed as shown in equation 4.4, 4.5 and 4.6

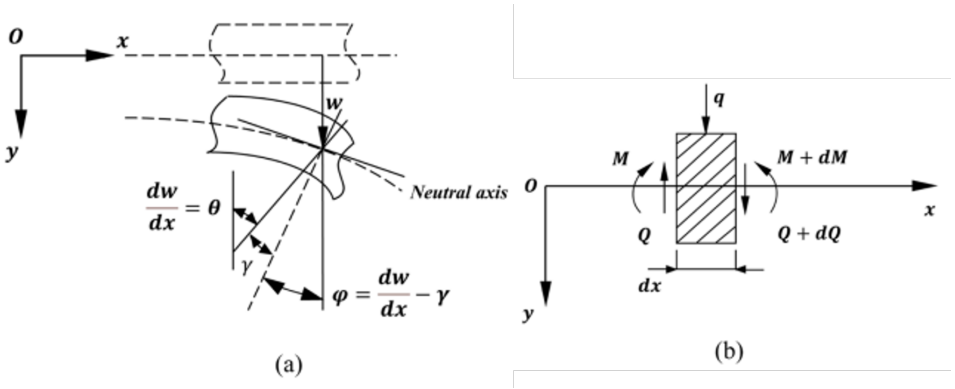


Figure 4.3: Small part of the Timoshenko beam with (a) deformation and (b) stresses (Y. Wang et al. 2023)

$$M(x) = EI\kappa = -EI * \frac{d\phi(x)}{dx} = -EI\frac{d^2w}{dx^2} + EI\frac{d\gamma}{dx} \tag{4.4}$$

$$V(x) = GA\gamma / f_s = GA * \left(\frac{dw(x)}{dx} - \phi(x) \right) / f_s \tag{4.5}$$

$$\phi(x) = \frac{dw(x)}{dx} - \frac{1}{\kappa GA} + \frac{dM(x)}{dx} \tag{4.6}$$

where

$V(x)$:	Shear force in beam $[kN]$
$M(x)$:	Moment in beam $[kNm]$
x:	Distance from left end of beam $[m]$
$w(x)$:	Deformation of beam $[m]$
$\phi(x)$	Rotation angle of the cross section $[rad]$
EI:	Bending stiffness of the element in $[kNm^2]$
GA:	Shear stiffness of the element in $[kNm^2]$
κ:	Curvature of the neutral axis $[-]$
$\frac{dw}{dx} - \phi(x)$:	Shear angle of the cross-section due to shear deformation $[rad]$
f_s:	The non-uniform distribution coefficient of the shear stress in the section $[-]$

It is assumed that the cross section of the beam (and tunnel element lengthwise) is constant over its length, with this assumption one derives at the material law for a Timoshenko beam shown in equation 4.7.

$$\frac{d^2w(x)}{dx^2} + \frac{dM(x)}{dx} - \frac{d}{dx}\left(\frac{V(x)}{\kappa GA} \right) = 0 \tag{4.7}$$

The application of equation 4.3 yields the function in equation 4.8 and the substitution of equation 4.3 and equation 4.8 into the material law yields equation 4.9. The integration constants C_1 to C_4 can be calculated with the boundary conditions following from the supports at the ends of the beam (fixed, simply supported, or free).

$$M(x) = -\iint q(x)dxdx + C_1x + C_2 \tag{4.8}$$

$$EIxw(x) = -\iint \left[\frac{EI}{\kappa GA} q(x) + M(x) \right] dxdx + C_3x + C_4 \tag{4.9}$$

Technical aspects of the coupling of models and the Kerr foundation model

The next step is the interaction between the structure itself and its foundation layer. This can be defined via several models, for example, the Kerr foundation, the Winkler foundation, the Vlasov foundation, or the Pasternak foundation model. The Kerr foundation is chosen here; an explanation for that choice will be given in Section 4.4.2.

The coupled Timoshenko beam and Kerr foundation model is shown in Figure 4.4. The following differential equations are obtained via the principle of stationary potential (for more information, see (Morfidis 2007)).

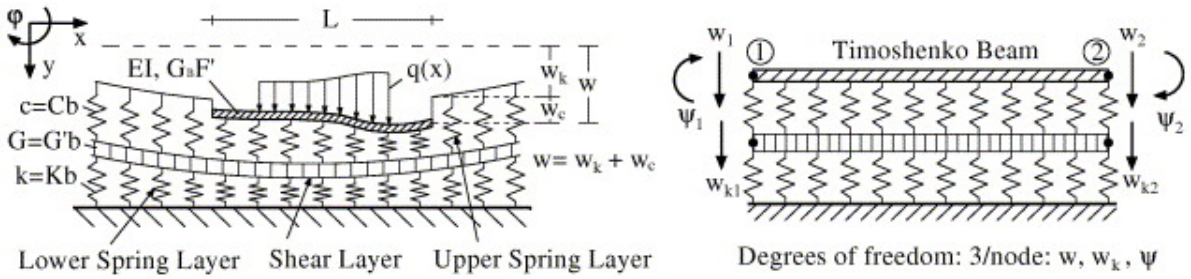


Figure 4.4: Coupled Timoshenko beam and Kerr foundation model from (Morfidis 2007)

$$c(w - w_k) - \Phi \frac{d}{dx}\left(\frac{dw}{dx} - \psi\right) - q(x) = 0 \tag{4.10}$$

$$w = \left(1 + \frac{k}{c}\right) w_k - \left(\frac{G}{c}\right) \frac{d^2 w_k}{dx^2} \tag{4.11}$$

and

$$-EI \frac{d^2\psi}{dx^2} + \Phi\left(\psi - \frac{dw}{dx}\right) \tag{4.12}$$

By combining the equations above, an uncoupling between the unknown displacements can be made, as depicted in equations 4.13 and 4.14.

$$-\frac{EIG}{c}\frac{d^6 w_k}{dx^6} + \left[EI\left(1 + \frac{k}{c}\right) + G\frac{EI}{\Phi}\right]\frac{d^4 w_k}{dx^4} - \left[G + k\frac{EI}{\Phi}\right]\frac{d^2 w_k}{dx^2} + k w_k + \left(\frac{EI}{\Phi}\right)\frac{d^2 q}{dx2} - q = 0 \tag{4.13}$$

$$-\frac{EIG}{c}\frac{d^6 \psi}{dx^6} + \left[EI\left(1 + \frac{k}{c}\right) + G\frac{EI}{\Phi}\right]\frac{d^4 \psi}{dx^4} - \left[G + k\frac{EI}{\Phi}\right]\frac{d^2 \psi}{dx^2} + k\psi - \left(1 + \frac{k}{c}\right)\frac{dq}{dx} + \frac{G}{c}\frac{d^3 q}{dx^3} = 0 \tag{4.14}$$

So, this means w_k, w and ψ all three consist of sixth-order differential equations ending up with six integration constants multiplied by the integrated differential equations. This means that the three deformations and the three nodal forces are as described in Equation 4.15.

$$w_k(x) = \sum_{i=1}^{6} C_i f_i \tag{4.15a}$$

$$\psi(x) = \sum_{i=1}^{6} C_i^{II} f_i \tag{4.15b}$$

$$w(x) = \sum_{i=1}^{6} C_i^{III} f_i \tag{4.15c}$$

$$M(x) = -EI\frac{d\psi}{dx} \tag{4.16a}$$

$$V(x) = -EI\frac{d^2\psi}{dx^2}\partial t \tag{4.16b}$$

$$V_G(x) = G\frac{dw_k(x)}{dx} \tag{4.16c}$$

where

G: Shear stiffness of the shear layer in $[kN/m^2]$
c: Compression stiffness of the upper layer in Kerr foundation $[kN/m]$
k_s: Compression stiffness of the lower layer in Kerr foundation $[kN/m]$
EI: Bending stiffness of the beam $[kNm^2]$
w_k: Deformation of the shear layer $[m]$
ψ: Rotation of the beam $[rad]$
w: Deformation of the beam $[m]$
$M(x)$: Bending moment of the beam $[kNm]$
V_G: Shear force of the shear layer $[kN]$
V: Shear force of the beam $[kN]$

which are defined by the soil characteristic as follows:

$$G_s = \frac{E_s}{2 * (1 - \nu_s)} \tag{4.17a}$$

$$G = \frac{4 * H_s * G_s * B}{9} \tag{4.17b}$$

$$k_s = \frac{4 * E_s * B}{H_s} \tag{4.18}$$

$$c = \frac{4 * E_s * B}{3 * H_s} \tag{4.19}$$

where

k_s: Compression stiffness of the lower layer in Kerr foundation $[kN/m]$
c: Compression stiffness of the upper layer in Kerr foundation $[kN/m]$
G_s: Shear stiffness of the soil $[kPa]$
E_s: Elastic modulus of the foundation $[kPa]$
B: Width of the tunnel element $[m]$
H_s: Thickness of the foundation $[m]$
ν_s: Poisson's ratio of the foundation soil layer $[-]$

Finite element model
A finite element model was built to define the relation between structure and soil. The outcome of this model is the deformation of the structure itself and the foundation layer beneath.

The first step in constructing the finite element model is the definition of the stiffness matrix, which is a matrix of 6 by 6 that contains the differential equations for deformation. The nodal displacements and force vectors are stated in equation 4.20 and 4.21, respectively. Equation 4.22 elaborates on the build-up of the stiffness matrix. The matrices R and Q are given in Appendix A.

$$u = \{w_1, \psi_1, w_{k1}, w_2, \psi_2, w_{k2}\} \text{ from (Morfidis 2007)} \tag{4.20}$$

$$s = \{V_1, M_1, V_{G1}, V_2, M_2, V_{G2}\} \text{ from (Morfidis 2007)} \tag{4.21}$$

where

$$c = R^{-1}u \text{ and } s = QR^{-1}u \text{ so } s = Ku \text{ and } K = QR^{-1} \tag{4.22}$$

The second step in constructing this model is in discretizing the uniform load into forces/moments that act on the nodes of the sub-elements. A visualization of this discretization is shown in figure 4.5 this discretization is based on the method and equations presented in (Morfidis 2007).

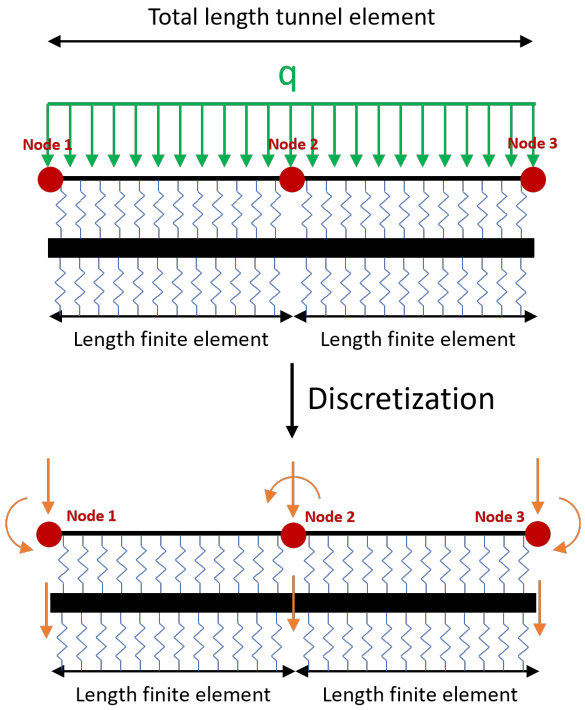


Figure 4.5: Visualization of discretization to nodal forces from uniform load

This discretization is done with the help of a transfer matrix of equations that are derived from the shape functions that come from the differential equation. The transfer matrix is shown in Appendix A. The equation used for the discretization is based upon the following equation 4.23.

$$y = F * y_0 + y_p \tag{4.23}$$

y is the vector containing the displacements and forces as follows:

$$y = y(x) = \{w(x) \;\; \psi(x) \;\; w_k(x) \;\; V(x) \;\; M(x) \;\; V_G(x)\} \tag{4.24}$$

and F is the transfer matrix that depends on the shape of f_i (shown in Appendix A) and is as follows:

$$F = F(x) = [f_{ij}(x)], \quad i, j = 1 - 6 \tag{4.25}$$

and y_0 and y_p are the vector of initial parameters and the vector that reflects the influence of the external forces, respectively. They are defined as follows:

$$y_0 = \{w_0 \;\; \psi_0 \;\; w_{k0} \;\; V_0 \;\; M_0 \;\; V_{G0}\} \tag{4.26}$$

and

$$y_p = y_p(x) = \{w_p(x) \;\; \psi_p(x) \;\; w_{kp}(x) \;\; V_p(x) \;\; M_p(x) \;\; V_{Gp}(x)\} \tag{4.27}$$

The influence vector for a uniform load is as depicted in Equation 4.28 and Figure 4.6 shows the direction and placement of forces. The equations within the matrices are all present in the appendix.

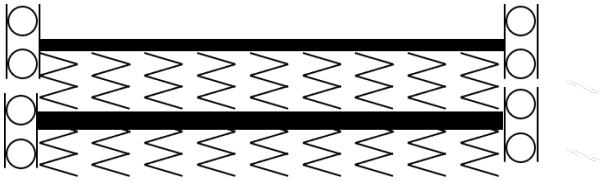


Figure 4.7: Element with free moving boundaries at the nodes

$$y_p^q(x) = \begin{cases} \{0\}, & x < aL \\ \int_{aL}^{x} q(u) f_{i4}(x-u)du (i = 1-6), & aL \le x \le bL, \\ & q(u) = q_a + \left[\frac{q_b - q_a}{(b-a)L}\right][u - (aL)] \\ \int_{aL}^{bL} q(u) f_{i4}(x-u)du (i = 1-6), & x > bL \end{cases} \tag{4.28}$$

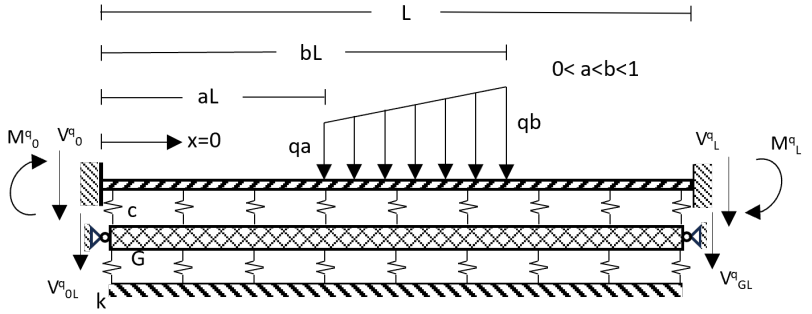


Figure 4.6: Visual representation forces in method of initial parameters from (Morfidis 2007)

In matrix form, equation 4.23 will look like shown in equation 4.29.

$$\begin{bmatrix} u_L \\ s_L \end{bmatrix} = \begin{bmatrix} F_{uu} & F_{us} \\ F_{su} & F_{ss} \end{bmatrix} \begin{bmatrix} u_0 \\ s_0 \end{bmatrix} + \begin{bmatrix} u_{pl} \\ s_{pl} \end{bmatrix} \tag{4.29}$$

The discretized forces and moments are calculated by introducing boundary conditions to the element (Morfidis 2007). The boundary conditions are

$u_0 = \{w_0, \psi_0, w_{k0}\} = 0$ and $u_L = \{w_L, \psi_L, w_{kL}\} = 0$

These boundary conditions together with equation 4.29 lead to the equations in Appendix A and provides the load vector with the loads at the nodes of the element.

These loads are then used as the external load on the beam and multiplied with the inverse of the stiffness matrix to end up with the deformations and rotation at the nodes. The boundary conditions used here are as shown in figure 4.7 with only the rotation fixed and the other directions free to move.

The result of this multiplication is the deformation and rotation (which is zero) at the nodes of the element.

The third step is to use a transfer matrix to develop deformations and forces over the entire length of the tunnel element. This is called the initial parameter method following (Selvadurai 1979). This method follows equation 4.23 which was described earlier. The difference from step one is that the influence of the distributed load is not accounted for now because it was discretized into forces on the nodes. The values calculated with the finite element method (deformations, rotations, shear forces, and moments)

at the beginning of the element are used here as the values of y_0 and multiplied with the transfer matrix that varies over x to get the deformation profile of the beam and the soil over the length of the element and beam.

4.4.2. Advantages and disadvantages of the TBKF model

Although the TBKF model offers improved accuracy for soil-structure interaction analysis, it also involves greater complexity and computational effort. This subsection discusses the strengths and limitations of the model, providing a balanced view of its applicability.

Advantages and disadvantages Timoshenko beam model

There are other examples of models that describe the deformation of a beam in the same matter as the Timoshenko method, for example, the Euler-Bernoulli beam (Haukaas 2023). However, this theory assumes that the beam is a slender beam with small deflections compared to the dimensions of the beam itself, which is not the case with an IMT, because this is not a slender structure, and more a rectangular structure with a distinguished width and height. Euler-Bernoulli also neglects shear deformation and axial deformation of the beam, which is important information to know for the immersed tunnel to prevent cracking. The Timoshenko beam theory does take these into account and is more accurate for shorter, thicker beams that are subjected to high shear loads. This is more fitting for an IMT element.

The most important advantages of the Timoshenko beam model are the following.

- Accuracy for short and thick beams: It accounts for shear deformation and rotational inertia, which makes it more accurate for short and thick beams compared to the Euler-Bernoulli method.
- Dynamic analysis: It includes rotational inertia, which is needed for dynamic analysis.
- Versatility: It is suitable for various types of beam, including composite and sandwich beams.

However, there are some disadvantages to this method. These disadvantages are as follows.

- Complexity: Involves higher-order derivatives and additional beam parameters.
- Computational effort: Takes up more computational power for the solving of the equation.

The computational effort and complexity will require much attention in regard to the time it will take to construct the model and execute it. Fortunately, there are no specific technical disadvantages that could cause errors in the final result.

Advantages and disadvantages of the Kerr foundation model

All of the foundation models mentioned in the introduction of section 4.4.1 (Winkler, Pasternak, and Vlasov) use a different translation of the subsoil to a spring system. The Kerr foundation system uses three parameters to define the foundation layer, while the Winkler foundation only uses one. An overview of the systems is shown in figure 4.8. Here it is shown that the Winkler model only uses one spring stiffness, the Pasternak model is a simplification of the Kerr model with only one spring stiffness instead of two, and the Vlasov model is also a simplification of the Kerr model, where the spring stiffness's are disregarded completely and the foundation consists of a single soil layer with subgrade and shear stiffness.

Furthermore, the performance of every model described in figure 4.8 is tested with respect to a reference 2D-finite solution. The results of displacements, bending moments, and stresses are shown in Figures 4.9, 4.10 and 4.11, respectively, clearly indicating that the performance of the Kerr model is the best in both cases.

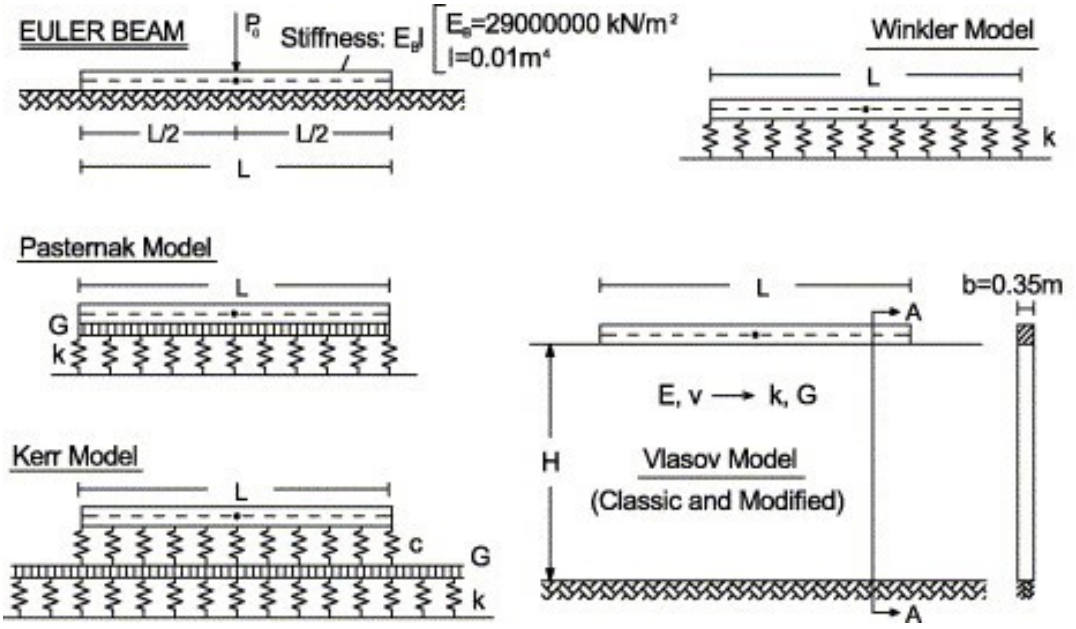


Figure 4.8: Comparison foundation models from (Avramidis and Morfidis 2006)

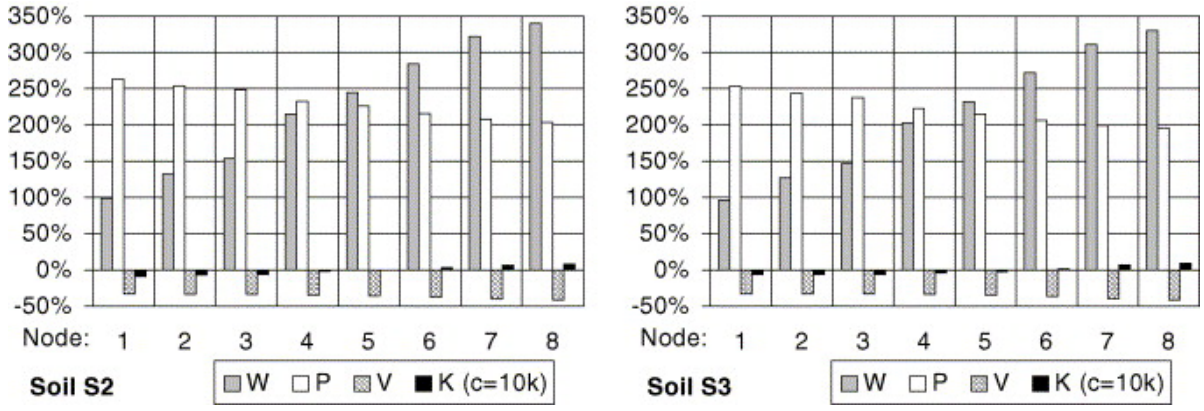


Figure 4.9: Deviations of deformations in the models in comparison with 2D-finite element solution from (Morfidis 2007)

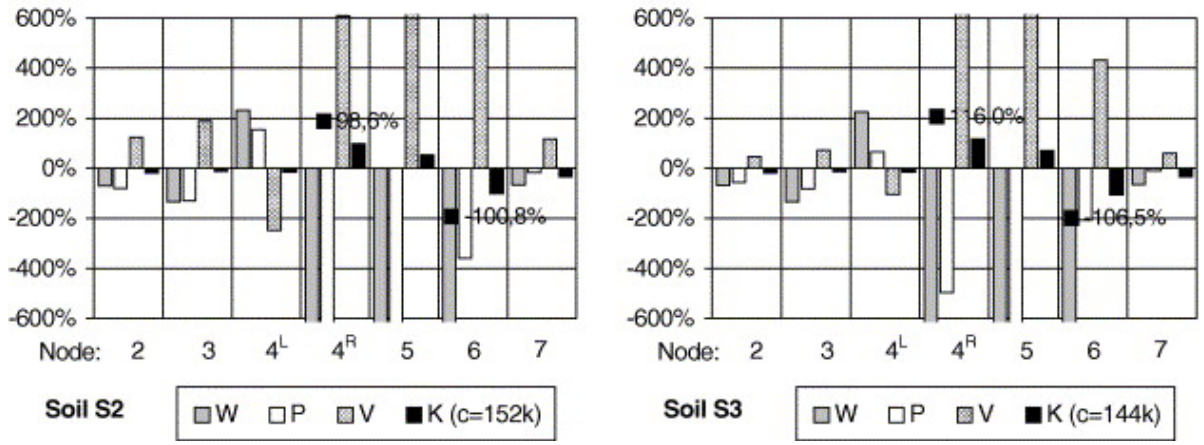


Figure 4.10: Deviations of bending moments in the models in comparison with 2D-finite element solution from (Morfidis 2007)

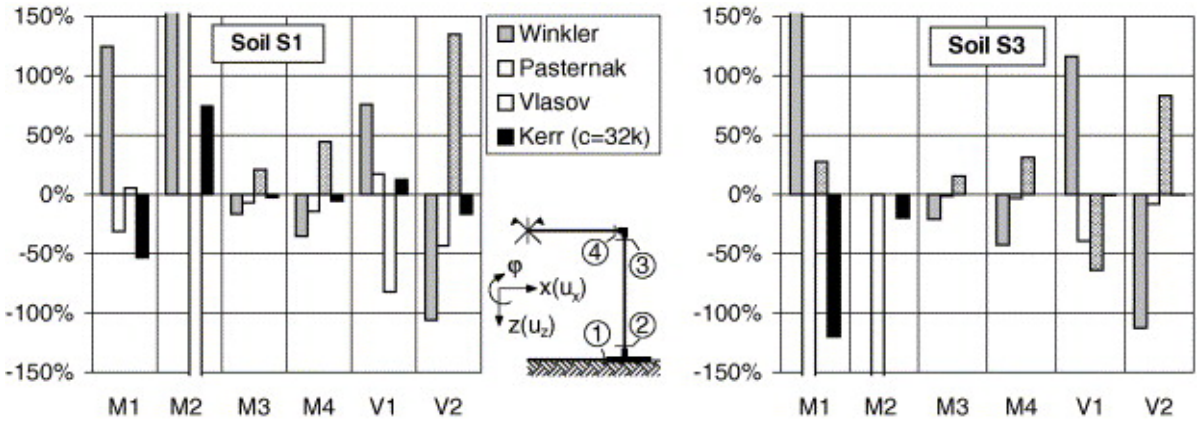


Figure 4.11: Deviations of stresses in the models in comparison with 2D-finite element solution from (Morfidis 2007)

Morfidis (2007) states that the formulation of the stiffness and transfer matrices for a Timoshenko beam on a Kerr foundation is relatively easy and simple to implement into a model, so no extra effort will be used when choosing the Kerr foundation. Morfidis (2007) also states that the Kerr foundation is superior to the other models mentioned and concludes with the following:

- The overall performance of the Kerr model is excellent and most notable in soil S1 (hard clay) and S2 (loose sand), with the exception of the bending moment of soil S3 (dense sand) in figure 4.10 where it exceeds the 100 % deviation. Unfortunately, the other models do not perform better here.
- The Pasternak model is also extremely efficient in general, with the exception of the bending moment M_2. In the bigger picture its efficiency is still somewhat lower than the Kerr foundation model
- The least precise models are the Winkler and Vlasov model with all divergences of stresses.

The reasons for making use of the Kerr foundation model are as follows.

- Detailed Soil-Structure interaction: Provides a comprehensive representation of soil-structure interaction with its three parameters.
- Flexibility: Can be adopted to various foundation conditions.
- Improved accuracy: Offers better accuracy in predicting deflections and stresses in beams on elastic foundations.

The downsides of the Kerr foundation model are as follows.

- Complexity: It is mathematically complex and requires detailed parameter estimation.
- Computational effort: Just as with the Timoshenko beam model, it takes up a lot more computational power to solve the equations.
- Sensitivity of parameters: The accuracy of the model heavily depends on the precise estimation of its parameters, which can be challenging.

These are no disadvantages that are impossible to overcome, if they are taken into account correctly. The computational effort will be covered by adjusting the code to machine code, so that it runs faster and saves its calculations. The complexity and sensitivity costs more time because they need to be covered carefully, and here a more precise way of determination is necessary.

4.5. Model to define the primary consolidation

Primary consolidation involves the dissipation of excess pore water pressure, leading to a reduction in soil volume. This section introduces the Conte and Troncone method, which provides an analytical solution for one-dimensional consolidation under time-dependent loading. It explains the technical aspects of the method and its advantages over other approaches.

All current models used to define primary consolidation are based on Terzaghi's method to determine primary consolidation (Terzaghi, Peck, and Mesri 1996). This consolidation settlement is defined by Terzaghi as depicted in Equation 4.30 (Das 2016). The stresses in a clay layer underneath a foundation are visualized in Figure 4.12.

$$S_{c(p)} = \int \varepsilon_z dz \quad \text{(Das 2016)} \qquad (4.30)$$

where

ε_z: Vertical strain = $\frac{\Delta e}{1+e_0}$ [-]
Δe: Change of void ratio [-]

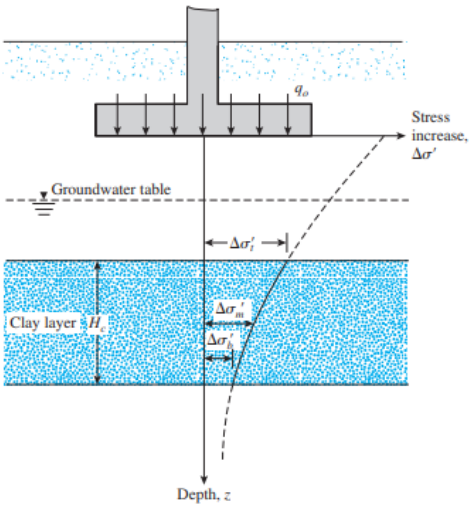


Figure 4.12: Visualization of the stress distribution in the soil column underneath a foundation (Das 2016)

Equation 4.30 produces different approaches for clayley soils consolidated differently. The equations for the three types of consolidation are shown in equation 4.31, 4.32 and 4.33.

For normally consolidated cohesive soils:

$$S_{c(p)} = \frac{C_c H_c}{1+e_0} \log \frac{\sigma_0' + \Delta\sigma_{av}'}{\sigma_0'} \quad \text{(Das 2016)} \qquad (4.31)$$

For overconsolidated cohesive soils with $\sigma_0' + \Delta\sigma_{av}' < \sigma_c'$

$$S_{c(p)} = \frac{C_s H_c}{1+e_0} \log \frac{\sigma_0' + \Delta\sigma_{av}'}{\sigma_0'} \quad \text{(Das 2016)} \qquad (4.32)$$

For overconsolidated cohesivel soils that fall in the range: $\sigma_0' < \sigma_c' < \sigma_0' + \Delta\sigma_{av}'$

$$S_{c(p)} = \frac{C_s H_c}{1+e_0} \log \frac{\sigma_c'}{\sigma_0'} + \frac{C_c H_c}{1+e_0} \log \frac{\sigma_0' + \Delta\sigma_{av}'}{\sigma_0'} \quad \text{(Das 2016)} \qquad (4.33)$$

where

σ_0': Average effective pressure on the clay layer before the construction of the foundation $[kPa]$
$\Delta\sigma_{av}'$: Average increase in effective pressure on the clay layer caused by the construction of the foundation $[kPa]$
σ_c': Preconsolidation pressure $[kPa]$
e_0: Initial void ratio of the clay layer $[-]$
C_c: Compression index $[-]$
C_s: Swelling index $[-]$
H_c: Thickness of the clay layer $[m]$

The time for completion of the primary consolidation settlement follows from the calculation of the primary consolidation settlement (Equation 4.31) and the degree of consolidation defined as shown in equation 4.34. The time that complete consolidation took will follow from the non-dimensional time factor represented in Equation 4.35.

$$U = \frac{S_{c(t)}}{S_{c(max)}} = 1 - \sum_{m=0}^{m=\infty} \left(\frac{2}{M^2} \right) e^{-M^2 T_v} \quad \text{(Das 2016)} \tag{4.34}$$

with

$$\begin{aligned} T_v &= \frac{C_v t}{H_d^2} \quad \text{(Das 2016)} \\ C_v &= \frac{k}{m_v \gamma_w} = \frac{k}{\frac{\Delta e}{\Delta \sigma' (1+e_{av})} \gamma_w} \quad \text{(Das 2016)} \end{aligned} \tag{4.35}$$

where

T_v: Non-dimensional time factor [−]
H_d: Length of the maximum drainage path [m]
M: [(2m+1)π]/2 [−]
m: An integer = 1,2,... [−]
t: Time [s]
C_v: Coefficient of consolidation [m^2/s]
k: Hydraulic conductivity [m/s]
Δe: Total change of void ratio caused by an effective stress increase of $\Delta \sigma'$ [−]
e_{av}: Average void ratio during consolidation [−]
m_v: Volume coefficient of compressibility = $\frac{a_v}{1+e_{av}}$=$\Delta e / [\Delta \sigma' (1 + e_{av})]$
a_v: $\frac{\Delta e}{\Delta \sigma'}$

The approach of Conte and Troncone

The approach of Conte and Troncone (2006) is also based on the theory of Terzaghi (1996). This research presented an analytical solution for a column of saturated soil layers subjected to a time-dependent load. It uses a Fourier series to define the loading. This method fits the purpose of this research very well, because it both involves incremental loading, one-dimensional consolidation, and a multi-layered system.

4.5.1. Technical aspects of Conte and Troncone

The Conte and Troncone method uses Fourier series to define loading and provides solutions for multi-layered soil systems. This subsection details the governing equations, boundary conditions, and the application of Duhamel's theorem in obtaining consolidation results.

The equation governing the one-dimensional consolidation in saturated soils is derived from the mass conservation equation for porous media (Conte and Troncone 2006) and shown in Equation 4.36.

$$\nu \beta \frac{\partial u}{\partial t} - \frac{\partial \epsilon_z}{\partial t} + \frac{\partial v_z}{\partial z} = 0 \tag{4.36}$$

where

z: The spatial coordinate $[m]$
t: Time $[s]$
u: Excess pore-water pressure depending on both z and t $[kPa]$
ε_z: Vertical strain of the soil $[m]$
v_z: Rate of water flow across a unit area of soil in the z direction $[m^2/s]$
v: Soil porosity $[-]$
β: Pore-water compressibility $[-]$

Conte and Troncone made the assumptions that the soil behaves as an isotropic soil (behaves the same in all directions), is linearly elastic, follows Darcy's law with constant coefficient of permeability for the water flow, and creep and inertial effects are ignored. These assumptions will be covered in Section 4.5.2 in further detail. With these assumptions, the governing equation simplifies to Equation 4.37.

$$c_v \frac{\partial^2 u}{\partial z^2} = \frac{1}{\eta} \frac{\partial u}{\partial t} - \frac{d\sigma}{dt} \tag{4.37}$$

where

$c_v = \frac{k_w}{m_v \gamma_w}$: The coefficient of consolidation $[m^2/s]$
$\eta = \frac{m_v + n\beta}{m_v}$: Parameter accounting for compressibility of the soil and pore fluid $[-]$
σ: The total vertical stress$[kPa]$
k_w: The coefficient of permeability $[m/s]$
m_v: The coefficient of volume change $[m^3/s]$
γ_w: The unit weight of water $[kN/m^3]$

The load applied to the soil will be defined as a harmonic motion that varies over time (Equation 4.38). If this equation is differentiated by time, one will end up with an equation for the loading rate, shown in equation 4.39. For general time-dependent loading it can be expanded in harmonic components with a Fourier series as shown in equation 4.40

$$\sigma(t) = Acos(\omega t) + Bsin(\omega t) \tag{4.38}$$

$$\frac{d\sigma}{dt} = -A\omega sin(\omega t) + B\omega cos(\omega t) \tag{4.39}$$

$$\sigma(t) = A_0 + \sum_{k=1}^{inf} [A_k cos(\omega_k t) + B_k sin(\omega_k t)] \tag{4.40}$$

where

A and B: Load amplitudes $[-]$
ω: Circular frequency $[rad/s]$
A_k= $\frac{2}{T} \int_0^T \sigma(t) cos(\omega_k t) dt$ $[-]$
B_k= $\frac{2}{T} \int_0^T \sigma(t) sin(\omega_k t) dt$ $[-]$
w_k= $\frac{2k\pi}{T}$ $[1/s]$
T: Period of the load $[s]$

The solution to the governing equation is then obtained with the given boundary conditions 4.41 and Duhamel's theorem (Conte and Troncone 2006) and shown in equation 4.42

$$\frac{\partial u}{\partial z}(o,t) = 0 \qquad \text{Impervious base} \tag{4.41a}$$

$$u(H,t) = 0 \qquad \text{Fully permeable upper surface} \tag{4.41b}$$

$$u(z,0) = 0 \qquad \text{Initial condition} \tag{4.41c}$$

$$u(z,t) = \int_0^t \frac{\partial \sigma(\tau)}{\partial \tau} \bar{u}(z, t-\tau) d\tau \tag{4.42}$$

where $\bar{u}(z, t-\tau)$ is the solution when the loading rate is kept at unity.

The accumulation of excess pore pressure is calculated with equation 4.43 and the accumulation of consolidation is calculated with equation 4.44

$$u_\omega(z,t) = \sum_{j=1}^{\inf} \left[Y_j cos\left(\frac{(2j-1)\pi z}{2H}\right) exp\left(-\eta\left(\frac{(2j-1)\pi}{2H}\right)^2 T_v\right)\right] \tag{4.43}$$

where

$Y_j =$ $(A + B\vartheta\chi^2)[\cos(\omega t) - \exp(-\eta\chi^2 T_v)] - (A\vartheta\chi^2 - B)\sin(\omega t)$
$\chi =$ $\frac{(2j-1)\pi}{2}$
$\vartheta =$ $\frac{\eta c_v^2}{\omega H^2}$
$T_v =$ $\frac{c_v t}{H^2}$.

$$s_\omega(t) = m_v H \left[Acos(\omega t) + Bsin(\omega t) - \sum_{j=1}^{\inf} \frac{2}{\eta\chi^2} Y_j \right] \tag{4.44}$$

The actual excess pore-water pressure and deformation are given by superimposing their accumulated parts mentioned in equation 4.43 and 4.44. This results in equation 4.45 for the excess pore pressure and equation 4.46 for the deformation.

$$u(z,t) = \sum_{k=1}^{M} u_k(z,t) \tag{4.45}$$

$$s(t) = m_v H \left[\sigma_0 + \sum_{k=1}^{M} s_k(t) \right] \tag{4.46}$$

where

$u_k(z,t)$: Pore-water pressure at each harmonic component $[kPa]$
$s_k(t)$: Deformation due to each harmonic component $[m]$
σ_0: Average load over the period $[kPa]$

4.5.2. Advantages and disadvantages of Conte and Troncone

The Conte and Troncone method is versatile and practical, but it relies on specific assumptions about soil behavior and boundary conditions. This subsection discusses the strengths and limitations of the method, emphasizing its suitability for the Fehmarnbelt Tunnel project.

The disadvantages are as follows.

- The method relies on the assumption that the soil behaves isotropic, is linearly elastic, and ignores creep.
- Its complexity will increase for non-periodic or irregular loading conditions.
- The accuracy of the method relies on the correct estimation of soil parameters.
- The boundary conditions (impervious base and fully permeable upper surface) in the method are very specific.

The second and third disadvantages can be easily solved with more time invested in the method and just need to be taken into account. The first and fourth disadvantages could pose a bigger threat to the results. Isotropic or anisotropic does not play a role in this thesis, the soils are highly overconsolidated, so will behave linearly elastic under the (relatively) small load applied and creep will be added manually, as will be explained in Section 4.6. The boundary condition involves the placement of permeable layer boundaries, which has an effect of the drainage path (H_d).

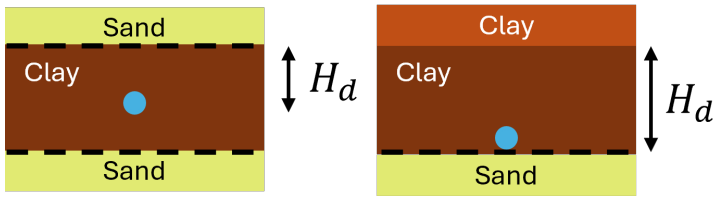


Figure 4.13: Explanation drainage path soil layer

As shown in Figure 4.13, the drainage path is the longest distance a water particle has to travel to leave the soil layer (Das 2016). If a cohesive layer is enclosed by another cohesive layer on top, this will count as an impervious boundary. If a permeable layer or a water body is present, this is a permeable boundary. The boundary condition problem can be tackled by, for example, subdividing the soil layer into two and inverting one of those parts; one could model a layer with two permeable boundaries then. This trick is shown in Figure 4.14.

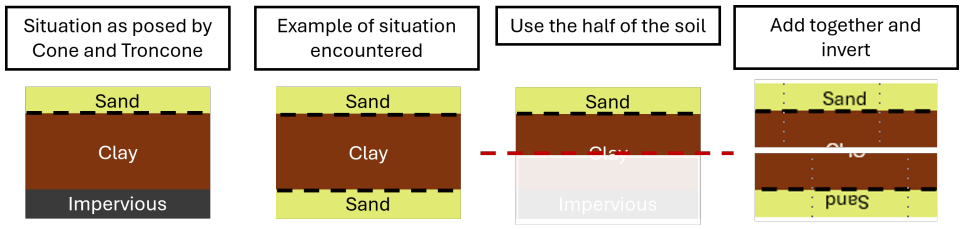


Figure 4.14: Solution for the boundary problem

The advantages of this method are its versatility, practicality, and accuracy. It is versatile because it can handle single and cyclic loads. Its practicality comes from the fact that it is perfect for one-dimensional consolidation for a multilayered soil column. Its accuracy was validated against the experimental results, concluding in a good accuracy of the method.

Other methods considered were Olson's method (1979), Schiffman and Stein's method (1970), Zhu and Yin's method (1999), Hsu and Lu's method (2006), Gibson's method (1958), and Lee and Sills' method (1981).

The latter three and Schiffman and Steins' method provide a more complex determination for the loading condition or soil parameters, so Conte and Troncone's method (2006) offers a simpler procedure that does not require numerical approaches. Furthermore, Conte and Troncone's method is more versatile than Zhu and Yin's method and Olsen's method, because it can handle different types of loading and is broader applicable in terms of amount of soil layers present in the soil column.

4.6. Model to define the secondary consolidation

Secondary consolidation, or creep, occurs over time under sustained load. This section introduces the Feng et al. method (2020), which is based on Hypothesis B (explained in section 2.1.7) for 1-D consolidation settlement of clayey soils under multistage ramp loading and the concept of equivalent time by Bjerrum (1967). It explains the technical aspects of the method and its application in the prediction of long-term soil deformation.

The equivalent time concept ($t_{ej,k}$) is a theoretical construct that allows engineers to model the deformation (creep) dependent on the time of the soils by transforming the actual time into an equivalent time that reflects the combined effects of primary and secondary consolidation. This concept is particularly useful in the context of Hypothesis B, which assumes that creep occurs simultaneously with primary consolidation (Bjerrum 1967).

4.6.1. Technical aspects of Feng et al.

The Feng et al. method calculates total consolidation by combining primary and secondary consolidation phases. This subsection details the equations governing creep strain and delayed creep strain, highlighting the importance of equivalent time in soil deformation analysis.

The total consolidation in Feng et al. (2020) is calculated with:

$$S_{totalB} = \sum_{j=1}^{m} (S_{consolidation,j} + S_{creep,j}) \tag{4.47}$$

where

S_{totalB}:	The total consolidation based on Hypothesis B $[m]$
$S_{consolidation,j}$:	The consolidation under j-th loading $[m]$
$S_{creep,j}$:	Creep consolidation under the j-th loading $[m]$
j:	The loading step numbering $[-]$

The $S_{consolidation,j}$ in this equation will follow from the Conte and Troncone approach (Conte and Troncone 2006). The $S_{creep,j}$ is the result needed from the approach of Feng et al. and is defined as shown in Equation 4.48.

$$S_{creep,j} = \sum_{k=1}^{n} (S_{creep,fj,k} + S_{creep,dj,k}) \tag{4.48}$$

where

$S_{creep,fj,k}$:	The creep consolidation with respect to the final j-th effective stress $[m]$
$S_{creep,dj,k}$:	The delayed creep consolidation due to the coupling of the excess pore water pressure $[m]$
k:	The layer numbering $[-]$

The creep consolidation and delayed creep consolidation are composed of a sum of the respective creep strains (ε_{creep}) times the height of the layer (h_k), as can be seen in the equations 4.49 and 4.50.

$$S_{creep,fj} = \sum_{k=1}^{n} (\varepsilon_{creep,fj,k}) * h_k \tag{4.49}$$

$$S_{creep,dj} = \sum_{k=1}^{n} (\varepsilon_{creep,dj,k}) * h_k \tag{4.50}$$

For an over-consolidated soil layer, the creep strain rate is defined as follows:

$$\varepsilon_{creep,fj,k} = \frac{C_{\alpha e}}{(1+e_0)} log\left(\frac{t_0 + t_{ej,k}}{t_0 + \Delta t_{ej,k}}\right) \tag{4.51}$$

where

$$\Delta t_{ej,k} = t_0 * 10^{\left((\varepsilon_{ej,k} - \varepsilon_{zp(j-1),k})\frac{(1+e_0)}{C_{\alpha e}}\right)} \left(\frac{\sigma'_{zj,k}}{\sigma'_{zp(j-1),k}}\right)^{-\frac{C_c}{C_{\alpha e}}} - t_0 \tag{4.52}$$

and $t_{ej,k}$ as seen in Equation 4.52 is defined as:

$$t_{ej,k} = t - \sum_{j=1}^{j-1} t_j - \frac{t_{cj}}{2} - t_0 + \Delta t_{ej,k} \tag{4.53}$$

For a normally consolidated soil layer, the following holds:

$$\varepsilon_{creep,dj,k} = \frac{C_{\alpha e}}{(1+e_0)} log\left(\frac{t_0 + t_{ej,k}}{t_0}\right) \tag{4.54}$$

and $t_{ej,k}$ as seen in Equation 4.55 is defined as:

$$t_{ej,k} = t - \sum_{j=1}^{j-1} t_j - \frac{t_{cj}}{2} - t_0 \tag{4.55}$$

The delayed creep strain is, as it says in the name, delayed by a time factor. The delay is defined by t_{EOP}. This t_{EOP} is the time at which primary consolidation reached its end, which is when all excess pore pressures are dissipated. The end of primary consolidation follows from the approach of Conte and Troncone (2006).

For an overconsolidated soil layer, the delayed creep strain is defined in Equation 4.56.

$$\varepsilon_{creep,dj,k} = \frac{C_{\alpha e}}{(1+e_0)} * log\left(\frac{t_0 + t_{ej,k}}{\Delta t_{ej,k} + t_{EOP}}\right) \tag{4.56}$$

Similarly for a normally consolidated state:

$$\varepsilon_{creep,dj,k} = \frac{C_{\alpha e}}{(1+e_0)} * log\left(\frac{t_0 + t_{ej,k}}{t_{EOP}}\right) \tag{4.57}$$

The parameters in this section are as follows.

$C_{\alpha e}$: The secondary compression index [−]
C_c: The primary compression index [−]
t_0: Creep parameter in units of time (1 day in this research (Wei-Qiang Feng et al. 2020)) [day]
e_0: Initial void ratio [−]
t_{cj}: Construction period of load [$days$]
t_j: Duration up until next load or till the end of consolidation [$days$]
j: Loading numbering [−]
k: The layer numbering [−]

The stress state of the soil
As shown in Equations 4.52 and 4.55 the difference in equivalent time depends on the current stress state and the stress state in the previous loading step. The $\varepsilon_{zp(j-1),k}$ and $\sigma'_{zp(j-1),k}$ represent the overconsolidation pressure and strain of the soil layer before the load step was applied to the soil column. A visualization of the creep strain is shown in Figure 4.15. In this figure, the numbers represent the loading steps. In the first loading step, it is still in its overconsolidated state (from point 0 to point 1).

In its second loading step, it is partially loaded in its overconsolidated state and partly in its normally consolidated state. The preconsolidation pressure and deformation increase with each loading step, as can be seen from the endpoints of the blue lines. The growth of the preconsolidation pressure is expressed as shown in Equation 4.58 (W.-Q. Feng and J.-H. Yin 2017).

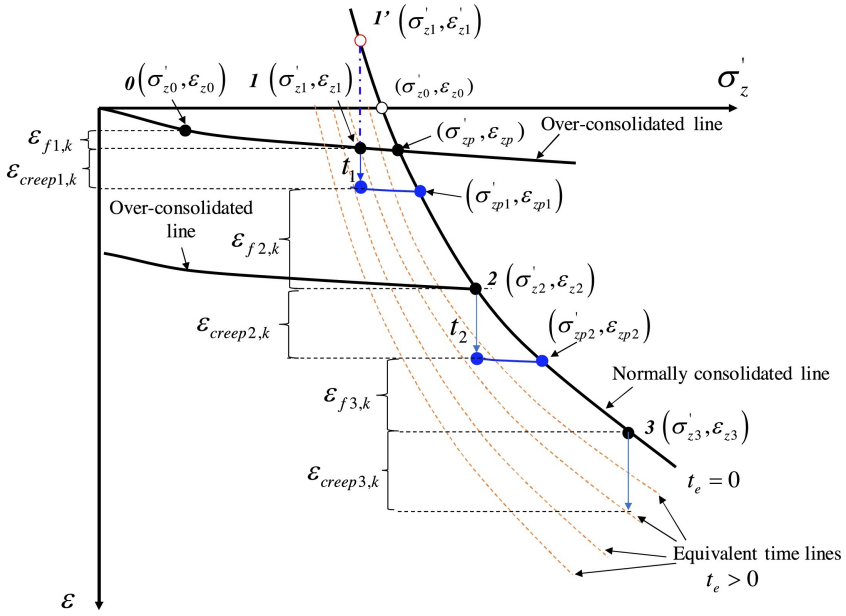


Figure 4.15: The relationship between vertical strain versus vertical effective stress with different time lines under various stress–strain states from (Wei-Qiang Feng et al. 2020)

$$\sigma'_{zp(j-1),k} = 10^{\left(\varepsilon_{z(j-2),t1,k}-\varepsilon_{zp(j-2),k}\right)*\frac{(1+e_0)}{(C_c-C_e)}} * \left(\sigma'_{z(j-1),k}\right)^{-\frac{C_e}{(C_c-C_e)}} * \sigma'_{zp(j-2),k}\frac{C_c}{(C_c-C_e)} \tag{4.58}$$

where

$C_{\alpha e}$: The secondary compression index [−]
C_c: The primary compression index [−]
C_e: The swelling index [−]
e_0: Initial void ratio [−]
j: Loading numbering [−]
k: The layer numbering [−]

4.6.2. Advantages and disadvantages of Feng et al.

The Feng et al. method offers simplicity and versatility, but requires precise parameter estimation and assumes linear elastic behavior. This subsection discusses the strengths and limitations of the method, providing insight into its applicability to the Fehmarnbelt Tunnel project.

There are other examples of models that describe the secondary consolidation process. Examples found are Olson's method (1979), Schiffman and Stein's method (1970), Zhu and Yin's method (1999), Hsu and Lu's method (2006), Gibson's method (1958), and Lee and Sills' method (1981). The same models are considered here as for the primary consolidation.

The latter three and the method of Schiffman and Stein provide a more complex determination of the loading condition or soil parameters, so Feng et al. (2020) offer a more simple method to determine secondary consolidation. Hypothesis B as a basis for the method is only present in the method of Zhu and Yin (1999) and Feng et al. (2020). However, the downside of Zhu and Yin is that it is only applicable to double-layered soil profiles, so it is not a good fit for the cause of this thesis.

The method proposed by Feng et al. (2020) stands out for its simplicity, versatility, and practical applicability, making it a valuable tool for engineers dealing with one-dimensional consolidation under multistage ramp loading. Other methods offer various approaches to handle different aspects of time-dependent loading and soil behavior, but do not offer the specific approaches needed in this case.

4.7. Coupling of tunnel elements per zone

To manage the complexity of soil analysis, tunnel elements are coupled together in longitudinal direction. This section explains the methodology for coupling elements, including the use of fixed supports and the impact of soil-structure interaction on overall tunnel stability.

This sectional analysis will be converted to a longitudinal analysis by coupling the elements together in longitudinal direction with the Timoshenko beam on the Kerr foundation as basis. The elements will be coupled together as visualized in Figure 4.16. The endpoints of each element will be modeled with roller supports, which means that the vertical movement will be unknown and the rotations will be zero. The transition between a regular to a special element is gradual in the real case, but modeled here as a discrete transition to make the coupling and modelling more simple.

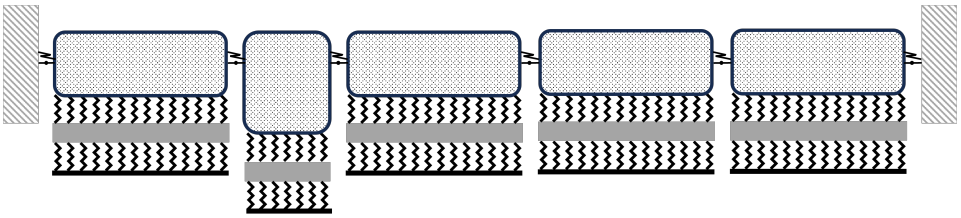


Figure 4.16: Schematic overview of a part of the Fehmarnbelt tunnel on longitudinal axis

4.8. Monte Carlo simulation

The Monte Carlo method is used to assess the variability in soil layer thicknesses and their impact on consolidation. This section provides an overview of the simulation process, including random sampling, generation of results, and analysis of variability in the prediction of soil deformation.

The Monte Carlo method is a technique for modeling the probability of different outcomes generated by an input of random generated variables and is used to understand the propagation of risk and uncertainty from input to output (Raychaudhuri 2008). It involves running a large number of simulated data to generate a range of possible outcomes based on random sampling of input variables. It works in the following way.

1. It starts by random sampling a range of values for the input variables, these values follow a given distribution. A doubly truncated lognormal distribution in this case.
2. These values are used as inputs in the model. This generates a range of outcomes.

3. The chosen distribution for the input values will then generate a distribution of output values, which can be analyzed.

Figure 4.17 provides a visual representation of the mathematical process of a Monte Carlo approach.

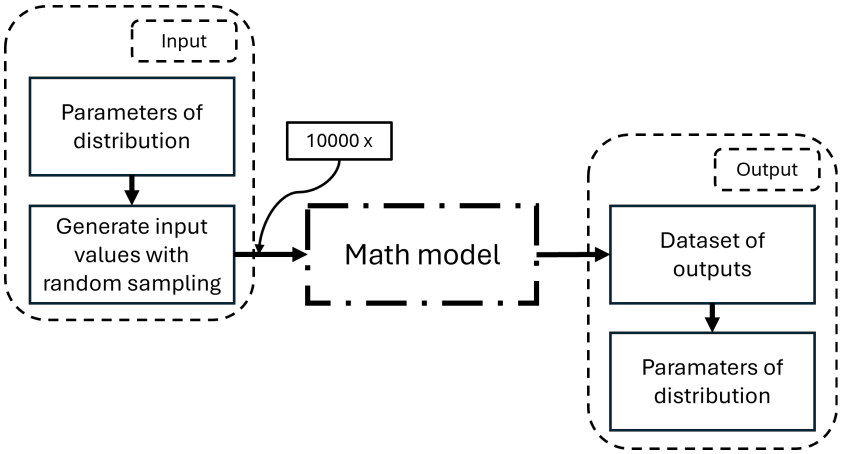


Figure 4.17: Process of Monte Carlo Simulation

As stated in 2.3.2 the lognormal distribution is the recommended distribution to capture the variation in the thicknesses of the soil layers. The advantage of this distribution is that it will not create negative values or zero values, due to its natural logarithmic nature. However, a truncation is still necessary so it will not create very thin layers that could create outliers in the model due to their very low drainage path, etc. So, the thickness of soil layers will be bounded between 0.1 meters in thickness and the remaining height there is left in the soil column. This upper boundary is added so that all the generated layer thicknesses fit into the soil column of their respective zone. This will mean that the effect of that upper boundary is only relevant for the last layer in the soil column because it has the thickness that is left in the soil column.

In conclusion, the following steps are taken in this research regarding the Monte Carlo simulation.

1. Per zone, the soil column is generated with random thicknesses for the soil layers present.
2. The mean used for this random generation are the values from the literature (Yumpu.com 2011).
3. The standard deviations used are 0.5 meter, 1 meters, 2 meters and 5 meters.
4. The random generation of thicknesses are log-normally distributed with truncations at 0.1 meter in thickness and the remaining thickness in the soil column.
5. 500 soil column configurations are generated for the complete analysis.
6. The calculation model is run for every soil column generation and the mean, standard deviation, 5th percentile, and 95th percentile are taken from all the results of these runs together to give an indication on the distribution and error of the results.

These steps lead to a doubly truncated lognormal distribution as shown in figure 4.18. These randomly generated thicknesses have a standard deviation of 0.5 meter and 500 simulations per soil layer were made. Especially in soil layer 1, the truncated lognormal distribution is clearly visible in the probability density function line, because it is cutoff at 0.1 meters of thickness. The rest of the soil layers have a mean that is further from the lower bound of the generated soil layer thicknesses.

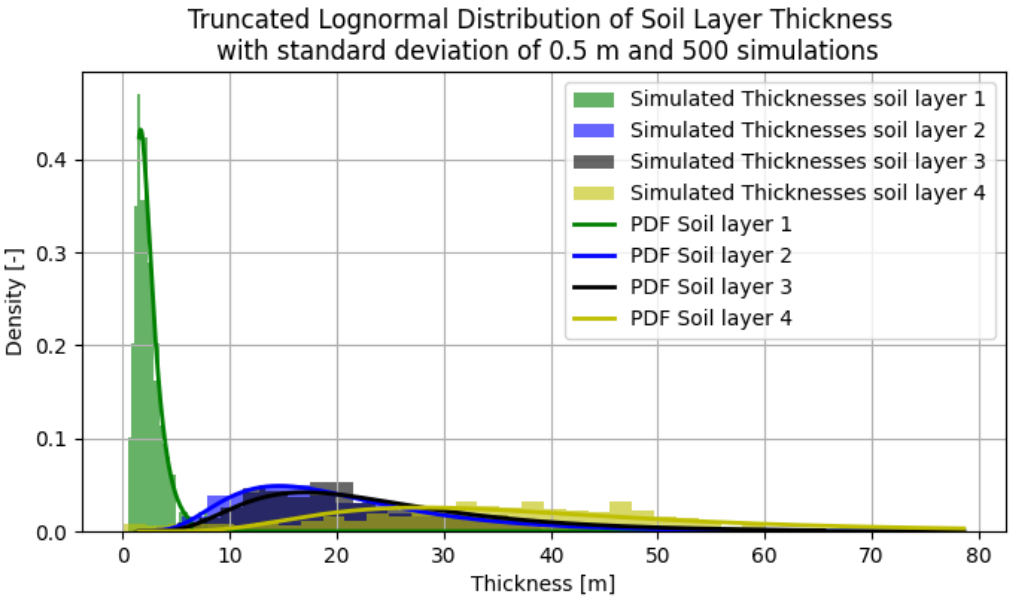


Figure 4.18: Doubly truncated lognormal distribution of soil layer thicknesses with a standard deviation of 0.5 meter and 500 simulations per soil layer

The upper boundary of the randomly generated input values with the doubly truncated lognormal distribution is clearly visible in the layer 3 distribution shown in figure 4.19.

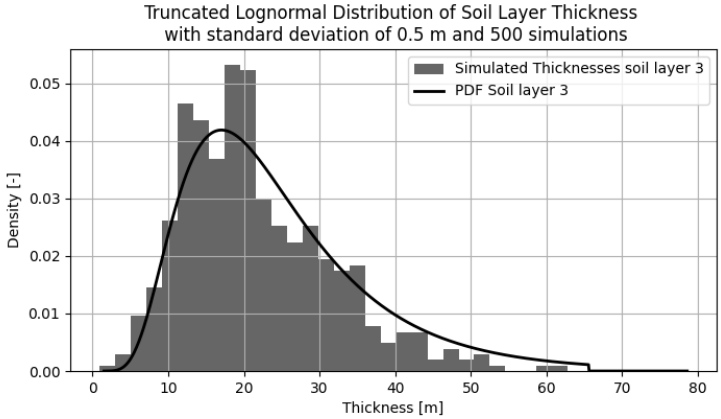


Figure 4.19: Example doubly truncated lognormal distribution of soil layer thickness soil layer 3

5

Results of the analytical models for one simulation

This chapter presents the results obtained from a single simulation, focusing on the thickness of the soil layer in different zones. It compares the performance of regular and special elements, highlighting the differences in their responses. In addition, it provides an overview and discussion of the behavior of all soil zones (introduced in Chapter 3) along with their corresponding soil layers.

The mean thicknesses of the soil layers per zone are given in Figure 5.1 and used for these results. The results are collected and ordered as shown in Figure 5.2 It can be seen from this figure that the results are subdivided into 3 categories, namely *initial, primary, and secondary deformation of separate soil layers in zone*, *initial, primary, and secondary deformation of one element in zone* and *initial, primary, and secondary deformation of tunnel in zone*. All of these categories also have their respective range in results, which will be shown in Chapter 6.

Zone A		Zone B		Zone C		Zone D		Zone E	
Soil layer	Thickness [m]	Soil layer	Thickness [m]	Soil layer	Thickness [m]	Soil layer	Thickness [m]	Soil layer	Thickness [m]
Water level	9.625	Water level	34.625	Water level	34.375	Water level	21.875	Water level	12.5
Palaeogene clay	80.475	Basin deposits[1]	3.6625	Upper till	0.625	Upper till	27.125	Meltwater sand[2]	2.152
		Upper till	14.0625	Lower till	12.5	Lower till	28.0625	Upper till	18.75
		Meltwater sand	6.5	Chalk	43.6	Palaeogene clay	14.0375	Lower till	21.875
		Lower till	32.25					Palaeogene clay	35.85

Figure 5.1: Mean layer thickness per zone

[1] The basin deposit layer disappears with a special element, because it will be removed to make the trench deep enough for the element.
[2] The meltwater sand layer disappears with a special element because it will be removed to make the trench deep enough for the element.

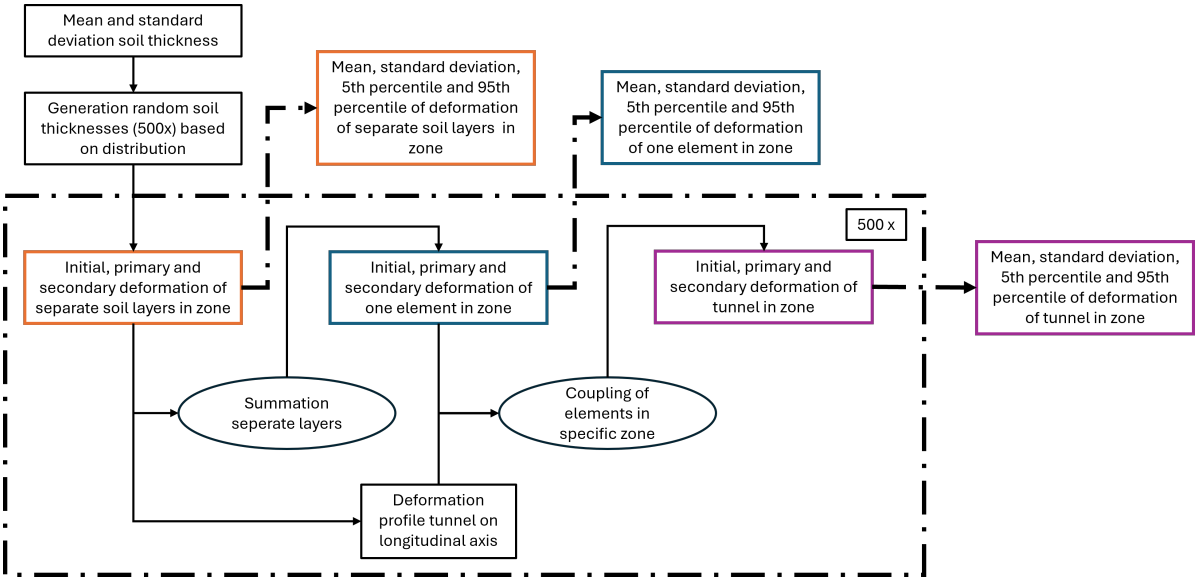


Figure 5.2: The overview of results and variability of the complete model

5.1. The result of the TBKF model for one simulation

The Timoshenko beam on the Kerr foundation model (TBKF model) is used to analyze the initial deformation of the soil-structure interaction. This section presents the results of the TBKF model for one simulation, focusing on the differences in deformation between regular and special tunnel elements in various zones. It discusses the factors that contribute to heave and settlement, providing a comprehensive understanding of the initial response of the soil to load.

The results of the Timoshenko beam Kerr foundation model of Morfidis (2007) are shown in Figure 5.3. In the figure, several interesting observations can be made. The soil beneath the special element reacts stiffer to the load applied. This substantial difference can be attributed to the length of the element, which is much shorter and therefore more rigid, as well as to the weight of the element, which is lower. Additionally, the negative sign on the y-axis indicates heave instead of settlement, meaning that the soil moves upward rather than downward.

The second thing that is clearly visible is that the reaction to the applied load is different at the ends than in the middle. The ends of the tunnel part are free to move upward and downward, making them more susceptible to loads.

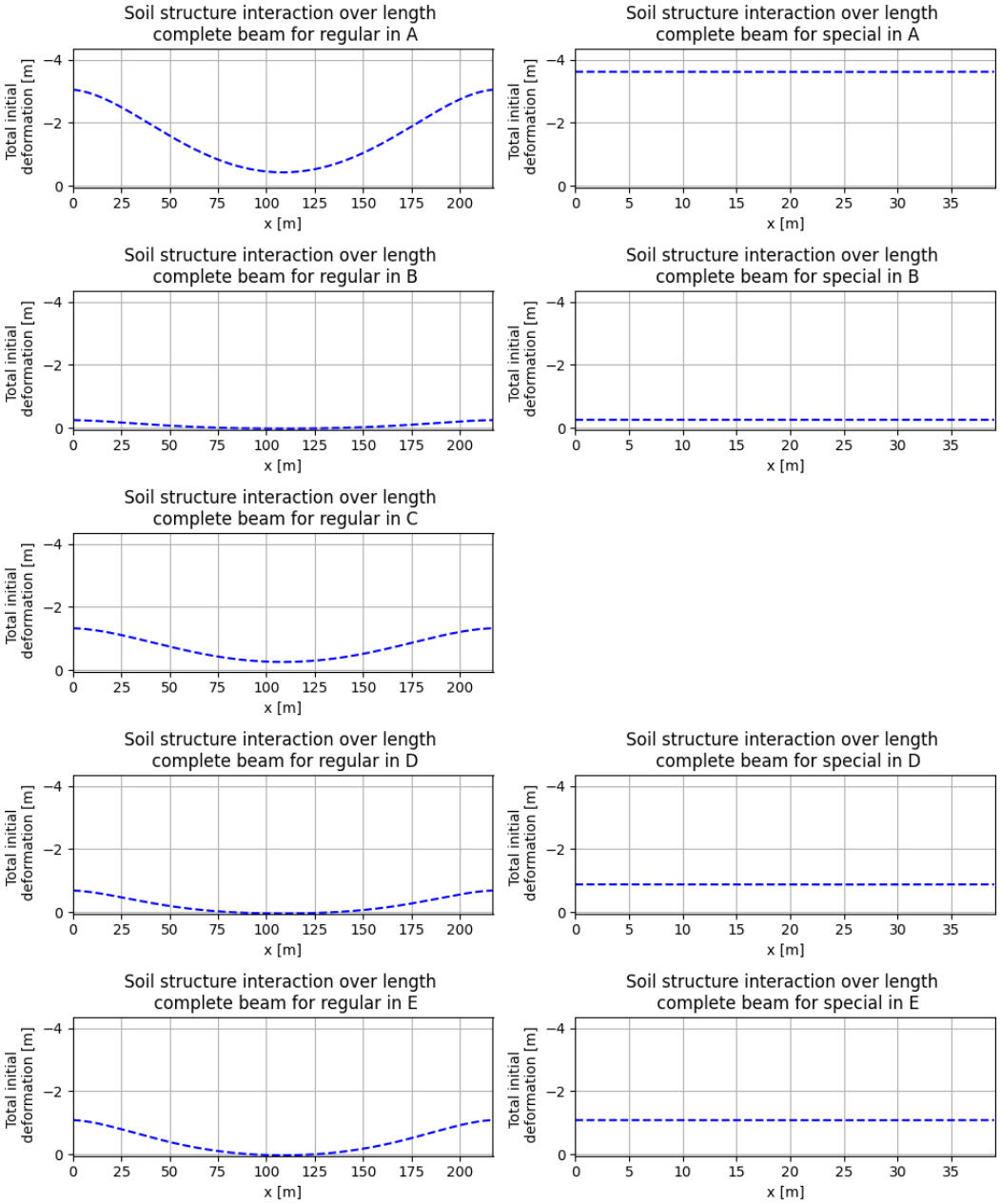


Figure 5.3: Soil structure interaction over the entire length of the entire beam

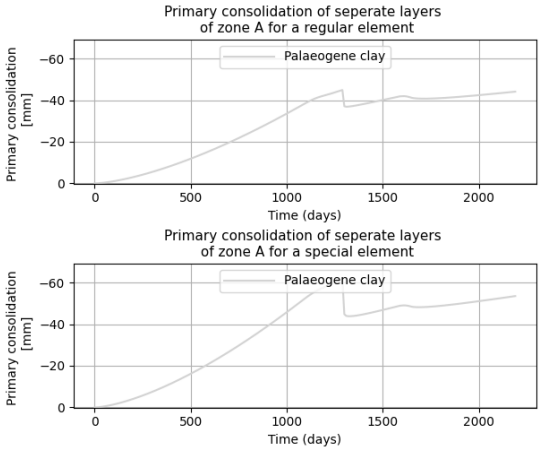
In Zone A, the heave is notably large. This can be explained by the tunnel's location on a very thick (80 m) overconsolidated Palaeogene clay layer, which likely caused this significant heave. Interestingly, the heaves in Zone D and Zone E are quite similar, despite their different soil compositions. This suggests that the influence of the Palaeogene clay at the bottom may not be significant to the initial deformation, possibly because it lies outside the influence boundary of a load at the top.

Furthermore, the initial heave in zone B is quite low compared to the other zones. Zone B has the advantage that there is a sand layer in between the cohesive layers, making the initial heave lower. Lastly, the difference in heave between Zone C and Zone D is quite substantial. Although the soil compositions of Zone C and Zone D are almost identical, except for their lowest layer, this suggests that the chalk layer or the height of the water column plays a significant role in the heave. This question will be further explored through variability analysis.

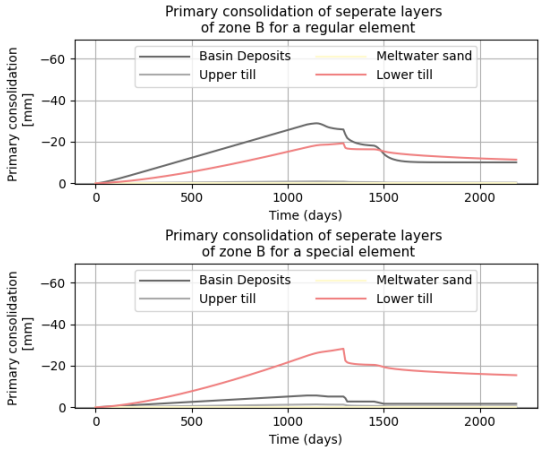
5.2. Result of the Conte and Troncone method for one simulation

The Conte and Troncone method is used to assess the primary consolidation of the soil layers. This section details the primary consolidation results for each zone, highlighting the variations in consolidation behavior due to differences in soil composition and thickness. This section provides a detailed analysis of the primary consolidation phase, highlighting the importance of soil parameters in predicting long-term deformation.

The results of the Conte and Troncone method (2006) are shown in Figures 5.4, 5.5 and 5.6. In the figures, several interesting observations can be made. First, it should be noted that the chalk and meltwater sand layers exhibit very low or zero primary consolidation. This is expected because these layers do not consolidate, as no excess pore pressure builds up within them. In addition, the negative values on the y-axis indicate the heave that occurs as a result of the unloading of the soil. After this unloading step, loads are applied, causing the primary consolidation to develop towards the x-axis from that point.



(a) Primary consolidation of separated layers Zone A



(b) Primary consolidation of separated layers Zone B

Figure 5.4: Primary consolidation of separated layers in zone A and B

Another notable observation is that the special element is heavier, which results in a larger decrease in heave around 1200 days for the special element case compared to the regular element. The primary consolidations in Zone A, Zone D and Zone E are close to each other, suggesting that the presence of Palaeogene clay in these zones plays a significant role in the primary consolidation.

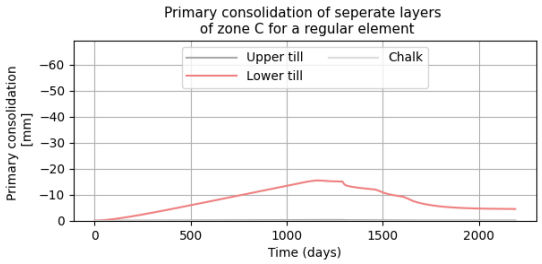
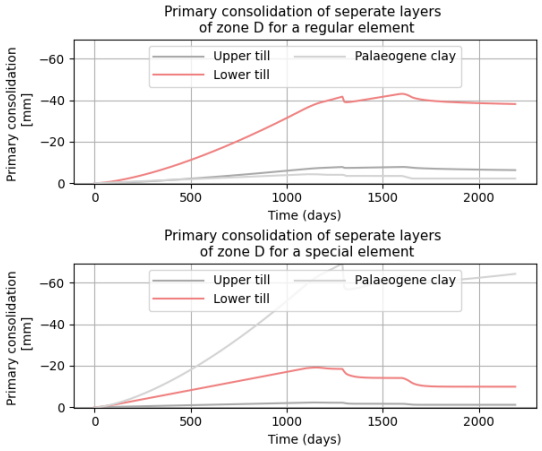
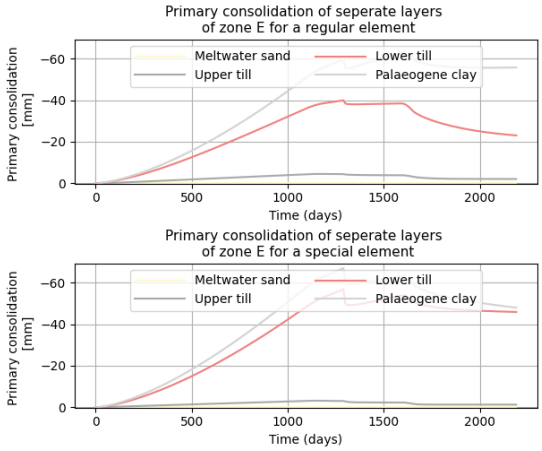


Figure 5.5: Primary consolidation of separated layers Zone C

Furthermore, the primary consolidation of the lower till layer in Zone C is lower than that in zone B and half as thick in Zone C. This is the case because the lower till layer has a permeable boundary at the bottom in Zone C, which is not the case in Zone D and Zone E. In addition, more soil must be excavated for a special element. This leads to larger heave values than Zone C because it only contains regular elements.



(a) Primary consolidation of separated layers Zone D



(b) Primary consolidation of separated layers Zone E

Figure 5.6: Primary consolidation of separated layers in zone D and E

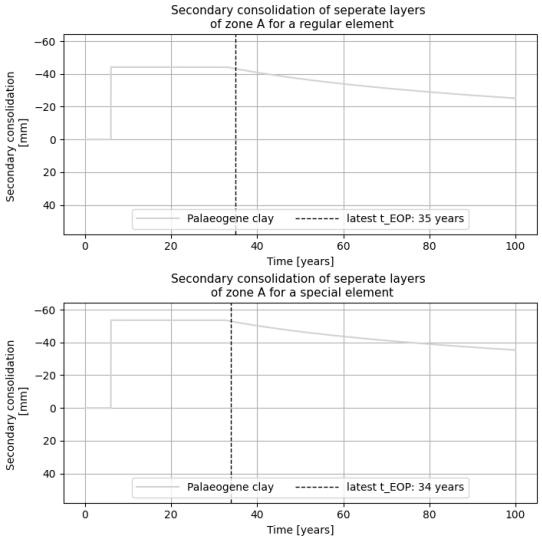
The upper till layer shows a very small value for primary consolidation compared to the other layers. This small value can be explained by the combination of a higher Over-Consolidation Ratio (OCR) than other soil layers (similar to the lower till layer) and lower oedometer stiffness (compared to the lower till layer). The other soil parameters are comparable to those of the other layers. As mentioned earlier, the soil parameters are highly sensitive due to the models used in this thesis.

Finally, all soil layers develop between 1 and 6 centimeters of heave over 2190 days, which is relatively small compared to their own thickness, varying from 2 to 30 meters in thickness.

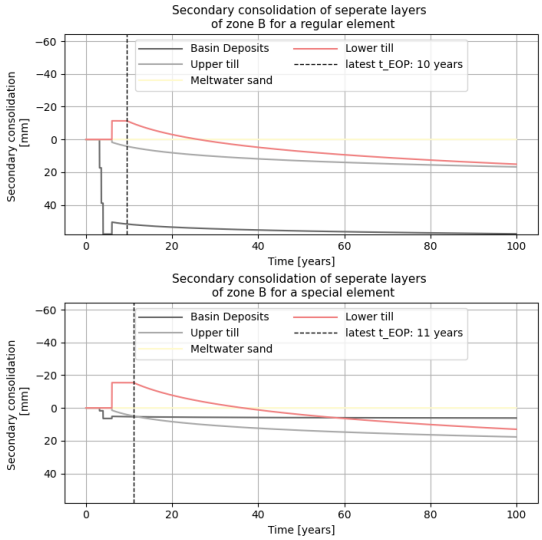
5.3. Result of the Feng et al. method for one simulation

The Feng et al. method is used to evaluate secondary consolidation, or creep, of soil layers. This section presents the secondary consolidation results for each zone, discussing the impact of sustained loading on soil deformation over time. This section explores the mechanisms driving secondary consolidation and the role of equivalent time in predicting creep behavior.

The results of the Feng et al. method (2017) are shown in Figures 5.7, 5.8 and 5.9. In the figures, several interesting observations can be made. Firstly, the zero value on the y-axis represents the heave of the soil layer as a result of the unloading. Both the meltwater sand and chalk layers exhibit zero secondary consolidation, which is expected because no excess pore pressures build up in these layers and the creep is insignificant.



(a) Secondary consolidation of separated layers Zone A



(b) Secondary consolidation of separated layers Zone B

Figure 5.7: Secondary consolidation of separated layers in zone A and B

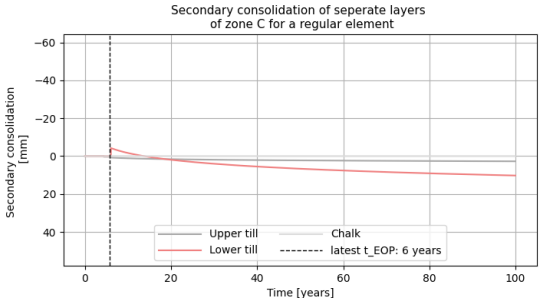
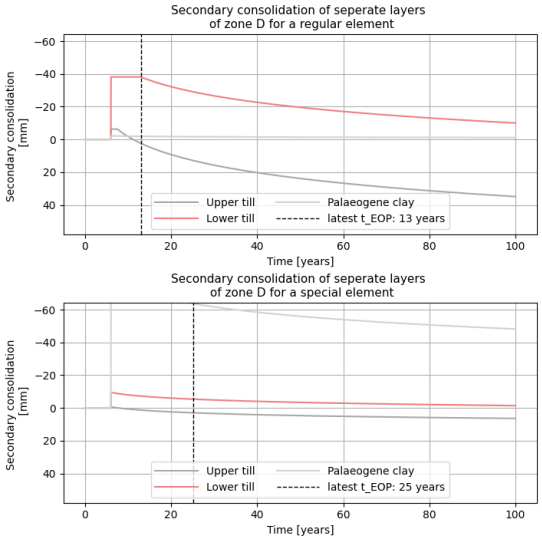
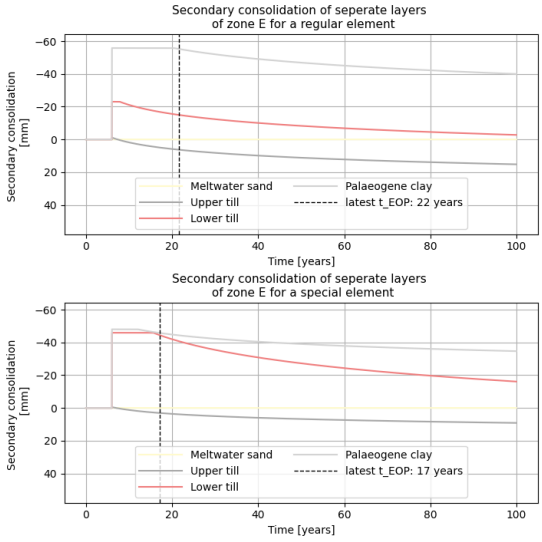


Figure 5.8: Secondary consolidation of separated layers Zone C

The vertical striped lines in the figure indicate the end of primary consolidations for each loading phase. The rightmost line not only marks the end of primary consolidation for the last loading phase but also signifies the start of delayed creep. The basin deposit layer is the only layer that exhibits creep during the loading phases, which can be attributed to its lower overconsolidation ratio (OCR). The OCR plays an important role in determining the stress state and the preconsolidation pressure.



(a) Secondary consolidation of separated layers Zone D



(b) Secondary consolidation of separated layers Zone E

Figure 5.9: Secondary consolidation of separated layers in zone D and E

In addition, both the basin deposit and the upper till layers exhibit enough secondary consolidation over time to transition from heave to settlement. In contrast, the lower till and Palaeogene clay layers show very small creep settlement over time across all zones. The difference between regular and special elements depends on the initial state in terms of stress and strain of the soil layer and on the starting time governed by the end of primary consolidation. The secondary consolidation of a regular and special element do not differ in shape from the end of primary consolidation.

5.3.1. Total consolidation per zone

Combining the results of primary and secondary consolidation provides a comprehensive view of total deformation for each zone. This subsection presents the total consolidation results, highlighting the cumulative impact of soil thickness variability on long-term deformation. It discusses the differences in the consolidation behavior between regular and special elements, providing valuable insights for tunnel design and construction.

In this section, the total consolidation per zone is shown. Figure 5.10 shows the total primary consolidation per zone, which is a sum of all the layers presented in Figures 5.4, 5.5 and 5.6. Figure 5.11 shows the total secondary consolidation per zone, which is a sum of all the layers in Figures 5.7, 5.8 and 5.9. The final figure (5.12) in this section gives the total consolidation per zone, where the primary and secondary consolidations are added at their respective and overlapping times and presented on a broader timescale. The dotted lines in all figures represent the starting times of the respective loading/unloading steps.

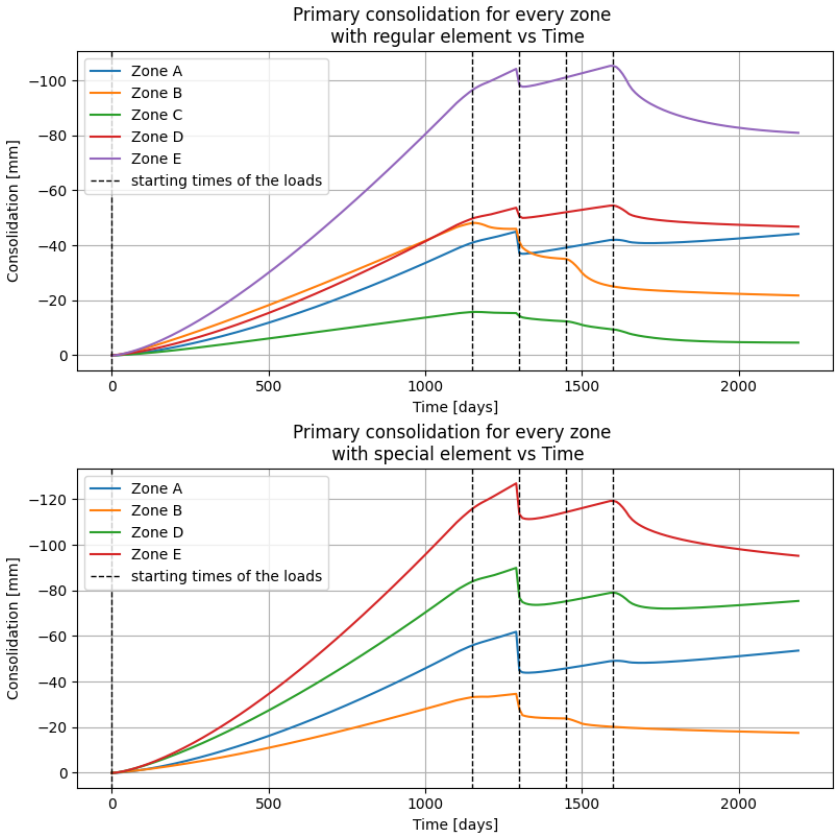


Figure 5.10: The primary consolidation of zones

The main takeaways from the primary consolidation in Figure 5.10 are as follows. Due to the large unloading step, all zones exhibit heave at 2190 days. The sudden decrease in values at 1300 days is much larger with the special elements than with the regular elements due to the difference in weight of the elements. This causes more heave during unloading because more soil needs to be removed, leading to larger settlement in the loading phase of the element. Additionally, it generates more pore pressures,

so the increase in primary consolidation is greater in the special element case.

Zone A and the special element in zone D continue to increase in heave at the end of the graph, while Zones B and E settle significantly toward the end of the graph. Zones D and E have the same soil composition in their special element case (because the meltwater sand layer in Zone E is removed to place the special element), but still show a significant difference in primary consolidation at 2190 days. This difference can only be attributed to the difference in soil thickness. Furthermore, Zone E for the special element case shows a much sharper increase in primary consolidation at the final loading step compared to the regular element case.

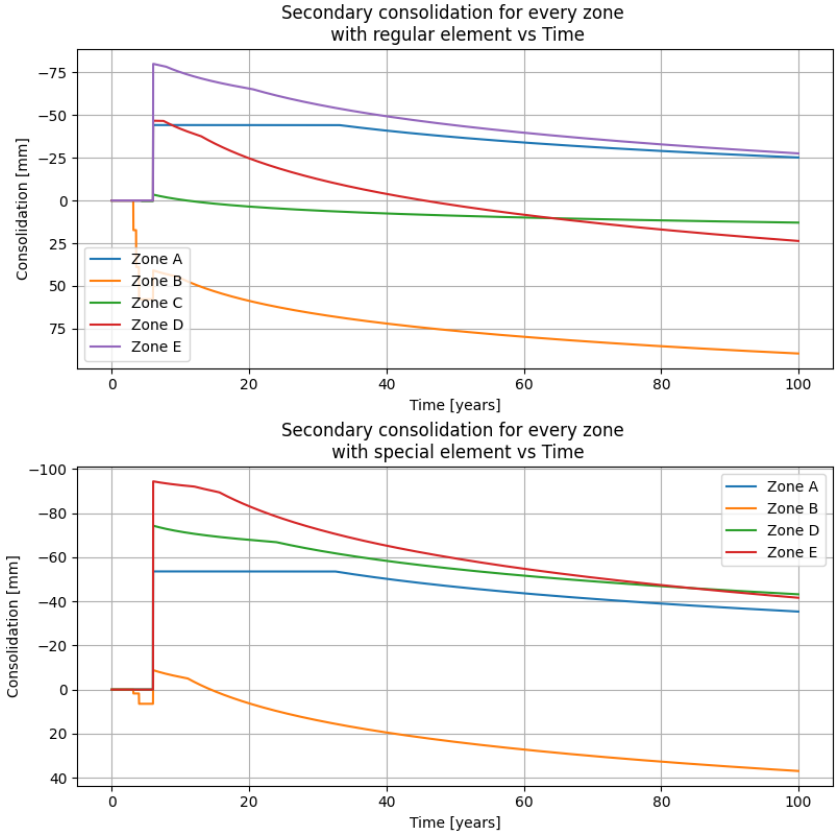


Figure 5.11: The secondary consolidation of zones

The main takeaways from the secondary consolidation in Figure 5.11 are as follows. Zone B is the only zone that exhibits creep deformation during the primary consolidation phase. This occurs because Zone B is the only zone with a basin deposit layer within its soil composition, which directly results in a positive value for secondary consolidation at the start of delayed creep deformation for the regular element. The special element however, starts its secondary consolidation phase with some heave. Additionally, Zone B, Zone C, and Zone D for the regular element case all end up generating settlement instead of generating heave.

The delayed creep differs a lot at both cases (special or regular element), but from a certain point in time they follow the same shape. This starting difference results in a significant difference between the cases of regular and special elements in Zone D, leading to a difference between heave and settlement. Zone A has a much longer time to reach the end of the primary consolidation, so it will start its delayed creep strain fairly late.

Zones E and D for the regular element follow a similar shape in secondary consolidation because their soil compositions are the same and only their mean thicknesses differ. Finally, the regular element case exhibits less heave than the special element case after 100 years of consolidation.

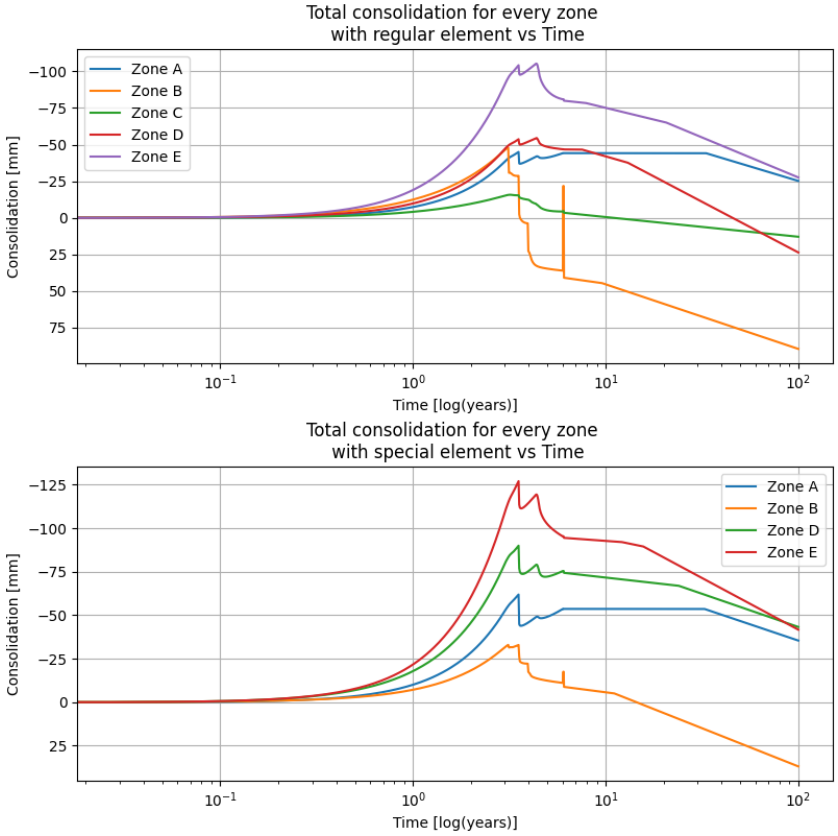


Figure 5.12: Total consolidation of zones

The total consolidation per zone is visualized in Figure 5.12. Several interesting findings can be observed in this figure. Firstly, Zone B for the regular element case shows a large peak. This peak is due to a mistake in converging the primary and secondary consolidations together and does not influence the rest of the values. In Zone A, primary consolidation contributes a larger part to the final deformation than secondary consolidation over a 100-year period.

The consolidation at 100 years for each zone varies between 4 centimeters of heave and 8 centimeters of settlement. Although this may not seem significant for a soil column of about 80 meters, it still represents a 12-centimeter difference between zones. It is important to note that this heave/settlement value is calculated without considering the initial heave/settlement.

In Zone C (for the regular element case), Zone D (for the regular element case) and Zone B (for the special element case) primary consolidation plays an equally significant role in the final deformation as secondary consolidation over a 20-year, 80-year and 20-year period respectively. In contrast, in Zone D (for the special element case) and Zone E (for the regular and special element case), primary consolidation contributes a larger part to the final deformation than secondary consolidation over a longer period.

5.4. Total tunnel deformation for one simulation

Understanding the total deformation of the tunnel structure is crucial to assessing its stability and serviceability. This section presents the total deformation results for the entire tunnel, considering both regular and special elements. This section provides a detailed analysis of deformation patterns over specific time intervals, emphasizing the importance of soil thickness variability in predicting tunnel performance.

The figures in this section show the deformation of the tunnel parts over their length axis. The change between a regular element and a special element on this x-axis is very abrupt, and the deformation does not gradually increase or decrease from a regular element to a special element or the other way around.

In reality, a gradual increase or decrease is present, but this is not included in this research and is not needed for the goal of this thesis.

The total deformation over the length of the tunnel is visualized at specific times. These times are *5 years, 10 years, 20 years, 25 years, 50 years, and 100 years*. The 5 years is chosen to be still within the primary consolidation period to give a nice comparison between primary and secondary consolidation, the 10 years is just after the end of primary consolidation for all the zones, so will show the consolidation at the onset of the delayed creep strain. The other times are chosen at random and just a round number and the double of the number of years before that.

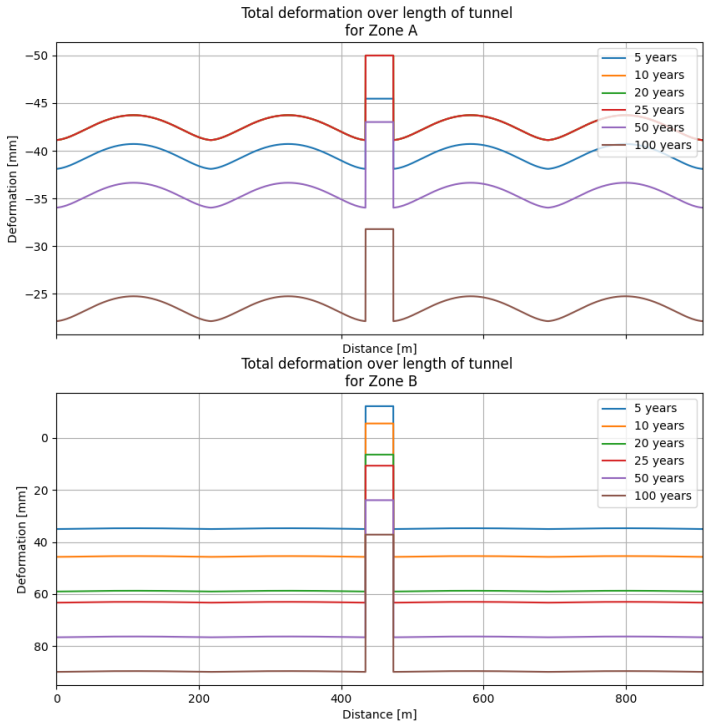


Figure 5.13: Total deformation of the soil column over the length of the tunnel at specific times of zone A and B

Zone B is the only zone that exhibits mainly settlement instead of heave. The difference in initial deformation over the length of the regular elements in Zone A is larger than in the other zones, creating a smaller difference in maximum deformation compared to the special element. Zone C exhibits very small heaves and settlements compared to the other zones. Furthermore it is interesting to notice in Zone C that the difference between 5 years and 10 years is almost the same as the difference between 50 and 100 years. Zone D shows the interesting fact that the special elements settles much slower than the regular element.

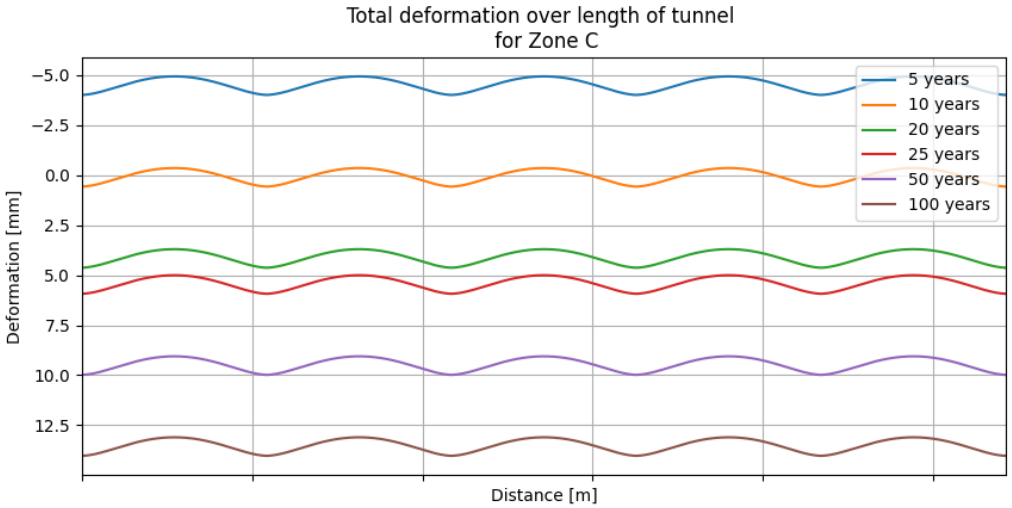


Figure 5.14: Total deformation of the soil column over the length of the tunnel at specific times of zone C

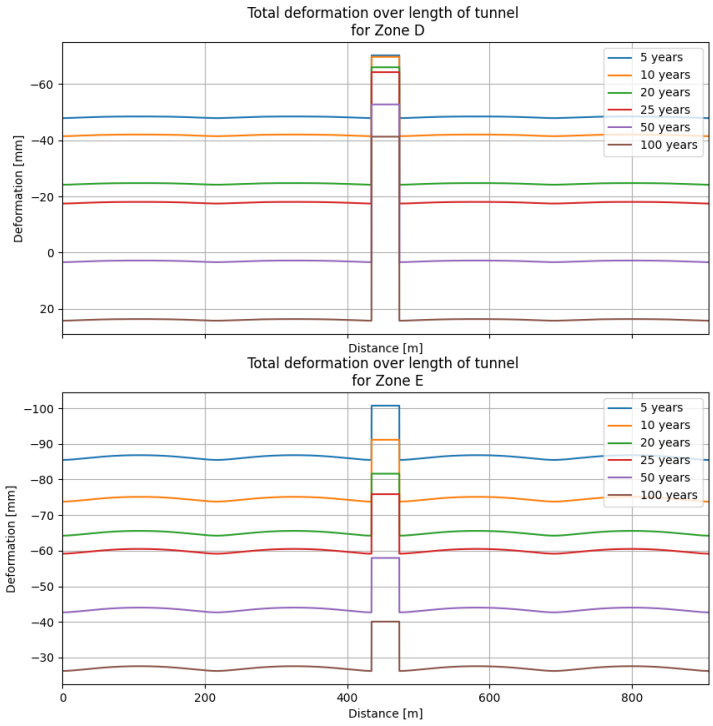


Figure 5.15: Total deformation of the soil column over the length of the tunnel at specific times of zone D and E

There are no visible differences in deformation between 20 and 25 years and between 5 and 10 years for Zone A. This is because the delayed creep strain will onset after 25 years. In Zone E, the difference in deformation from 5 years to 10 years is much greater than the difference between 20 years to 25 years, indicating that the deformation changes more rapidly in the earlier and later years. The regular elements of Zone B, Zone C, and Zone D end up settling after 100 years. There is a discrepancy between the elements that settle or heave after 100 years in Zone B and Zone D.

5.5. Remarkable results in deformation for one simulation

Certain results stand out due to their significant impact on soil deformation predictions. This section highlights the most remarkable findings from the single simulation, discussing their implications for tunnel design and construction. It provides a summary of key observations, emphasizing the importance of considering soil variability in geotechnical analysis.

A summary of the results presented in this chapter is shown below.

Zone A:

- **Primary Consolidation:** Continues to increase in heave towards the end of the graph.
- **Secondary Consolidation:** The delayed creep starts fairly late due to a longer time to reach the end of the primary consolidation and there is no creep present during the primary consolidation, due to the high OCR of the soil.
- **Total Consolidation:** Primary consolidation plays a larger role in the final deformation than secondary consolidation over 100 years.
- **Total deformation:** No visible difference in deformation between 20 and 25 years and between 5 and 10 years due to the onset of delayed creep deformation after 25 years.

Zone B:

- **Primary Consolidation:** Exhibits creep strain during the primary consolidation phase due to the presence of a basin deposit layer.
- **Secondary Consolidation:** Generates settlement instead of heaving.
- **Total Consolidation:** Contributions of primary and secondary consolidations to the final deformation are equal over 100 years.
- **Total deformation:** Settlement occurs for every timestep. The difference in deformation from 5 to 10 years is much greater than from 20 to 25 years, indicating rapid changes in the earlier years.

Zone C:

- **Primary Consolidation:** The primary consolidation of the lower till layer is half that of Zones D and E due to thinner layer thickness.
- **Secondary Consolidation:** Slightly greater than primary consolidation in terms of impact on final deformation over 100 years for the regular element. Its impact is far lower for the special element.
- **Total Consolidation:** Varies between -0.5 cm heave to a 1.2 cm settlement over 100 years.
- **Total deformation:** Regular elements settle after 100 years.

Zone D:

- **Primary Consolidation:** Plays an equally significant role in final deformation as secondary consolidation over 100 years (regular element case).
- **Secondary Consolidation:** Plays a smaller role in final deformation than primary consolidation over 100 years (special element case).
- **Total Consolidation:** Special elements settle slower than the regular elements
- **Total deformation:** Regular elements end up settling after 100 years. Discrepancy between elements for settling or heaving after 100 years.

Zone E:

- **Primary Consolidation:** Shows a greater increase in primary consolidation at the final loading step for special elements compared to regular elements.
- **Secondary Consolidation:** Similar increase in secondary consolidation for special elements due to identical soil compositions after soil removal.
- **Total Consolidation:** Secondary consolidation plays a lesser role in final deformation as primary consolidation over 100 years.

- **Total deformation:** The differenc in total deformation between the regular and special element decreases over time.

And for the separate soil layers the results for one simulation were as follows:

Palaeogene clay

- **Zone A:** The presence of a very thick (80 m) overconsolidated Palaeogene clay layer causes significant heave.
- **Zone D and Zone E:** The influence of Palaeogene clay is less significant, possibly due to its position outside the influence boundary of a load at the top.
- **Secondary consolidation:** Exhibits very small secondary consolidation over time in all zones.

Basin deposits

- **Zone B:** Exhibits creep during the loading phases due to its lower Over-Consolidation Ratio (OCR), which plays a significant role in the stress state and preconsolidation pressure.
- **Secondary consolidation:** Contributes to the transition from heave to settlement over time.

Upper till

- **Primary consolidation:** Shows a very small value for primary consolidation compared to other layers, possibly due to a combination of higher OCR and lower oedometer stiffness.
- **Secondary consolidation:** Contributes to the transition from heave to settlement over time.

Lower till

- **Zone C:** Primary consolidation is half that of Zones D and E due to the thinner layer thickness.
- **Secondary consolidation:** Exhibits very small creep settlement over time in all zones.

Meltwater sand

- **Primary and secondary consolidation:** Exhibits zero primary and secondary consolidation due to the absence of excess pore pressures and insignificant creep.
- **Zone E:** Removal of the meltwater sand layer for the special element results in a soil composition similar to that of Zone D.

Chalk

- **Primary and secondary consolidation:** Exhibits zero primary and secondary consolidation due to the absence of excess pore pressures and insignificant creep.

Water level

- **Heave and settlement:** The height of the water column plays a significant role in the heave and settlement, particularly in Zone C and Zone D.

6

Variability results for the entire tunnel configuration

This chapter covers the results of the models generated with 500 different simulations for the soil thicknesses in the soil column. As a baseline, a standard deviation of 1 meter is chosen for the thicknesses of the soil layers. In Section 6.5 an overview with other values of standard variation and what their effect is on the final variability in deformation is given. All the negative values for deformation in this chapter indicate heave and all positive values for deformation indicate settlement of the soil.

6.1. Variability of results of the TBKF model for 500 simulations

The Timoshenko beam on the Kerr foundation model (TBKF model) is used to analyze the initial deformation of the soil-structure interaction. This section presents the variability results of the TBKF model for 500 simulations, focusing on the differences in deformation between regular and special tunnel elements in various zones. It discusses the factors that contribute to the variability in heave and settlement, providing a comprehensive understanding of the initial response of the soil to loading.

Zone A
Zone A does not produce variable results. The composition of the soil column is determined by Monte Carlo simulations that generate 500 times a new soil column with different soil thicknesses for the layers present within that column. Zone A only has the Palaeogene layer present beneath the tunnel elements, so will not generate different results for different layer thicknesses, because the place of the tunnel and the height of the water level do not change. This makes zone A a perfect check for the model because if strange values appeared here, something would be wrong with the calculation model.

Zone B
The variability in the initial deformation of a regular and a special element are shown in Figures 6.1 and 6.2 . The interesting findings in this figure are:

- The variation in initial deformation close to the center is very close to zero or even zero.
- The variation in initial deformation grows towards the ends of the beam of the element.
- The variation is between -0.1 and -0.5 mm for a regular element.
- The variation is between -0.1 and -0.6 mm for a special element.
- The distribution of results at the end of the beam is a truncated lognormal distribution, just as the input in the simulation.
- The distribution of results is wider for the special element than for the regular element.

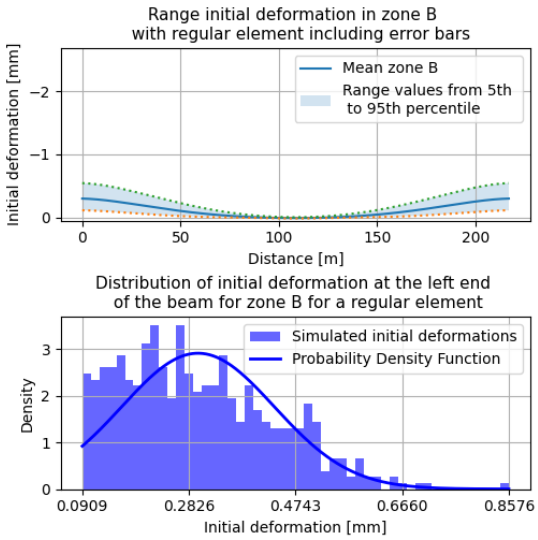


Figure 6.1: Variability initial deformation regular elements for zone B

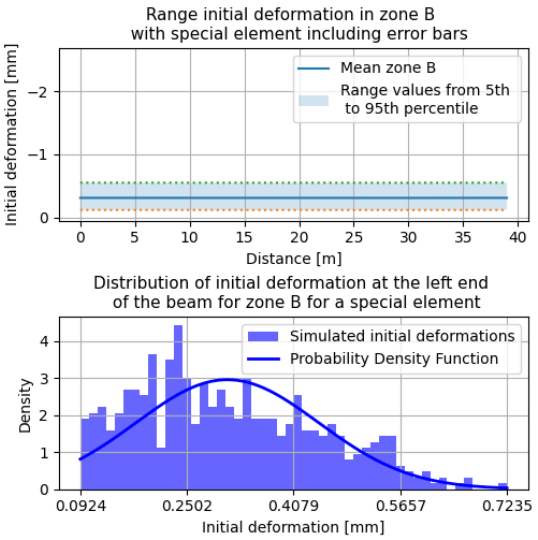


Figure 6.2: Variability initial deformation special elements for zone B

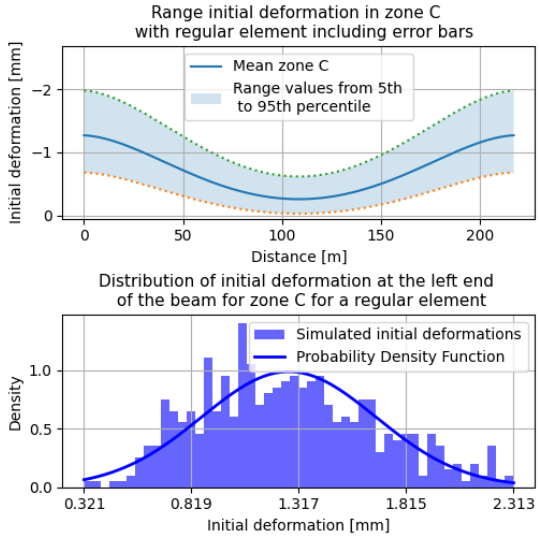


Figure 6.3: Variability initial deformation regular elements for zone C

Zone C

The variability in initial deformation of a regular element is shown in Figure 6.3. The interesting findings in this figure are:

- the mean deformation is twice as big as in Zone B.
- The variation in deformation is increasing towards the endpoints of the beam.
- The variation is between -0.7 and -2.0 mm for a regular element.
- The distribution of results at the end points is a truncated Gaussian distribution. The input was lognormal, so that is an interesting fact. This change can be explained by the fact that the averaging of the soil parameters for the soil stiffness parameter causes this effect.

Zone D

The variability in the initial deformation of a regular and special element are shown in Figures 6.4 and 6.5. The interesting findings in this figure are:

- The mean initial deformation of zone D is about as high as the initial deformation of zone C.
- The overall variance is significantly lower than with zone C or zone E, making this zone more robust in terms of initial deformation.
- The variation lies between -0.4 and - 1.8 mm for a regular element.
- The variation lies between -0.4 and - 2.1 mm for a special element.
- The distribution of values at the end of the beam is a truncated lognormal distribution, just as the input.
- The range of the lognormal output distribution is greater than its input, suggesting that the model amplifies the variability of the input.

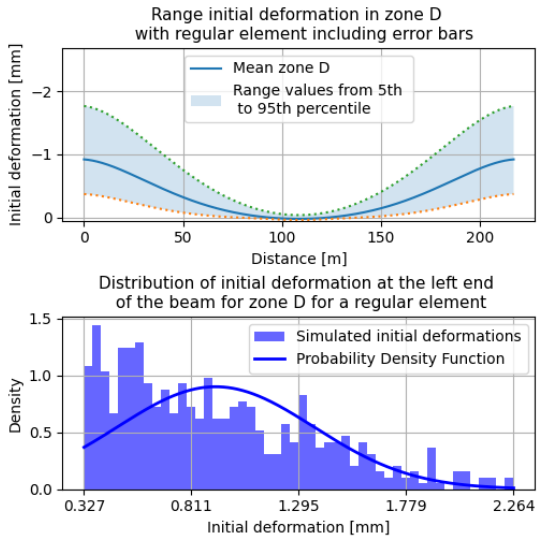


Figure 6.4: Variability initial deformation regular elements for zone D

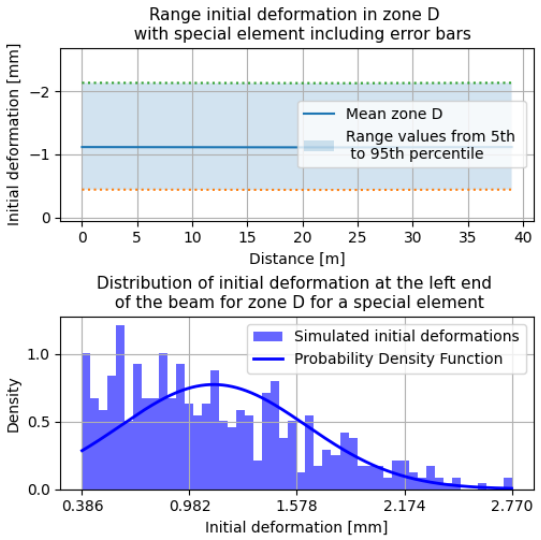


Figure 6.5: Variability initial deformation special elements for zone D

Zone E

The variability in initial deformation of a regular and special element are shown in Figures 6.6 and 6.7. The interesting findings in this figure are:

- The mean deformation is about the same as in Zones C and D.
- The variation is between -0.5 and -2.2 mm for a regular element.
- The variation is between -0.4 and -2.2 mm for a special element.
- The distribution of values at the end of the beam is a truncated lognormal distribution, just as the input.
- The range of the lognormal output distribution is greater than its input, suggesting that the model amplifies the variability of the input.
- The range of deformation values at the end of the beam is the greatest of all zones. This Zone presents a greater risk for designing, due to its broader range in deformation values.

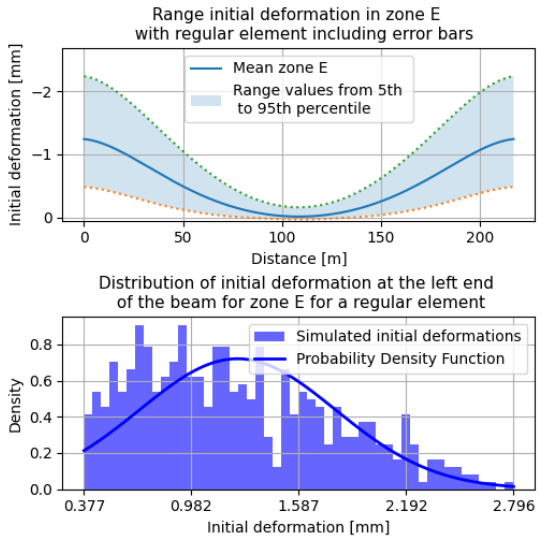


Figure 6.6: Variability initial deformation regular elements for zone E

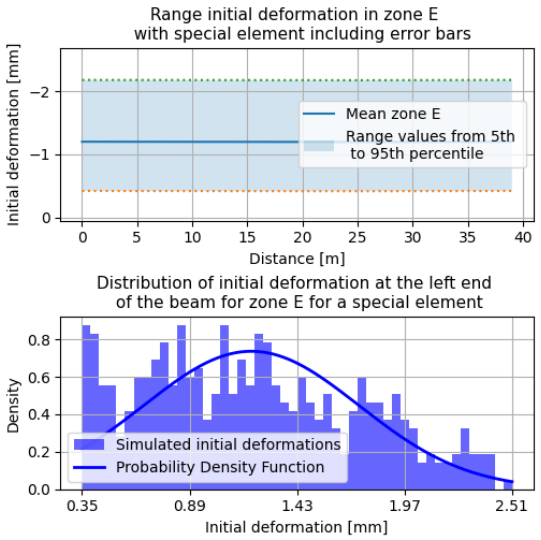


Figure 6.7: Variability initial deformation special elements for zone E

6.2. Variability of results of the Conte and Troncone method for 500 simulations

The Conte and Troncone method is used to assess primary consolidation of the soil layers. This section details the variability in primary consolidation results for each zone, highlighting the variations in consolidation behavior due to differences in soil composition and thickness. Provides a thorough analysis of the primary consolidation phase, highlighting the importance of soil parameters in predicting long-term deformation.

6.2.1. Variability primary consolidation for all layers per zone

This subsection presents the variability of primary consolidation for each layer of soil within the different zones. This subsection discusses the impact of soil thickness variability on the consolidation behavior of individual layers, providing insight into the factors driving variability in primary consolidation.

Zone A

As mentioned in the previous section, there is no variability in zone A.

Zone B

The variability of the primary consolidation of the separate layers in the soil composition of zone B is shown in figures 6.8 and 6.9.

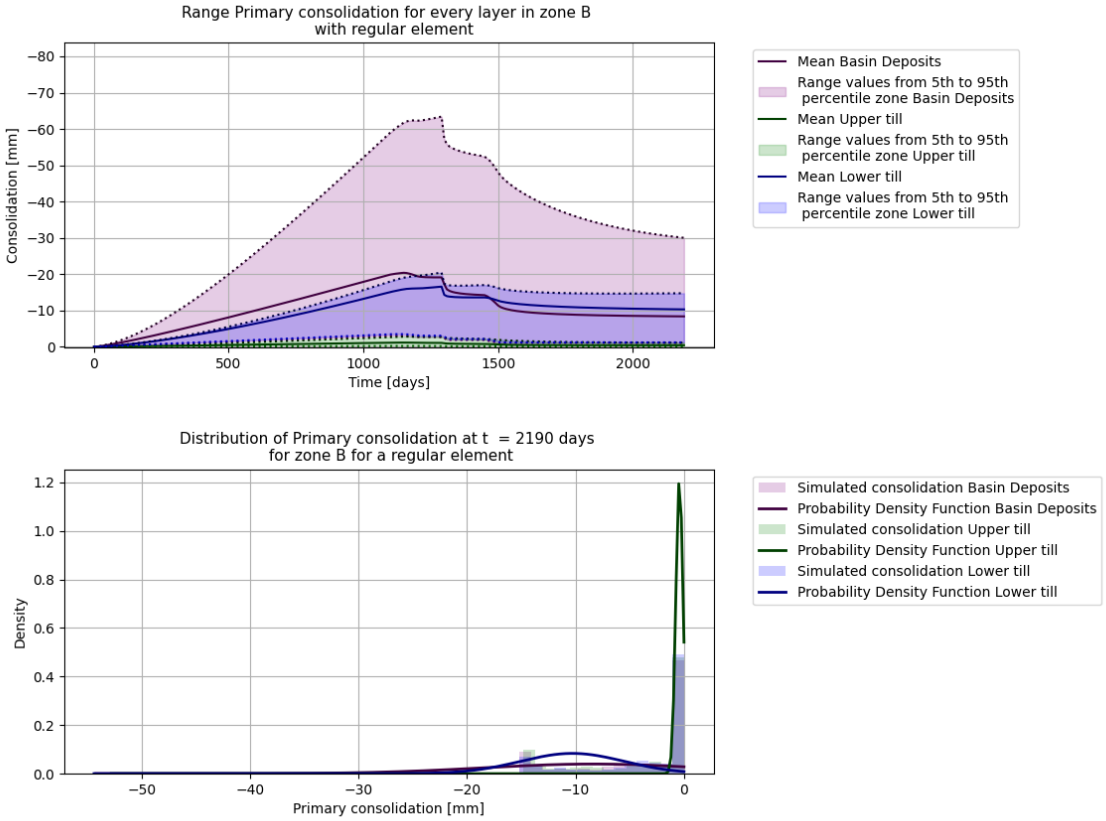


Figure 6.8: Variability primary consolidation regular element zone B

From these results, it stands out that the variation in the basin deposit layer is very high but decreases from the point where the element is installed. This means that during unloading the soil response becomes more sensitive to small differences in input and during loading the system is converging towards a more deterministic response. This means that the unloading path is not simply the reverse of the soil loading path.

Another notable fact is that the variation in the upper till layers is quite low compared to the other layers and follows a truncated lognormal distribution with a high peak. This leads to the conclusion that thinner soil layers suppress the variability of the input and are (partly) insensitive to the input range.

The distributions of the other two layers (basin deposits and lower till) are spread wider. Amplifying the uncertainty of the input and increasing their sensitivity to the input range. Furthermore, the variation in the meltwater sand layer remains zero, as expected, because it does not consolidate.

Regarding the special element, the peaks for the lower till and basin deposit layer are higher, suggesting that the layers are more sensitive to the uncertainty of the input when the soil layers are subjected to a larger unloading of the soil.

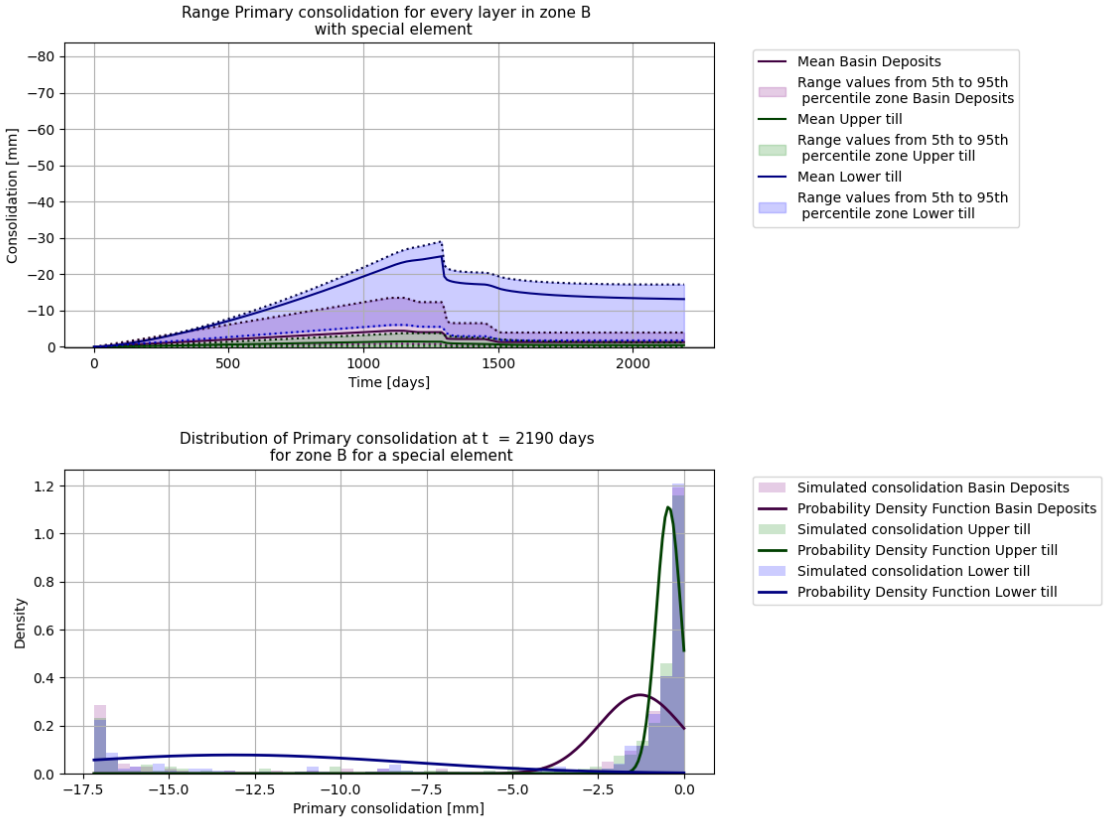


Figure 6.9: Variability primary consolidation special element zone B

The variations in primary consolidation at the end of the graph (2190 days) for layers in zone B are as follows:

Table 6.1: Variation in primary consolidation at 2190 days for soil layers in zone B

Soil type	**Zone B**	
	Regular	***Special***
Meltwater sand	-	-
Chalk	-	-
Upper till	between -0.1 and -1.1 mm	between -0.1 and - 1.1 mm
Lower till	between - 1.2 and -14.7 mm	between -1.7 and -17.2 mm
Basin deposits	between -1.0 and -30.0 mm	between - 0.2 and - 3.9 mm
Palaeogene clay	-	-

Zone C
The variability of the primary consolidation of the separate layers in the soil composition of zone C is shown in Figure 6.10.

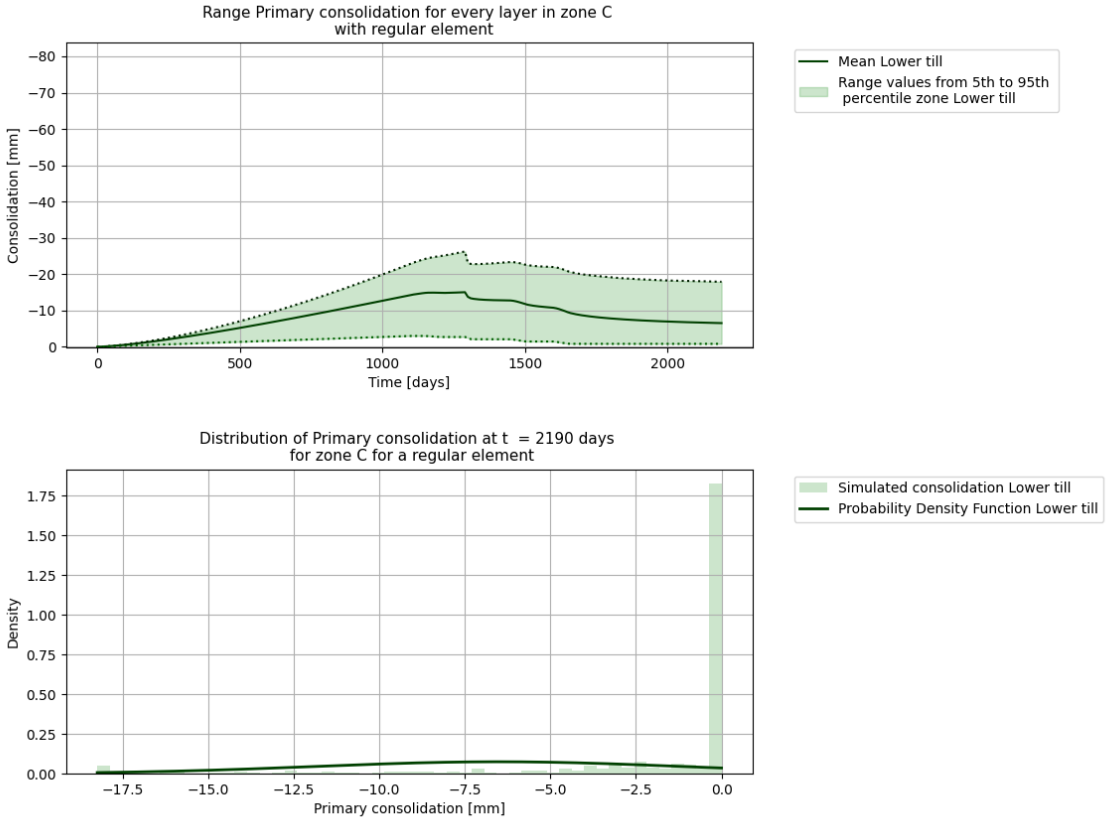


Figure 6.10: Variability primary consolidation separate layers zone C

From the results, it stands out that the variation in the upper till layer is very low and because of this it was removed from the figures. This low variation could be due to either the low consolidation itself or the robust parameters that lead to low variation. The variation in the chalk layer is zero, as expected, because it does not consolidate. In contrast, the variation in the lower till layer is quite high and provides a broader range than in zone B (regular element), with this range remaining constant over time. A broader range means the amplification of uncertainty in this layer and making it more sensitive to input values.

The variations in primary consolidation at the end of the graph (2190 days) for layers in zone C are as follows:

Table 6.2: Variation in primary consolidation at 2190 days for soil layers in zone C

Soil type	**Zone C**	
	Regular	***Special***
Meltwater sand	-	-
Chalk	-	-
Upper till	between - 0.0 and - 0.1 mm	-
Lower till	between - 0.8 and -17.9 mm	-
Basin deposits	-	-
Palaeogene clay	-	-

Zone D
The variability of the primary consolidation of the separate layers in the soil composition of zone D is shown in figures 6.11 and 6.12.

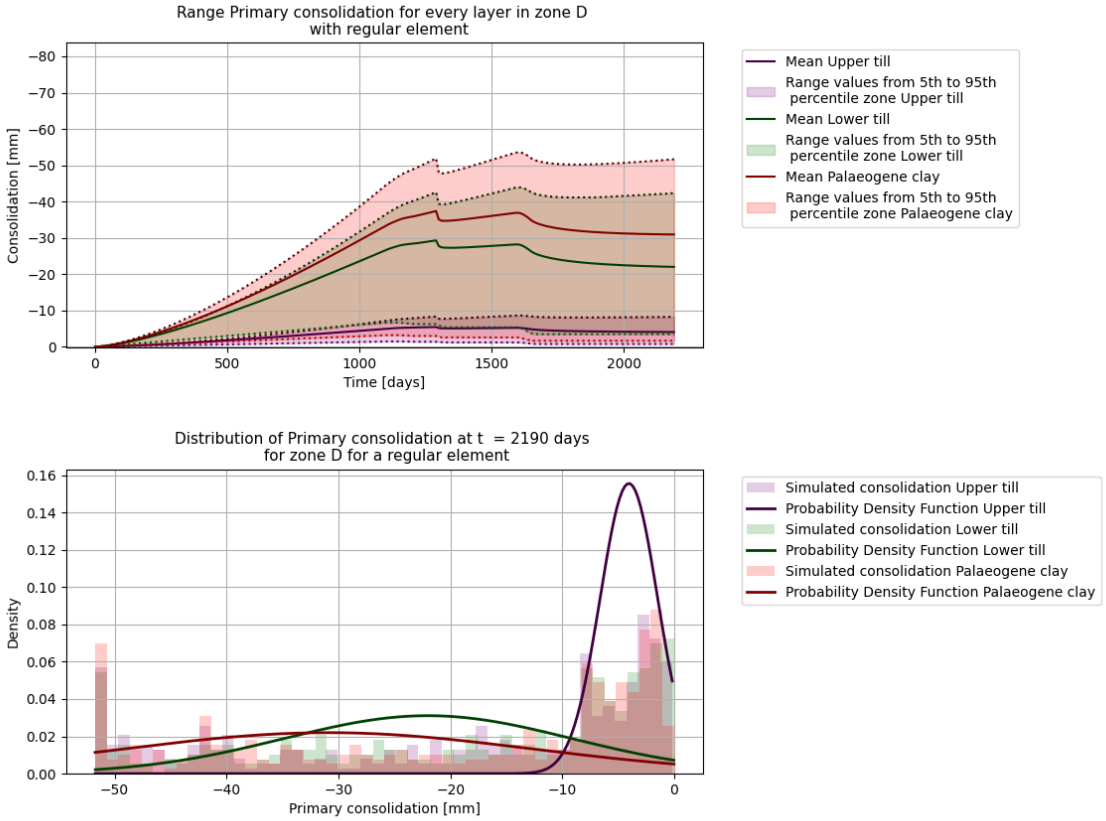


Figure 6.11: Variability primary consolidation regular element zone D

From these results, it stands out that the mean of the upper till layer is higher here than in the other zones, but the range remains very narrow. This suggests that the range in primary consolidation for the upper till layer increases if the primary consolidation itself grows, emphasizing that this layer compresses the variability and is more robust even for higher results in the model.

The variation of the lower till layer increases from the point of placement of the element, making it less robust from that point. The range in primary consolidation for the Palaeogene clay layer is broad and even grows larger at the end of soil unloading. This makes these layers very susceptible to input values and increases the risks that these layers bring to the design phase. In addition, the range of variation for the special and regular elements is quite similar, showing that the effect of larger unloading plays a lesser role here than, for example, in Zone C.

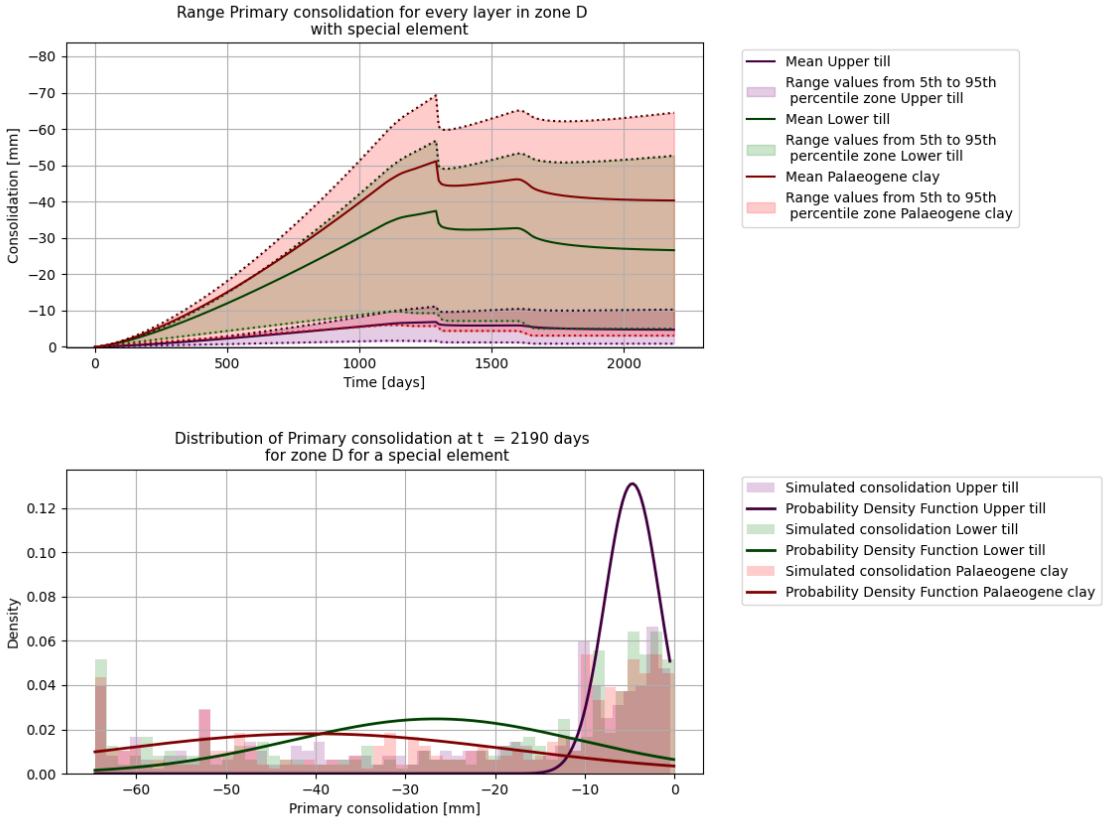


Figure 6.12: Variability primary consolidation special element zone D

The variations in primary consolidation at the end of the graph (2190 days) for layers in zone D are as follows:

Table 6.3: Variation in primary consolidation at 2190 days for soil layers in zone D

Soil type	**Zone D**	
	Regular	***Special***
Meltwater sand	-	-
Chalk	-	-
Upper till	between -0.8 and - 8.2 mm	between -0.8 and - 10.2 mm
Lower till	between -3.5 and -42.3 mm	between -5.0 and -52.7 mm
Basin deposits	-	-
Palaeogene clay	between - 1.6 and - 51.7 mm	between - 3.1 and - 64.5 mm

Zone E
The variability of the primary consolidation of the separate layers in the soil composition of zone E is shown in figures 6.13 and 6.14.

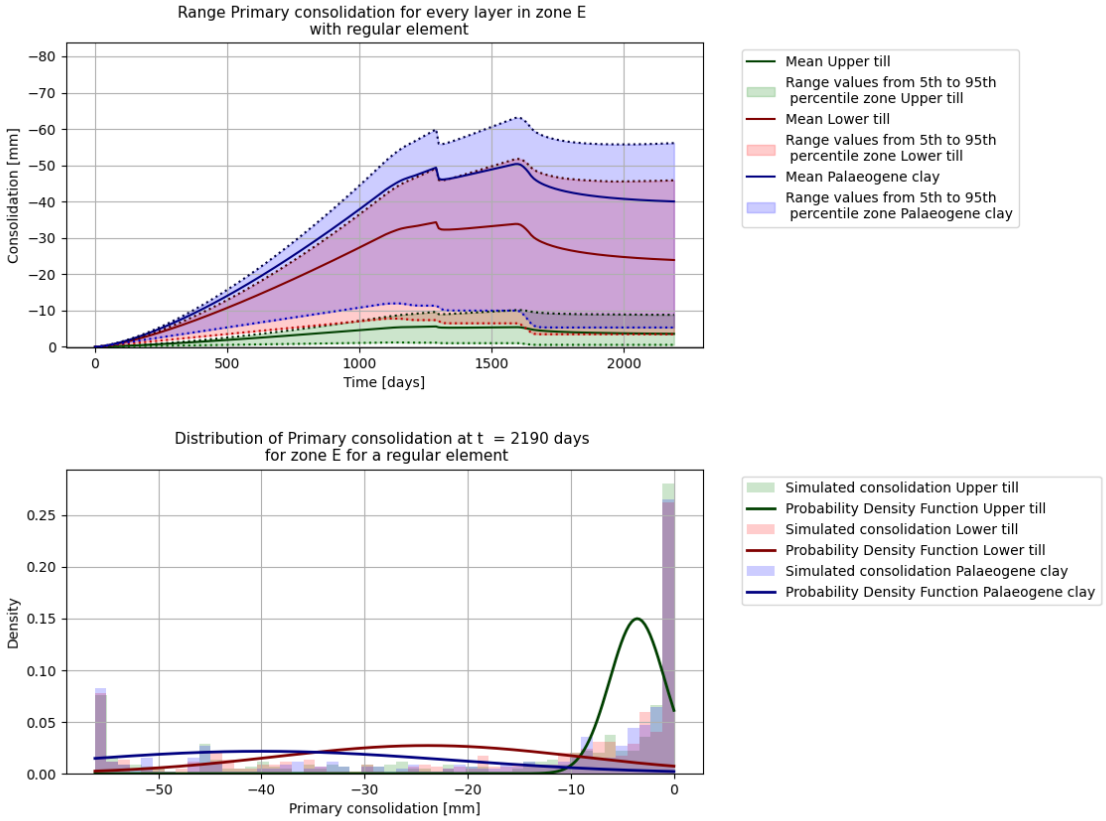


Figure 6.13: Variability primary consolidation regular element zone E

From these results, it stands out that the meltwater sand has no variation, which is correct.

The variation in the upper till is again very narrow, indicating that the uncertainty in the thickness of this layer does not play a major role on the total uncertainty in the primary consolidation phase. The range of variation in the lower till and Palaeogene clay increases over time up to a very broad range in output values, again emphasizing that these layers are very sensitive to the input values and could pose a risk during the design phase. In addition, there is no visible difference in the range between the special element and the regular element.

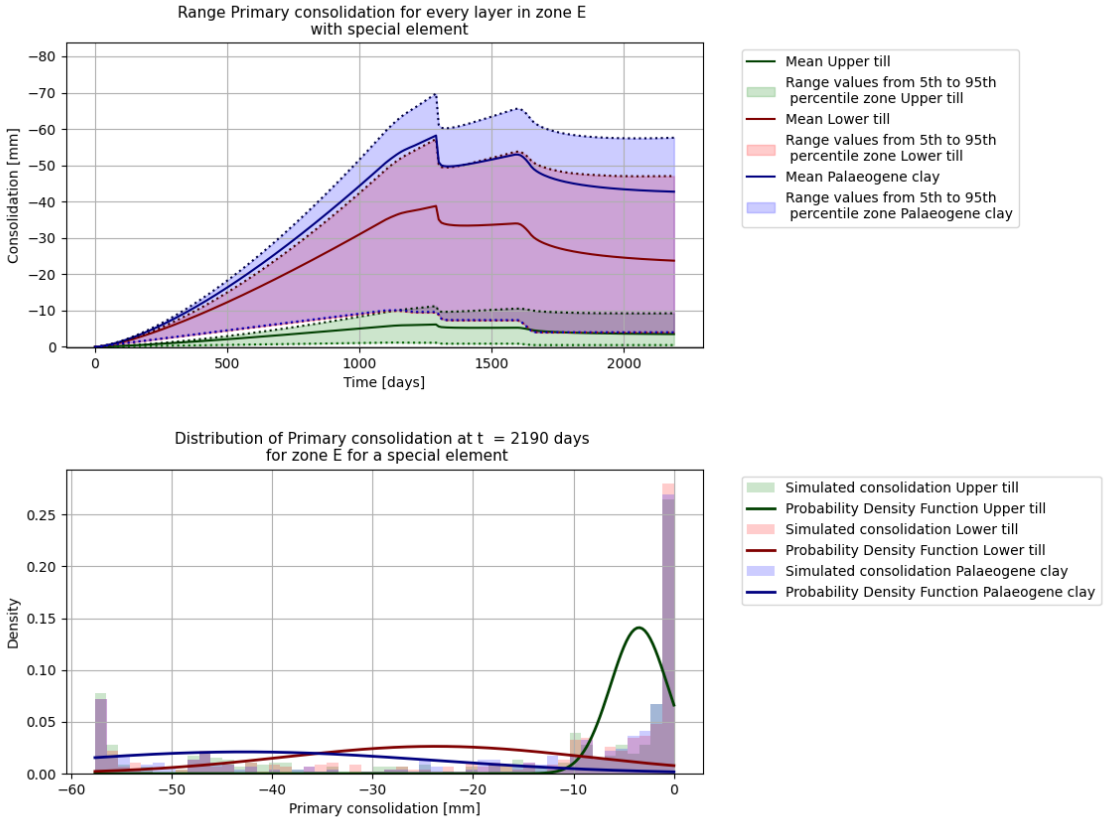


Figure 6.14: Variability primary consolidation special element zone E

The variations in primary consolidation at the end of the graph (2190 days) for layers in zone E are as follows:

Table 6.4: Variation in primary consolidation at 2190 days for soil layers in zone E

Soil type	**Zone E**	
	Regular	***Special***
Meltwater sand	-	-
Chalk	-	-
Upper till	between -0.5 and - 8.8 mm	between - 0.5 and - 9.2 mm
Lower till	between - 3.4 and - 45.9 mm	between -3.9 and - 47.1 mm
Basin deposits	-	-
Palaeogene clay	between - 5.3 and - 56.1 mm	between -4.0 and - 57.6 mm

6.2.2. Variability primary consolidation for all zones

Combining the results of individual soil layers provides a comprehensive view of the variability in primary consolidation for each zone. This subsection presents the overall variability in primary consolidation, highlighting the cumulative impact of soil thickness variability on long-term deformation.

The previous section 6.2.1 described every zone separately and discussed the variation of the soil layers present within the soil composition per zone. This section gives an overview of the variation of the total primary consolidation per zone, and therefore an addition of their separate layers. The overview is visible in Figure 6.15.

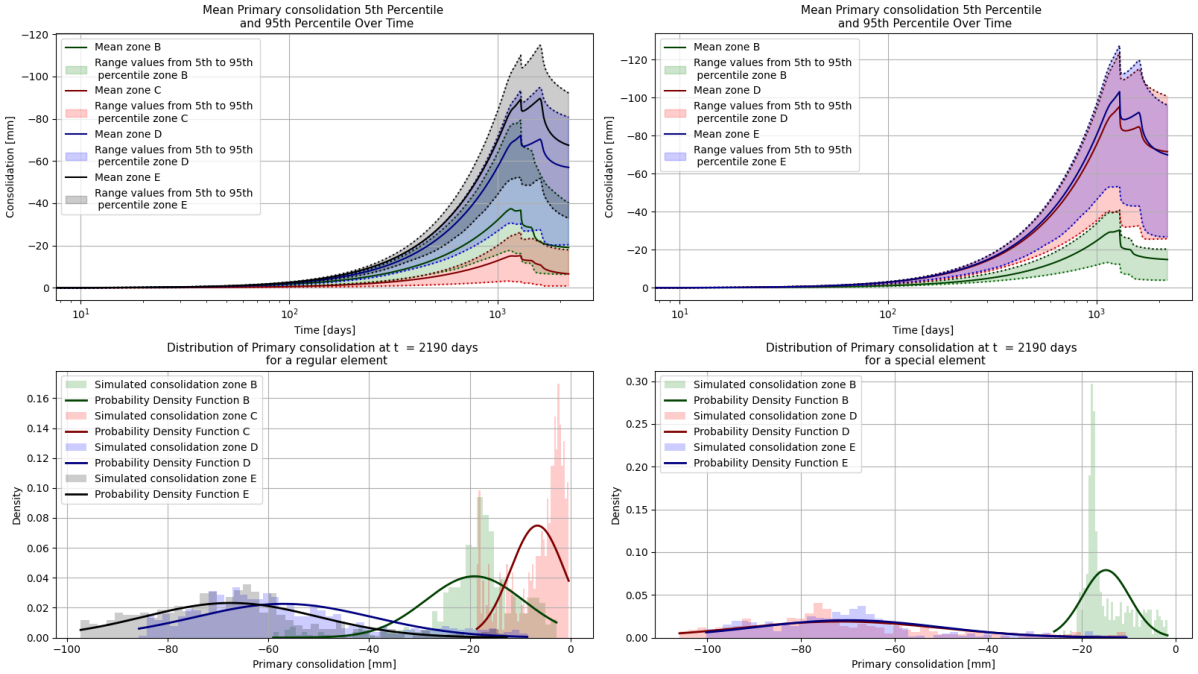


Figure 6.15: Variability of primary consolidation per defined zone

From this overview, it is interesting to note that the variation during the unloading phase starts small and builds up until the next phase. Indicating that the unloading phase amplifies the uncertainty and the loading phase dampens the uncertainty and is more robust.

As can be seen from the distribution, zones B and C have a relatively narrow range. This means that these zones are less sensitive to the input values and give a more robust output. This is perfect for the robustness of the design and the determination of safety margins.

Zone D and Zone E are more problematic during the design phase because these zones amplify the uncertainty of the input values, making it more difficult to determine safety margins for the design. The positive side of these zones is that they normalize the input, resulting in a truncated Gaussian distribution for the output.

The variations per zone are as follows:

Table 6.5: Effect on the range of primary consolidation due to the change in soil thickness

Primary Consolidation	**Element**	**Range in values at 2190 days**
Zone A	***Regular***	-
	Special	-
Zone B	***Regular***	between -6.33 and -40.2 mm
	Special	between -4.0 and -20.4 mm
Zone C	***Regular***	between -0.8 and -17.9 mm
Zone D	***Regular***	between -20.4 and -80.8 mm
	Special	between -25.8 and -100 mm
Zone E	***Regular***	between -33 and -92.2 mm
	Special	between -26.8 and -96.0 mm

In general, it can be said that the range in results for the unloading phase increases over time, and during the loading phases, the range decreases. The influence at the end of the primary consolidation phase is still very strong, ranging from -0.8 mm in zone C to even -100 mm in zone D.

6.3. Variability of results of the Feng et al. method for 500 simulations

The Feng et al. method is used to evaluate secondary consolidation, or creep, of soil layers. This section presents the variability in the secondary consolidation results for each zone, discussing the impact of sustained loading on soil deformation over time. This section explores the mechanisms driving variability in secondary consolidation and the role of equivalent time in predicting creep behavior.

6.3.1. Variability secondary consolidation for all layers per zone

This subsection presents the variability in secondary consolidation for each layer of soil within the different zones. This subsection discusses the impact of soil thickness variability on the creep behavior of individual layers, providing insight into the factors driving variability in secondary consolidation.

Zone A

As said in Section 6.1, no variability is present in zone A.

Zone B

The variability of secondary consolidation of the separate layers in the soil composition of zone B is shown in figures 6.16 and 6.17.

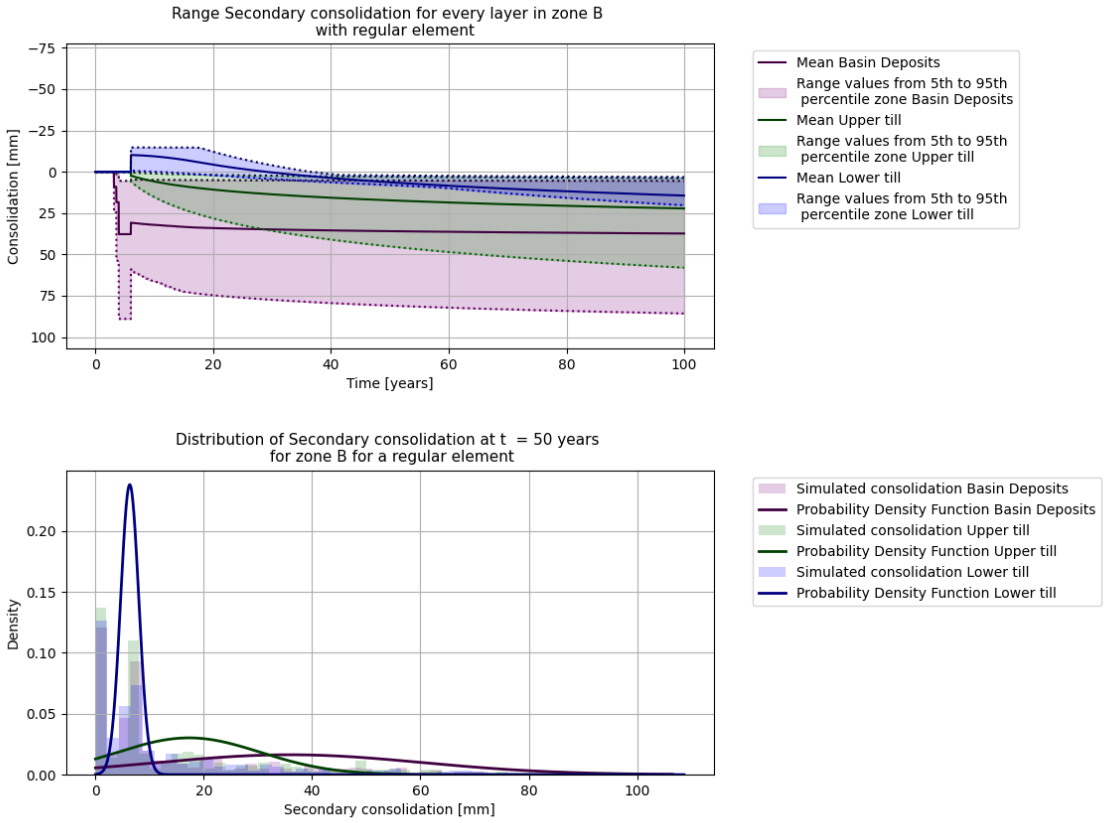


Figure 6.16: Variability secondary consolidation regular element zone B

From these results, it stands out that the range of variation for the basin deposits is very high, but it does not increase with time. The range of variation for the upper till layer increases with time and is relatively large. In contrast, the range of variation in the lower till layer decreases with time up to 45 years, and then starts to increase again over time.

The distribution shows a very narrow range for the lower till layer at 50 years, which is interesting

because the opposite was true for the primary consolidation phase. The lower till layer becomes more robust towards this time and becomes less robust from this point in time, making it a point of interest to check the functionality of the tunnel at 45 years. The other two layers have a much broader range and are more sensitive to the uncertainty of the input values.

The same holds for the special element; the only difference here is that the basin deposit layer settles less over time, creating a more narrow range in output values.

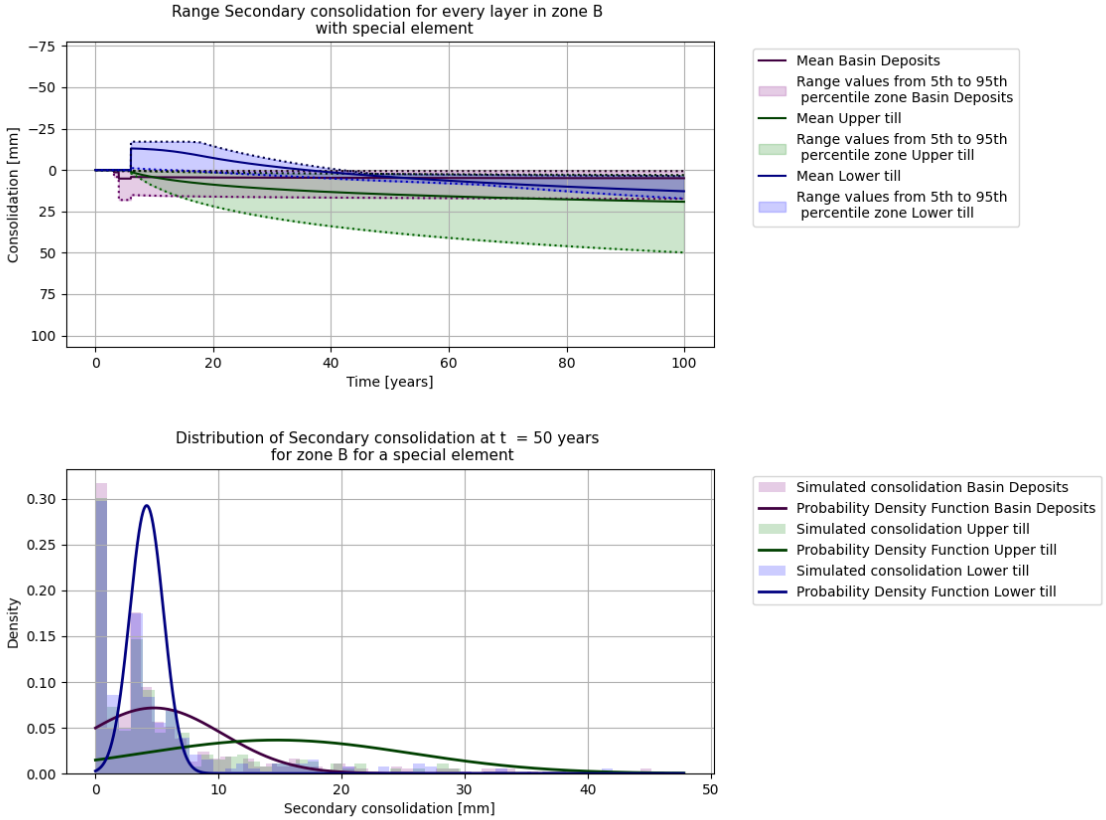


Figure 6.17: Variability secondary consolidation special element zone B

The variations in secondary consolidation at the end of the graph (100 years) for layers in zone B are as follows:

Table 6.6: Variation in secondary consolidation at years days for soil layers in zone B

Soil type	**Zone B**	
	Regular	***Special***
Meltwater sand	-	-
Chalk	-	-
Upper till	between 3.8 and 58.0 mm	between 3.0 and 49.9 mm
Lower till	between 3.1 and 20.1 mm	between 3.6 and 17.2 mm
Basin deposits	between 5.5 and 85.7 mm	between 0.6 and 17.4 mm
Palaeogene clay	-	-

Zone C

The variability of the primary consolidation of the separate layers in the soil composition of zone C is shown in figure 6.18.

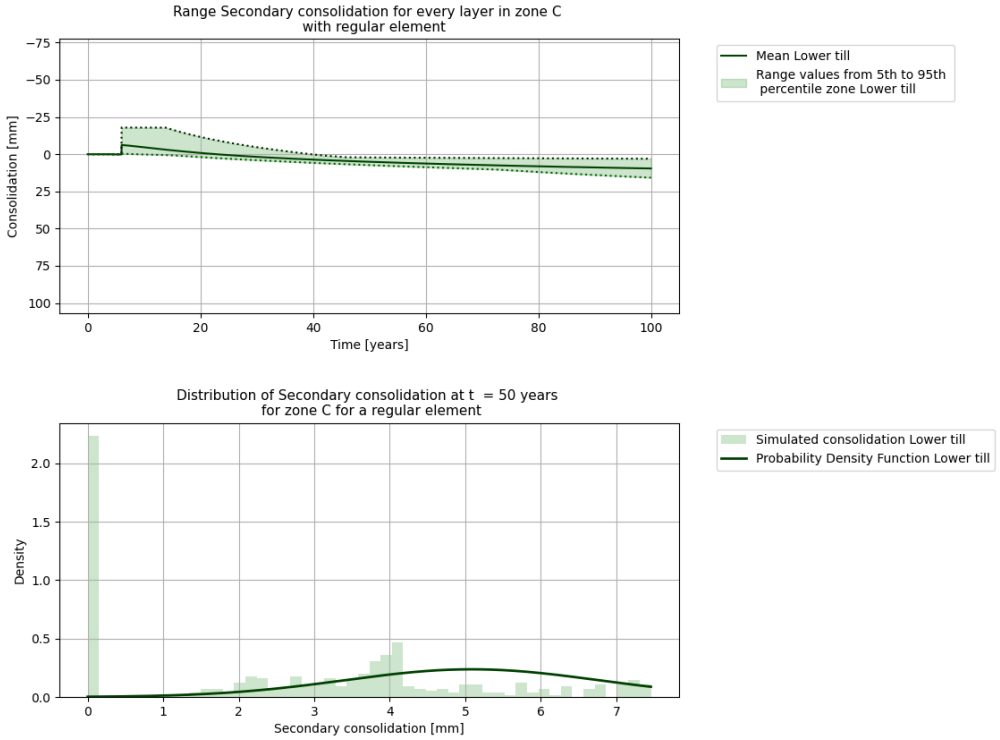


Figure 6.18: Variability secondary consolidation separate layers zone C

From these results, it stands out that the range in variation for the lower till layer decreases up to 50 years, and then starts to increase again with time. The range and mean in the upper till layer were insignificantly small, so is not present in the figure.

The distribution of the output values of the lower till layer follows a truncated Gaussian distribution except for the large peak at zero. The range in distribution is quite wide, indicating that this layer is sensitive to the uncertainty of the input value.

The variations in secondary consolidation at the end of the graph (100 years) for layers in zone C are as follows:

Table 6.7: Variation in secondary consolidation at years days for soil layers in zone C

Soil type	**Zone C**	
	Regular	***Special***
Meltwater sand	-	-
Chalk	-	-
Upper till	between 2.2 and 4.4 mm	-
Lower till	between 3.0 and 15.8 mm	-
Basin deposits	-	-
Palaeogene clay	-	-

Zone D
The variability of the secondary consolidation of the separate layers in the soil composition of zone D is shown in figures 6.19 and 6.20.

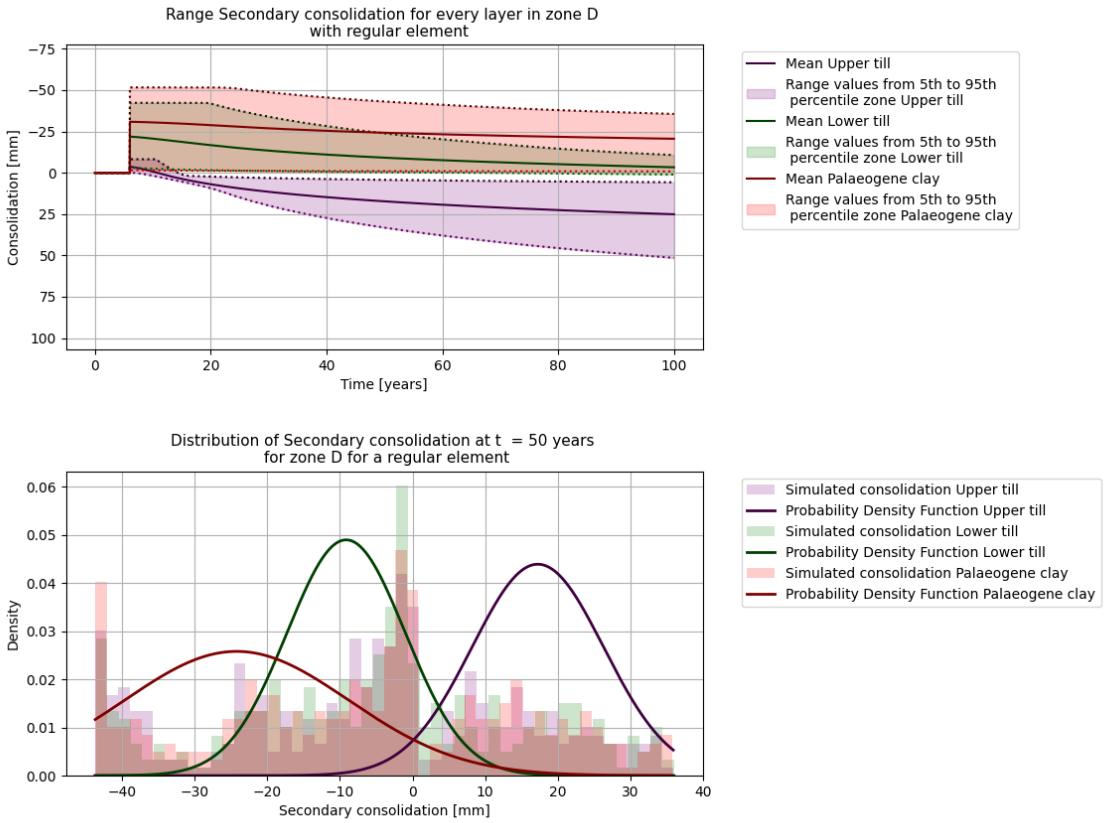


Figure 6.19: Variability secondary consolidation regular element zone D

From these results, it stands out that the range in variation of the Palaeogene clay remains consistently high over time. The range of variation for the lower till decreases with time, while the range in variation for the upper till layer increases significantly with time. The distribution of the Palaeogene clay layer stands out from the graph because it is very sensitive to the uncertainty of the input, shown by its very broad range in output distribution values. This means that this layer could pose a risk to the design of the tunnel foundation with its broad safety margins.

Additionally, with respect to the lower till layer and the Palaeogene clay layer, there is little to no difference between the regular and special element cases. However, the upper till layer is more robust in the special element case, making it less sensitive to the variation of the input values and acting as a stabilizer of these uncertainties.

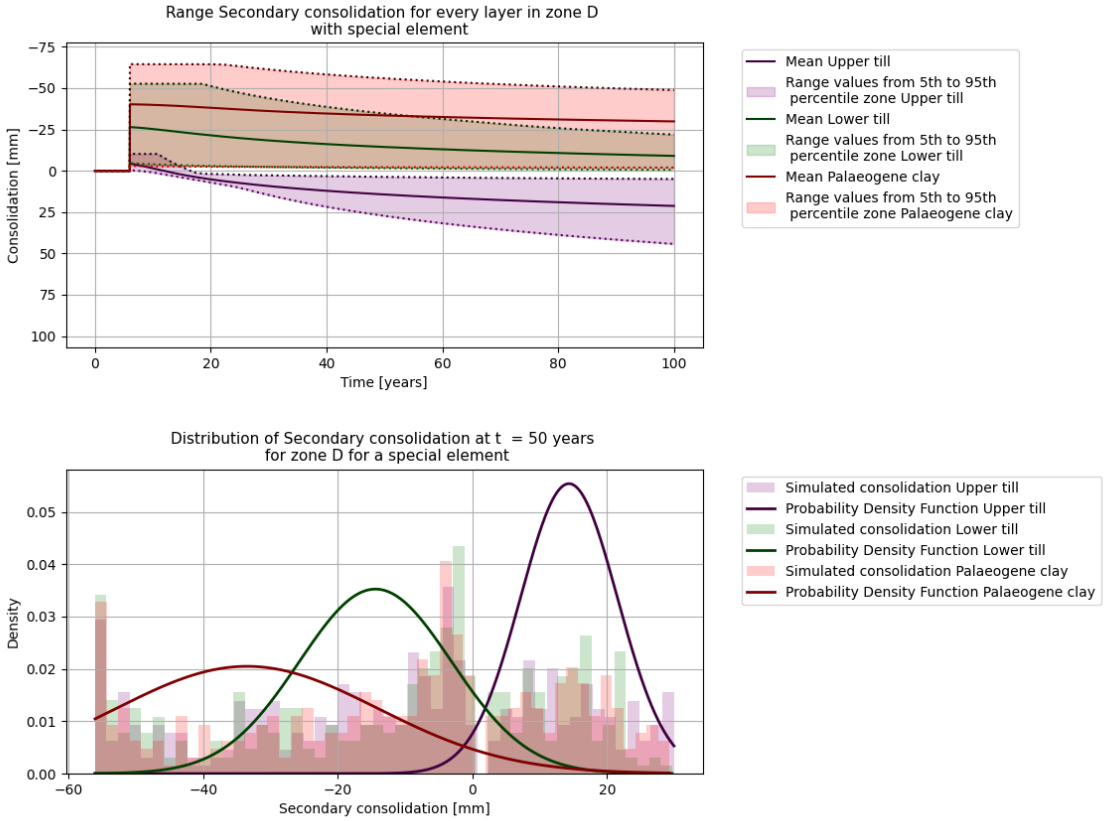


Figure 6.20: Variability secondary consolidation special element zone D

The variations in secondary consolidation at the end of the graph (100 years) for layers in zone D are as follows:

Table 6.8: Variation in secondary consolidation at years days for soil layers in zone D

Soil type	**Zone D**	
	Regular	***Special***
Meltwater sand	-	-
Chalk	-	-
Upper till	between 5.7 and 51.5 mm	between 4.9 and 44.3 mm
Lower till	between -10.8 and 1.0 mm	between -0.6 and-21.9 mm
Basin deposits	-	-
Palaeogene clay	between - 0.8 and - 35.6 mm	between - 1.9 and - 48.9

Zone E
The variability of the secondary consolidation of the separate layers in the soil composition of zone E is shown in figures 6.21 and 6.22.

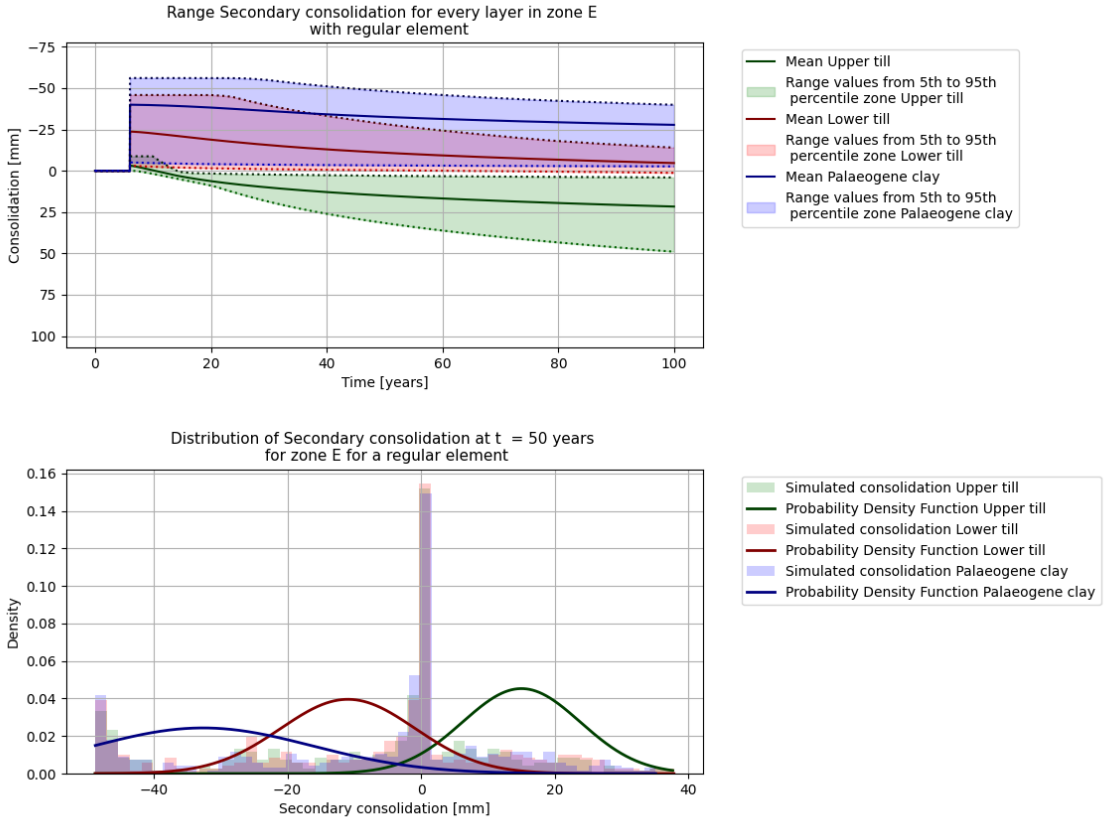


Figure 6.21: Variability secondary consolidation regular element zone E

From these results, it stands out that the range in variation for the upper till increases significantly with time, while the range in variation for the Palaeogene clay layer decreases slightly over time. Similarly, the range of variation for the lower till layer decreases greatly with time. There is no substantial difference in variation between the special element and the regular element case. Interestingly, Zone D and Zone E for the special element have the same soil composition and this is also shown in their variations and means of the results.

From the distributions, it can be concluded that the upper till layer and lower till layer are still quite robust from input values to output values, relatively to the other soil types they do not create a very broad range. The Palaeogene clay layer does create a very broad range, which means that it is very sensitive to the uncertainty in input values.

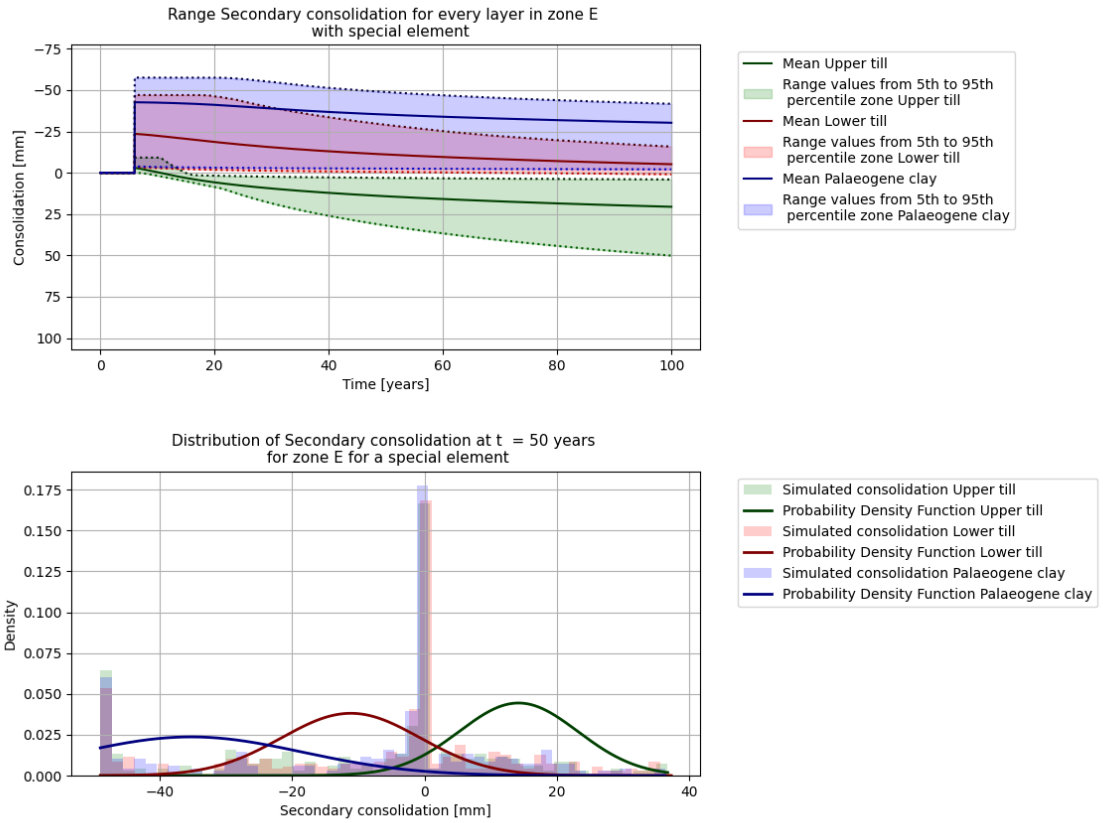


Figure 6.22: Variability secondary consolidation special element zone E

The variations in secondary consolidation at the end of the graph (100 years) for layers in zone E are as follows:

Table 6.9: Variation in secondary consolidation at years days for soil layers in zone E

Soil type	**Zone E**	
	Regular	***Special***
Meltwater sand	-	-
Chalk	-	-
Upper till	between 4.0 and 48.9 mm	between 4.0 and 50.1 mm
Lower till	between - 14.0 and 1.2 mm	between -15.9 and 0.9 mm
Basin deposits	-	-
Palaeogene clay	between -2.7 and -39.9 mm	betweeen - 2.1 and -41.7 mm

6.3.2. Variability secondary consolidation for all zones

Combining the results of individual soil layers provides a comprehensive view of the variability in secondary consolidation for each zone. This subsection presents the overall variability in secondary consolidation, highlighting the cumulative impact of soil thickness variability on long-term deformation.

The previous section 6.3.1 described every zone separately and discussed the variation of the soil layers present within the soil composition per zone. This section gives an overview of the variation of the total primary consolidation per zone, so an addition of their separate layers. The overview is visible in Figure 6.23.

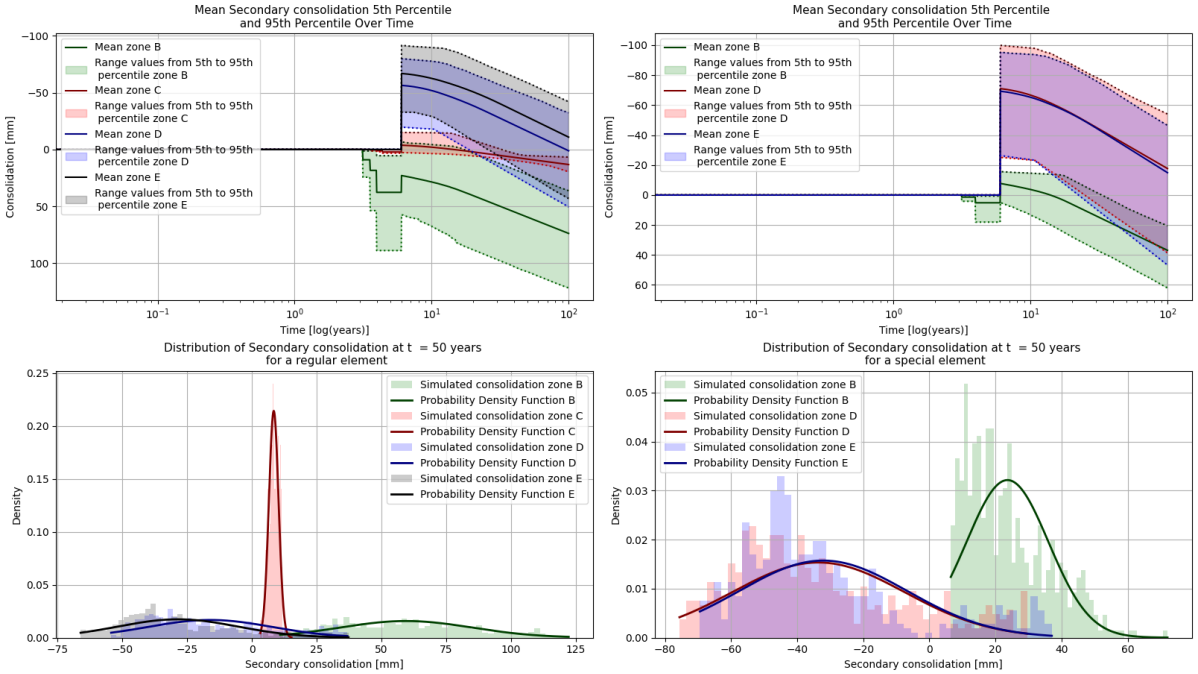


Figure 6.23: Variability of secondary consolidation per defined zone

From this overview, it is interesting to note that there are zones where a standard deviation of 1 meter could cause a difference between the settling and heaving of the soil, as can be seen from the distributions. These are zones D and E. The range in output values in zone B for a regular element is quite high, while the range in output values for a special element in the same zone is much narrower, indicating that the special element cases are more robust and less susceptible to large uncertainties in input values, at least in zone B. In the worst case, the difference in secondary consolidation will be 12.5 centimeters (settlement) for a regular element in zone B and minus 7.5 centimeters (heave) for a special element in zone D.

This leads to a variation per zone of the following:

Table 6.10: Effect on the range of secondary consolidation due to the change in soil thickness

Secondary Consolidation	Element	Range in values at 100 years
Zone A	***Regular***	-
	Special	-
Zone B	***Regular***	between 36.3 and 122 mm
	Special	between 20.5 and 62.1 mm
Zone C	***Regular***	between 6.7 and 19.6 mm
Zone D	***Regular***	between -32.3 and 50.6 mm
	Special	between -54.0 and 38.6 mm
Zone E	***Regular***	between -42.1 and 43.2 mm
	Special	between - 46.5 and 46.7 mm

6.4. Variability of results of complete deformation profile for 500 simulations

Understanding the overall variability in deformation across the entire tunnel profile is crucial for assessing its stability and serviceability. This section presents the variability results for the complete deformation profile, considering both regular and special elements. It provides a detailed analysis

of deformation patterns over specific time intervals, emphasizing the importance of soil thickness variability in predicting tunnel performance.

6.4.1. Variability of complete deformation over the entire length of the tunnel

This subsection presents the variability in total deformation across the entire length of the tunnel for different time intervals. It discusses the impact of soil thickness variability on the overall deformation profile, highlighting the differences in deformation behavior between regular and special elements.

Adding all the deformations together and coupling of the elements lead to a total variation over the entire length of the tunnel per zone. This was done for 6 different times (*5 years, 10 years, 20 years, 25 years, 50 years, and 100 years*) as explained before in Section 5.4. The results are analyzed in this section.

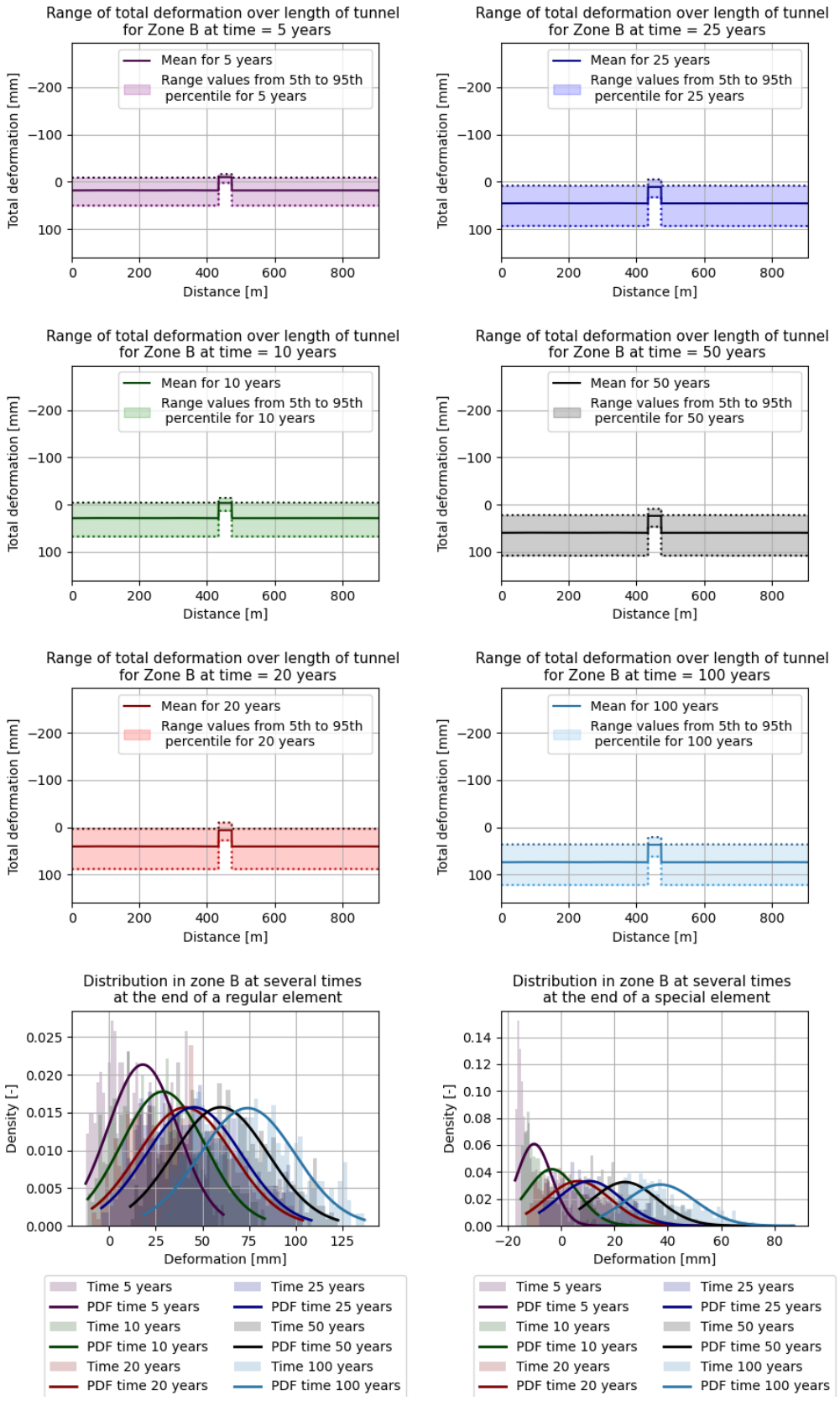


Figure 6.24: Total deformation over the length of the tunnel including variation range for zone B

Figure 6.24 illustrates the range of variation for zone B, with the x-axis representing the distance from the start to the end of the zone and the y-axis showing the variation in meters.

Notably, the variations at the regular elements are significantly greater than at the special elements. During this period, the variation at the special element fluctuates between settling for 1.4 centimeters and heaving for 1.5 centimeters, which could pose a design challenge. At other times, the variation at the regular elements is much greater than at the special element. The range in case of the regular elements increases with time, whereas the range in case of the special elements also grows, but only during the secondary consolidation phase. The distributions of the regular element case all follow a Gaussian distribution that increases its range over time. This indicates that the effect of the uncertainty of the input values also increases over time.

Summarized:

Table 6.11: Variation in total deformation for different times and per element type at the beginning of the element in zone B

Zone B	***Regular***	***Special***
5 years	between 2.9 and 33.2 mm	between - 37.9 mm and 17.7 mm
10 years	between 21.7 and 36.0 mm	between - 27.4 and 20.5 mm
20 years	between 35.5 and 46.1 mm	between - 10.6 and 23.3 mm
25 years	between 32.7 and 58.1 mm	between 1.3 and 20.0 mm
50 years	between 23.4 and 96.2 mm	between 10.0 and 37.9 mm
100 years	between 14.5 and 133.9 mm	between 3.4 and 71.0 mm

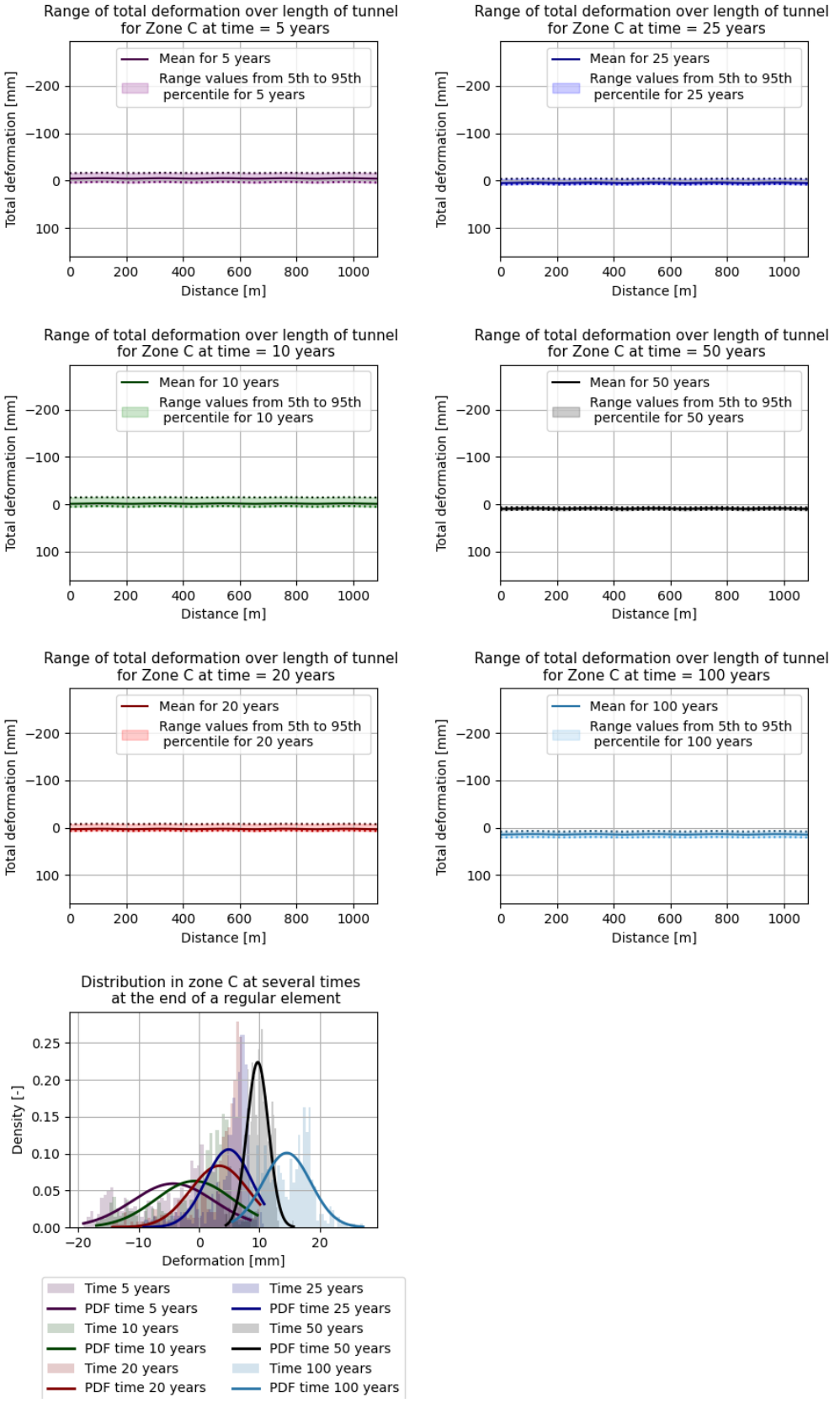


Figure 6.25: Total deformation over the length of the tunnel including variation range for zone C

Figure 6.25 illustrates the range of variation for zone C, with the x-axis representing the distance from the start to the end of the zone and the y-axis showing the variation in meters.

Notably, the range over the distance in the zone differs a lot and shifts from heaving to settling. At the earlier times it has a broad range, which decreases over time up until the 50 years, from that point the range starts to increase again. This could pose a problem in the prediction of heaving and settling of regular elements in this zone. The maximum relative range in values occurs at the earlier years (5 and 10), during the primary and secondary consolidation phase respectively.

Summarized:

Table 6.12: Variation in total deformation for different times and per element type at the beginning of the element in zone C

Zone C	***Regular***	***Special***
5 years	between - 31.8 and 23.3 mm	-
10 years	between -25.1 and 23.5 mm	-
20 years	between -10.2 and 16.9 mm	-
25 years	between -2.4 and 12.2 mm	-
50 years	between 0.8 and 18.6 mm	-
100 years	between 3.4 and 25.6 mm	-

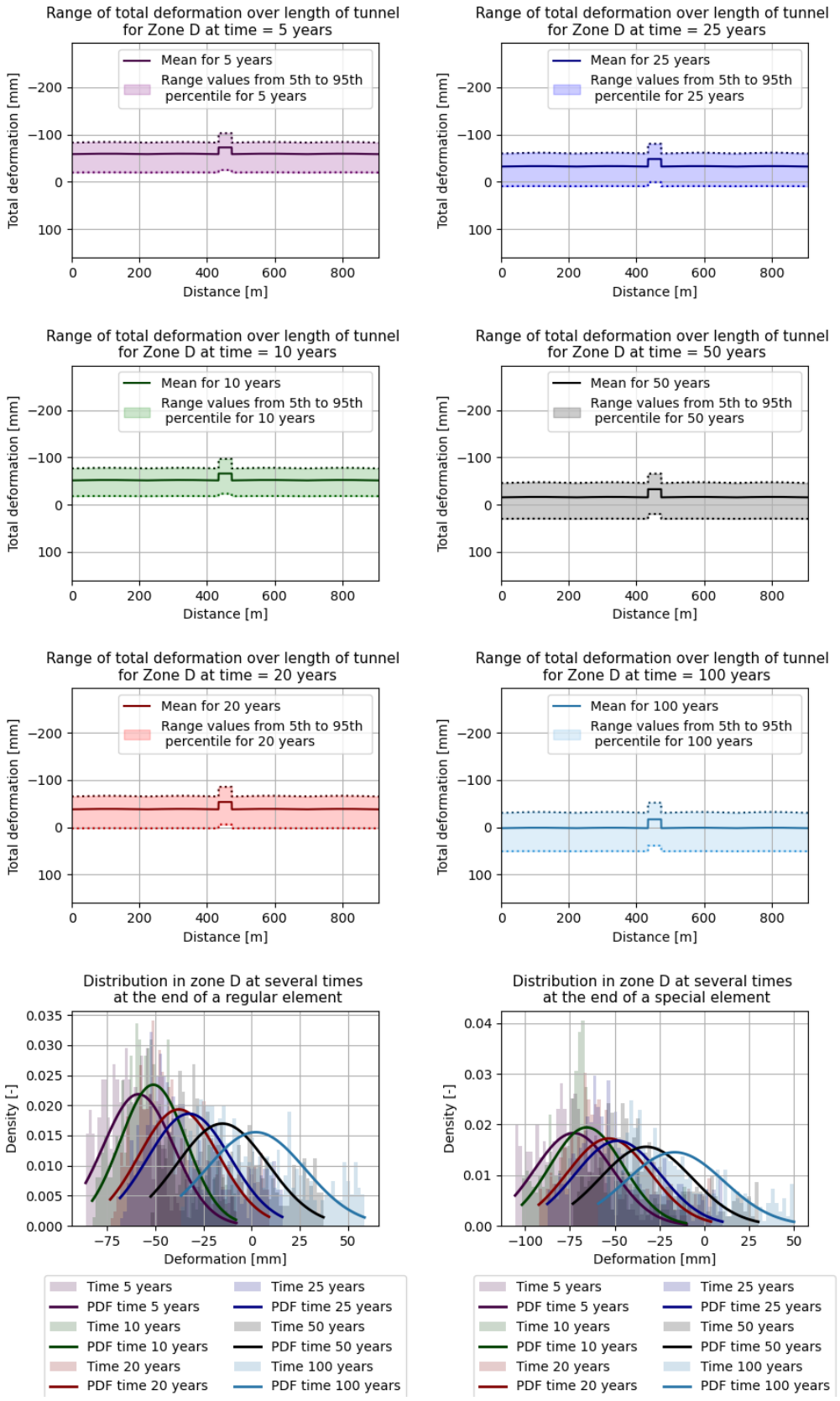


Figure 6.26: Total deformation over the length of the tunnel including variation range for zone D

Figure 6.26 illustrates the range of variation for zone D, with the x-axis representing the distance from the start to the end of the zone and the y-axis showing the variation in meters. By comparing the ranges of the regular elements and special elements it stands out that they are quite alike. The special element has a broader range and a lower peak, meaning it is more susceptible to uncertainty of the input values. Another point of interest from the figure is the fact that the range of the distributions grow over time, increasing the uncertainty and risks.

Summarized:

Table 6.13: Variation in total deformation for different times and per element type at the beginning of the element in zone D

Zone D	***Regular***	***Special***
5 years	between - 197.7 and 80.2 mm	between -245.4 and 99.6 mm
10 years	between - 179.0 and 76.9 mm	between -227.1 and 95.8 mm
20 years	between -147.2 and 71.6 mm	between - 196.8 and 90.2 mm
25 years	between -134.4 and 69.4 mm	between - 183.9 and 87.1 mm
50 years	between -93.2 and 62.8 mm	between -143.3 and 78.4 mm
100 years	between -51.0 and 55.1 mm	between - 105.5 and 72.4 mm

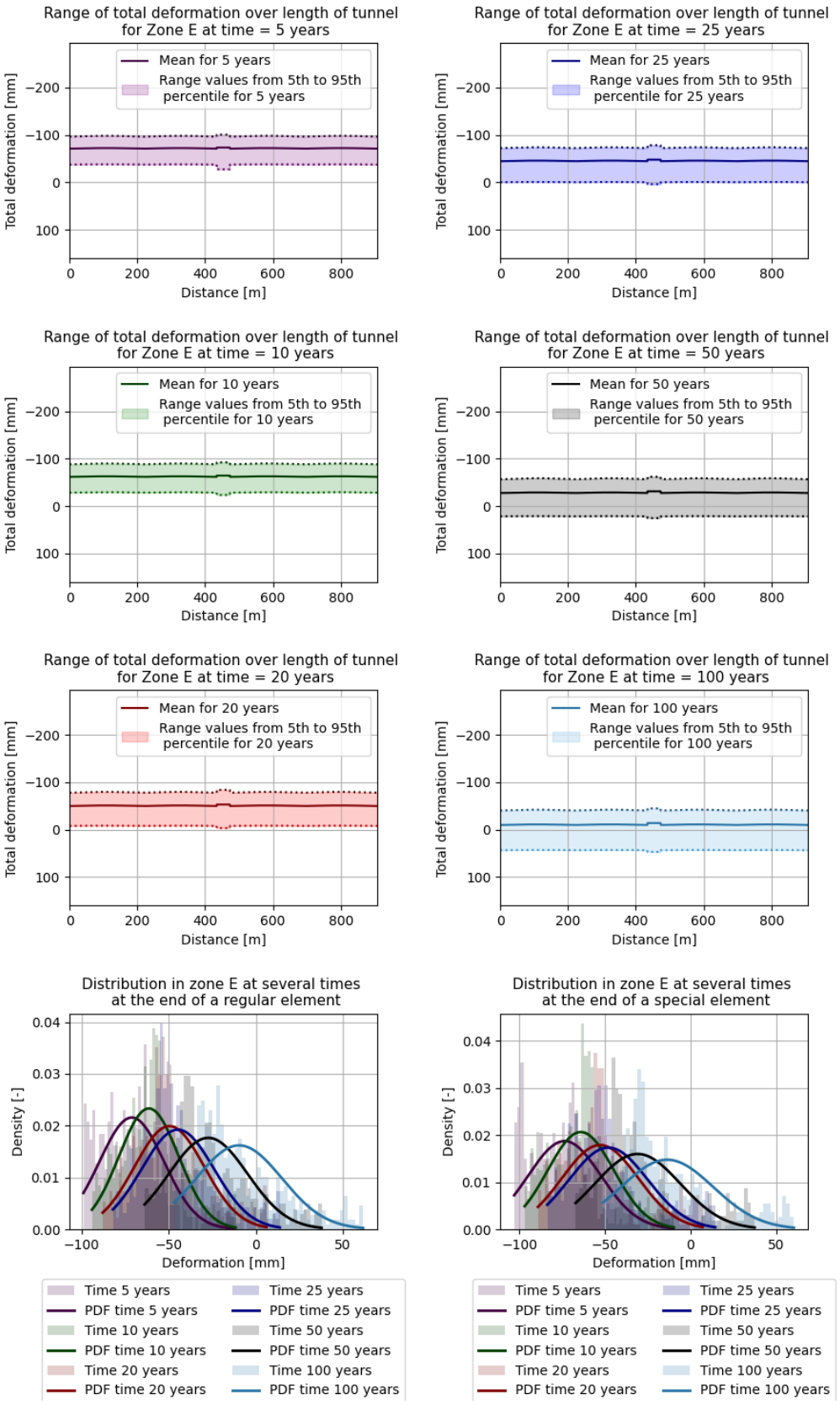


Figure 6.27: Total deformation over the length of the tunnel including variation range for zone E

Figure 6.27 illustrates the range of variation for zone E, with the x-axis representing the distance from the start to the end of the zone and the y-axis showing the variation in meters. The first thing that stands out from the results is that the range in output values for the special element is larger than for the regular element, which was not the case at the other zones. When comparing the regular and special elements, it becomes evident that their distributions are quite similar. However, the special element exhibits a wider range and a lower peak, indicating a greater sensitivity to input uncertainty. In addition, the figure highlights that the distribution ranges expand over time, reflecting an increase in uncertainty and associated risks.

Summarized:

Table 6.14: Variation in total deformation for different times and per element type at the beginning of the element in zone E

Zone E	***Regular***	***Special***
5 years	between -233.1 and 90.7 mm	between -242.2 and 95.1 mm
10 years	between -209.1 and 86.3 mm	between -218.5 and 90.3 mm
20 years	between -180.6 and 81.4 mm	between -193.6 and 87.9 mm
25 years	between -167.0 and 77.5 mm	between -180.3 and 84.3 mm
50 years	between -124.2 and 69.5 mm	between - 135.3 and 73.4
100 years	between - 78.9 and 59.6 mm	between -90.2 and 62.8 mm

6.5. Results due to a change in standard deviation

The hypothesis is that the relative range of deformation values changes due to a change in standard deviation in the thickness of the soil layers. The standard deviations chosen to prove or disprove this are *0.5 meter, 1 meter, 2 meter, and 5 meter*. A standard deviation of ten meters was also tested, but the model could not handle a variation of 10 meters within a framework of about 80 meters of soil, especially when the soil composition consists of more than three layers and the soil layer thickness cannot be zero. The results of the change in standard deviation of the thickness of the soil layers for the range of total consolidation at 100 years are shown in Table 6.15. The resulting figures for all the different standard deviations can be found in Appendix C.

Table 6.15: Effect on the range of total consolidation at 100 years due to the change in standard deviation of soil thickness

Total consolidation	**Standard deviation**				
	Element type	**0.5 meter**	**1 meter**	**2 meter**	**5 meter**
Zone B	*Regular*	between 48.1 and 97.9 mm	between 36.3 and 122 mm	between 26.5 and 129.1 mm	between 24.5 and 133.7 mm
	Special	between 25.4 and 54.0 mm	between 20.5 and 62.1 mm	between 18.5 and 118.9 mm	between 18.8 and 147.5 mm
Zone C	*Regular*	between 9.4 and 17.5 mm	between 6.7 and 19.6 mm	between 5.2 and 34.3 mm	between 3.9 and 50.1 mm
Zone D	*Regular*	between -22.2 and 46.2 mm	between -32.3 and 50.6 mm	between -37.9 and 47.1 mm	between -38.3 and 32.0 mm
	Special	between -43.4 and 33.8 mm	between -54.0 and 38.6 mm	between -55.7 and 34.3 mm	between -54.9 and 22.0 mm
Zone E	*Regular*	between -34.8 and 20.4 mm	between -42.1 and 43.2 mm	between -44.8 and 44.6 mm	between -43.4 and 37.7 mm
	Special	between -40.4 and 9 mm	between - 46.5 and 46.7 mm	between -46.1 and 44.2 mm	between -46.0 and 24.4 mm

The important conclusions from these results are as follows. The range in total consolidation values **increases** (even significantly for zones B and E) with **an increase in the standard deviation** from 0.5 to 1 meter.

The most interesting conclusion that can be drawn from Table 6.15 is that the range of results for the final deformation at 100 years **decreases** with an i**ncrease in the standard deviation** of the thickness of soil

layers (starting at a standard deviation of 1 meter in the thickness of the soil layers) present at the tunnel parts in Zone D. The same happens in Zone E, but starts from a standard deviation in soil layer thickness of 5 meter. This could mean that there is a certain threshold in the model from which point the behavior of the model becomes less affected by the input values, or due to the non-linear behavior of the model it dampens extreme input variations, possibly due to physical constraints, averaging effects, or dominant mechanisms that override input noise. The threshold that can cause this effect is the fact that the last layer in the model will have the thickness that is left in the soil column during generation of this soil column, if the standard deviation becomes bigger, the thickness that is left in the soil column becomes smaller, and the last layer will be thinner and has less influence to the range in the total deformation.

The model used for primary consolidation is known to incorporate **non-linear soil behavior** and **stress history effects**. If the input variability increases (e.g., in layer thickness or load), the model might:

- **Dampen the response** due to nonlinear stiffness or preconsolidation effects.
- Show **reduced sensitivity** in overconsolidated soils or soils with high stiffness.

This could explain why the output standard deviation decreases even as input variability increases, especially if the soil layers are reaching a **saturation point** in their compressibility.

The model used for secondary consolidation is dependent on time. This means:

- Once primary consolidation is complete, secondary consolidation progresses at a rate that is **less influenced** by input variability such as thickness or load.
- If input variability affects only the **initial conditions**, the long-term secondary consolidation could converge to a **narrower range**.

Regarding the different types of soils:

- Certain soil layers (e.g., stiff clays or overconsolidated tills) can **dominate the response**, masking variability from other inputs.

Overall, the zones that exhibit the largest range in total consolidation at 100 years is Zone B (ranges up to 110 mm) followed by Zone D and Zone E (ranges up to 85 mm).

6.6. Important takeaways from the results

This section summarizes the key findings from the variability analysis, discussing their implications for tunnel design and construction. This section highlights the most significant observations, emphasizing the importance of considering soil variability in geotechnical analysis. The section provides recommendations for future research and practical applications, ensuring that the insights gained from this study contribute to more resilient and reliable tunnel structures.

Some of the conclusions presented in this chapter are repetitive and come back in every zone or consolidation stage. The following takeaways are important from this chapter:

Disregarded soil layers and zones
Zone A has no variation in the thickness of soil layer, so it is only used as a comparison for results and to check if the Python code worked correctly. Furthermore, the meltwater sand layer and the chalk layer do not consolidate and only contribute to the initial deformation.

Influence of initial deformation
The variation in initial deformation in the middle of a regular element is close to zero or even zero (except for Zone C). The shape in the deformation profile is present due to the boundary conditions chosen. The boundary conditions at the ends of each tunnel element are modeled as roller supports, and this means that vertical displacement is allowed but rotations are fixed. This setup creates a localized stiffness effect at the ends of the beam, which artificially increases the resistance to deformation near the supports. As a result, the middle of the beam appears more flexible, and the ends appear more anchored, even though, in reality, the tunnel is continuous.

If the entire tunnel is modeled as a continuous beam, the boundary conditions change significantly and the continuity of displacement and rotation between elements becomes critical, the stiffness of the beam will be distributed over a much longer length and the influence of individual element stiffness

will become less dominant. This would lead to a smoother deformation profile along the length of the tunnel, a reduction of artificial stiffness effects at the ends of the elements, and a more realistic representation of how the tunnel interacts with the soil, especially in zones with gradual transitions in soil properties.

However, the beam stiffness is still an important factor in the initial deformation, but its relative influence decreases in the single-element model here the beam stiffness dominates because the element is short and isolated even very much in the special element case, due to the fact that this element is much shorter than the regular element.

In Zone C, the distribution of results follows a truncated Gaussian distribution rather than the doubly truncated lognormal distribution used as input. This shift may be due to the average stiffness of the soil, with the dominant chalk layer strongly influencing this value. In addition, the special element shows a wider range of results than the regular element, indicating a greater sensitivity to input variability. In particular, the initial deformation range in the special element in Zone E is the largest among all zones, suggesting a higher risk to the design. However, it is important to note that the initial deformation ranges from 0 to -2.2 mm, while secondary consolidation can reach up to 85 mm, making the impact of the initial deformation relatively minor. *Variation in soil layer thickness does not play an important role in the initial deformation, due to the insignificant load difference.*

Influence of primary consolidation
The variation in primary consolidation is lower for thin soil layers compared to thicker ones across several zones. This suggests that thinner layers are less sensitive to input variability, whereas thicker layers are more affected. In the case of the special element, the peaks for the lower till and basin deposit layers are more pronounced, indicating increased sensitivity to input uncertainty when these layers experience significant unloading. *Thicker layers need a more accurate thickness value if more certainty is needed for the amount of primary consolidation.*

The lower till layer shows a wide range of output values for primary consolidation. This can be attributed to its lower coefficient of consolidation compared to the upper till layer. A similar pattern is observed in the Palaeogene clay layer in all zones. Although the Basin Deposits layer has a comparable coefficient of consolidation, it behaves differently due to its significantly lower oedometer stiffness. The upper till layer exhibits varying peaks in all zones, suggesting that the range of primary consolidation increases with the magnitude of the consolidation itself, which is a trend that is consistent in all types of soil.*The combination of a high coefficient of consolidation and high oedometer stiffness causes a wider range of output values for primary consolidation.*

An overview of primary consolidation across all zones reveals that the unloading phase amplifies uncertainty in model results, while the loading phase tends to dampen it.

Influence of secondary consolidation
For secondary consolidation, the Basin Deposits layer shows a wide range of deformation values over time, although this range does not increase with time. This layer consistently exhibits the broadest range in deformation, likely due to its low Overconsolidation Ratio (OCR), highlighting the significant role OCR plays in both primary and secondary consolidation variability.

In Zone B, the variation in the lower till layer decreases over time (until 45 years) and then begins to increase again. Interestingly, the distribution narrows significantly at 50 years, contrasting with the broader range seen during the primary consolidation phase. This suggests that the lower till layer becomes more robust around 45 years, but less so afterward, making this a critical time point for tunnel performance evaluation. In Zone B and Zone D, the lower and upper till layers show markedly different ranges, probably due to the presence of the meltwater sand layer, which alters the drainage paths and the onset of secondary consolidation.

The Palaeogene clay layer demonstrates a broad range of output values during the secondary consolidation phase, indicating high sensitivity to input uncertainties. *So, it is advised to have a more certain thickness of the Palaeogene clay and lower till layer in zone D and zone E.*

It should be noted that in some zones, a standard deviation of just 1 meter in input parameters can change the soil behavior from settling to heaving, as observed in the distributions for Zones D and E. *Zones D and E need a more certain layer thickness to determine whether the soil will heave or settle over time.*

In Zone B, the secondary consolidation output range for a regular element is quite large, while the range for a special element is narrower. This suggests that special elements may be more robust and less sensitive to input variability in this zone. However, the opposite is true in Zones D and E, where special elements show greater variability.

Influence total deformation
The total deformation range over time increases for tunnel sections in Zone B, decreases in Zones D and E, and follows a more complex pattern in Zone C, where it initially decreases and then starts to increase again from a certain point in time. In Zone C, the maximum relative range in the deformation values occurs in earlier years. *There could be a certain point in time that operates as a turning point for the variability changing from decrease in range to increase in range.*

Change in standard deviation
Zone B exhibits the largest range in total consolidation at 100 years. This range even increases with an increase in the standard deviation of the input, *making this a zone of interest for a better indication of the thicknesses of the soil layer*. Zone C exhibits the narrowest range in output values, but does increase with an increase in standard deviation, so could pose a problem after 100 years of operation, the side note to place here is that this is so far in the future, that it does not need to be take into account.

The primary consolidation model tends to dampen the system's response due to the effects of nonlinear stiffness and preconsolidation. It also exhibits lower sensitivity in soils that are overconsolidated or possess high stiffness.

In the case of secondary consolidation, the model evolves over time in a way that is less affected by variations in inputs such as layer thickness or applied load. When input variability primarily influences the initial conditions, long-term deformation behavior tends to stabilize within a narrower range.

Furthermore, *overconsolidated cohesive soils often play a dominant role in the behavior of the system, effectively suppressing the influence of variability in other input parameters.*

7

Conclusion and recommendations

This chapter summarizes the key findings of the study, highlighting the impact of soil thickness variability on secondary soil consolidation beneath the elements of the Fehmarnbelt Tunnel. It provides a comprehensive overview of the conclusions drawn from the analysis and offers recommendations for future research and practical applications. The insights gained from this study contribute to a better understanding of the behavior of the soil beneath immersed tunnels and inform more accurate predictions and designs in future projects.

7.1. Conclusion

This thesis aimed to investigate the impact of the variability in soil layer thickness of overconsolidated clays on the secondary consolidation of soil beneath immersed tunnel elements and did so by investigating four main topics.

The analysis revealed that the variation in the initial deformation does not play an important role in the complete system of deformation. The range in initial deformation is a factor 10^{-1} to 10^{-3} lower than the ranges for primary and secondary consolidation, making its contribution insignificant in total deformation. This insignificance is due to the fact that a small load difference causes a low deformation. However, this would change if the boundary conditions were changed, if rotations are allowed, and the complete tunnel was taken into account. This would cause the deformation to be more evenly distributed per element, the influence of the beam stiffness would be diluted by the continuity of the structure and the initial deformation profile would likely be flatter and more realistic.

The impact of soil thickness variability on primary consolidation was found to be substantial. The study demonstrated that the effect is great in the soil unloading phase and increases and from that point the range narrows down over time. This indicates that the unloading phase amplifies the uncertainty in the results, and the loading phase dampens this uncertainty. Furthermore, a high coefficient of consolidation combined with an elevated oedometer stiffness leads to a wider range of primary consolidation outcomes, as well as thicker soil layers. So, more certainty is needed in the thickness of the lower till layer for the primary consolidation.

The analysis of secondary consolidation highlighted the importance of accuracy in soil thickness measurements, especially for zone B and zone E. The first thing to note was that zone E exhibits the largest heave value during secondary consolidation, and zone B exhibits the largest settlement value during secondary consolidation. The problem in zone B is the basin deposits layer that gives high mean values for settlements and causes a wide range of variations on the resulting values; therefore, with the removal of this layer or a more accurate layer thickness, this problem will be resolved. The second thing to note here was that zone D has a very broad range in heave results too, something that could be limited by investigating the soil layer thicknesses more accurately here and at zone E. In general, variations in soil thickness in soil columns where all soil layers are cohesive, were shown to have a significant impact on the secondary consolidation process, emphasizing the need for an accurate estimation of soil parameters and thickness.

The analysis in this thesis ended with an indication of what the change in the standard deviation of the soil layer thicknesses would do with the range in the resulting deformations per zone. These results were eye-opening and showed that by even a change of 0.5 meters in the standard deviation of the thickness, the range in the resulting deformation values increased greatly, from 97.9 mm more settlement to 122 mm more settlement in Zone B or from 34.8 mm of heave to 42.1 mm of heave in Zone D. However, when the standard deviation in the thickness of soil layers increases from 1 to 2 meters or even to 5 meters, the soils reach a certain threshold from where the range in output values starts to decrease, probably caused by the upper threshold in the lognormal distribution of the input values.

These results and the conclusion of the thesis together answer the research question stated in the introduction: *What is the effect of soil thickness variability on secondary consolidation along the alignment of the Fehmarnbelt Immersed Tunnel Project, and what method can be employed to characterize and quantify this?* The TBKF model, the model of Conte & Troncone (2006) and the model of Feng et al. (2020) provided the framework to quantify the range of secondary consolidation values for the five assigned specific tunnel zones. In general, this research highlights the importance of considering soil variability in the design and construction of immersed tunnels and shows that with a standard deviation of 0.5 meters in randomly generated thicknesses of the soil layers present can result in a range from 48 millimeters to 98 millimeters of settlement for regular elements in Zone B or even a range from 43 millimeters of heave to 34 millimeters of settlement for special elements in Zone D.

By addressing the soil regions and layers that are very susceptible to a wide range of variation in the resulting secondary consolidation, engineers can take measures improve the durability and safety of tunnel structures, contributing to more resilient infrastructure.

7.2. Recommendations and discussion

In this research, some fundamental assumptions were made based on knowledge and the models used. These assumptions are as follows.

The results could differ when the whole tunnel was modeled instead of zones or separate elements. First of all the change between a regular element and a special element on this x-axis is very abrupt, and the deformation does not gradually increase or decrease from a regular element to a special element or the other way around. In reality, this should be the case, but falls outside the scope of this thesis. Secondly, the interaction between individual parts of the tunnel was not considered. Consequently, the effects of tunnel elements rotating relative to each other or the effects of Gina caskets that prevent rotation and deformation were not present. The mean of deformation would be different if each element rotates relative to another and deformations of the sides of an element are influencing the neighboring elements. Lastly, modeling the complete tunnel instead of parts could lead to a more uniform distribution of deformation across each element, as the continuous nature of the tunnel structure would reduce the localized influence of beam stiffness. As a result, the initial deformation profile would appear smoother and more representative of actual behavior. The interaction between zones will also play a role in an analysis of the complete structure where the (abrupt) differences between soil compositions could lead to amplification or dampening of deformations.

As seen in this thesis, the special element reacts very differently than the regular element, and it is recommended to delve into this difference to sharpen the knowledge about the elements reacting to each other and the behavior of soil that is close to the special element and under the regular element. The cross section of the tunnel elements over their length was kept constant to satisfy the conditions for a Timoshenko beam model. However, this may differ in reality, increasing or decreasing the initial deformation. Furthermore, the contact between the beam and the soil was considered bilateral in the TBKF model, so the normal forces in the beam were ignored.

Linear elastic behavior was assumed with the TBKF model. This assumption holds for the tunnel itself, as it does not experience forces high enough to reach its plastic deformation point. However, it is not entirely accurate for the soil beneath the tunnel. Most cohesive soils are highly overconsolidated and do not reach their preconsolidation pressure, thus avoiding plastic deformation. However, the Palaeogene clay layer has a lower overconsolidation ratio and will partially deform plastic. Since it is present at the bottom of the soil column for zones C and D, and the load difference at the top is not as significant as with, for example, placing a building, the change is likely minimal. Furthermore, the TBKF model was

only used for the initial deformation of the soil, so there is no primary consolidation occurring in this phase. The plastic deformation of the layers was taken into account by the Conte and Troncone model (2006) and the Feng et al. model (2020).

This research did not provide a complete assessment of risks and failure mechanisms, as it did not account for tidal loads or fluctuations in groundwater levels. Cyclic loads, such as tidal loads and back siltation loads, were not considered because this would increase the scope of this thesis rather than decrease it. Implementing would cost a lot of extra time. It is advised to implement these loads in the existing model to give a more realistic representation of the deformation values. One thing to take into account is that the degradation of the secant shear modulus due to cyclic loading was not taken into account; although this often occurs in soft clays, the clays in the study area are much stiffer, meaning that the secant shear stiffness will change less due to cyclic loading, though it will still change. So, extra attention on this part of the model would be advised if cyclic loading is considered in future research.

The uncertainty of other soil parameters was not included in this research, but it could significantly influence the results due to the mathematical complexity of the models and the need for detailed parameter estimations. Without precise parameter estimations, the results could deviate considerably from what happens to the tunnel in reality. A parameter sensitivity analysis is required to understand the impact of errors in parameter estimation on the results. Moreover, in Section 2.2.1 a specific model was chosen to approach the equivalent soil stiffness for a whole soil column. The method chosen is a method that is used in academic research predominantly, but there are more accurate methods to determine the equivalent stiffness of the soil. Research in these methods is the next step to more accurately determine these equivalent soil parameters for the soil-structure interaction.

The distribution used in this research was a doubly truncated lognormal distribution with an upper limit of the remaining height in the soil column and a lower boundary of 0.1 meters of soil layer thickness. A limitation in this truncated distribution was the height of the soil column, this height acted as an upper limit for soil layers so that the combination of randomly generated thicknesses of the soil layers would not exceed the total height of the soil column; a new approach could be to remove this limitation and see how big the ranges in output value would grow with a higher standard deviation for the randomly generated input values. The mean of the randomly generated layer thicknesses is determined by other research (Yumpu.com 2011) and the standard deviation (of 0.5, 1, 2, 5 and 10 meters) of the randomly generate thickness of soil layers is the same for every soil type. In further research a subdivision in layers can be made where every layer has its own distribution and standard deviation to investigate the uncertainty and risk. This results in a specialized research that comes closer to the situation found in-situ.

It is important to note that the filling itself does not significantly alter the loading conditions, as the soil is first excavated and then replaced at the same location. The actual change in loading occurs only at the site where the tunnel segment is installed. As a result, the study focused solely on changes in soil thickness rather than conducting a complete 3D evaluation of the change in soil volume.

In addition, the depth of the trench was treated as a fixed value, determined by the requirements for tunnel placement. Variability in trench depth, such as potential inaccuracies introduced by excavation equipment, was not considered. This simplification presents an opportunity for future research, particularly in exploring how such variations could influence the overall behavior and reliability of the tunnel system.

References

't Hart, Cornelis Marcel Pieter, Oswaldo Morales-Nápoles, and Bas Jonkman (May 1, 2024). "The influence of spatial variation on the design of foundations of immersed tunnels: Advanced probabilistic analysis". In: *Tunnelling and Underground Space Technology* 147, p. 105624. ISSN: 0886-7798. DOI: 10.1016/j.tust.2024.105624. URL: https://www.sciencedirect.com/science/article/pii/S0886779824000427 (visited on 03/18/2024).

Abu-Farsakh, Murad Y., Qiming Chen, and Louisiana Transportation Research Center (Mar. 1, 2012). *Evaluation of the base/subgrade soil under repeated loading : phase II, in-box and ALF cyclic plate load tests [tech summary].* URL: https://rosap.ntl.bts.gov/view/dot/24052 (visited on 04/11/2025).

Andersen, Jørgen, Claus Iversen, and Eelco Van Putten (2012). "MARINE WORKS OPERATIONS AND ENVIRONMENTAL CONSIDERATIONS WHEN BUILDING THE FEHMARNBELT TUNNEL". In: *Terra et Aqua*.

Avramidis, I.E. and K. Morfidis (Jan. 2006). "Bending of beams on three-parameter elastic foundation". In: *International Journal of Solids and Structures* 43.2, pp. 357–375. ISSN: 00207683. DOI: 10.1016/j.ijsolstr.2005.03.033. URL: https://linkinghub.elsevier.com/retrieve/pii/S0020768305001459 (visited on 07/29/2024).

Barden, Laing (Sept. 1962). "Distribution of Contact Pressure Under Foundations". In: *Géotechnique* 12.3. Publisher: ICE Publishing, pp. 181–198. ISSN: 0016-8505. DOI: 10.1680/geot.1962.12.3.181. URL: https://www.icevirtuallibrary.com/doi/10.1680/geot.1962.12.3.181 (visited on 04/11/2025).

Barros, De and S. Thenn (1966). "DEFLECTION FACTOR CHARTS FOR TWO-AND THREE-LAYER ELASTIC SYSTEMS". In: *Highway Research Record* 145. URL: https://trid.trb.org/View/126729 (visited on 04/11/2025).

Been, K. and G. C. Sills (Dec. 1981). "Self-weight consolidation of soft soils: an experimental and theoretical study". In: *Géotechnique* 31.4. Publisher: ICE Publishing, pp. 519–535. ISSN: 0016-8505. DOI: 10.1680/geot.1981.31.4.519. URL: https://www.icevirtuallibrary.com/doi/abs/10.1680/geot.1981.31.4.519 (visited on 04/18/2025).

Bell, Frederic Gladstone (1992). *Engineering properties of soils and rocks*. 3rd ed. Oxford Boston: Butterworth-Heinemann. ISBN: 978-0-7506-0489-5.

Bjerrum, Laurits (June 1967). "Engineering Geology of Norwegian Normally-Consolidated Marine Clays as Related to Settlements of Buildings". In: *Géotechnique* 17.2. Publisher: ICE Publishing, pp. 83–118. ISSN: 0016-8505. DOI: 10.1680/geot.1967.17.2.83. URL: https://www.icevirtuallibrary.com/doi/10.1680/geot.1967.17.2.83 (visited on 04/20/2025).

Bowles, Joseph E. (1996). *Foundation analysis and design*. 5. ed., internat. ed. New York: McGraw-Hill. 1175 pp.

Buoyancy | History, Science, & Applications | Britannica (Mar. 28, 2025). URL: https://www.britannica.com/science/buoyancy (visited on 04/17/2025).

Chen, Ze-Jian, Wei-Qiang Feng, and Jian-Hua Yin (Oct. 2021). "A new simplified method for calculating short-term and long-term consolidation settlements of multi-layered soils considering creep limit". In: *Computers and Geotechnics* 138, p. 104324. ISSN: 0266352X. DOI: 10.1016/j.compgeo.2021.104324. URL: https://linkinghub.elsevier.com/retrieve/pii/S0266352X21003220 (visited on 04/11/2025).

Commission, Metropolitan Transportation (Nov. 2015). *Appendix A - Engineering Methodology, Environmental Engineering, and Permitting*. Metropolitan Transportation Commission. URL: https://mtc.ca.gov/sites/default/files/CCTS_InitialEngineeringStudy_Appendix_Nov2015.pdf (visited on 03/26/2024).

Conte, Enrico and Antonello Troncone (Nov. 1, 2006). "One-dimensional consolidation under general time-dependent loading". In: *Canadian Geotechnical Journal* 43.11, pp. 1107–1116. ISSN: 0008-3674, 1208-6010. DOI: 10.1139/t06-064. URL: http://www.nrcresearchpress.com/doi/10.1139/t06-064 (visited on 11/21/2024).

Dan, Yihua et al. (2023). "Influence analysis of calculated horizontally layered soil parameters on grounding parameters". In: *High Voltage* 8.2. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1049/hve2.12257, pp. 421–430. ISSN: 2397-7264. DOI: 10.1049/hve2.12257. (Visited on 02/11/2025).

Daryani, K E and H Mohamad (2015). "Effects of Soil Spatial Variability on Bearing Capacity of Shallow Foundations". In: *Geotechnical Safety and Risk V*. DOI: 10.3233/978-1-61499-580-7-371.

Das, Braja (2016). *Principles Of Foundation Engineering 7th Braja Das*. 8th. CENGAGE Learning. URL: http://archive.org/details/PrinciplesOfFoundationEngineering7thBrajaDas (visited on 07/30/2024).

Degago, S.A. et al. (Oct. 2011). "Use and misuse of the isotache concept with respect to creep hypotheses A and B". In: *Géotechnique* 61.10, pp. 897–908. ISSN: 0016-8505, 1751-7656. DOI: 10.1680/geot.9.P.112. URL: https://www.icevirtuallibrary.com/doi/10.1680/geot.9.P.112 (visited on 04/10/2025).

Di, Honggui et al. (Apr. 8, 2016). "Investigation of the long-term settlement of a cut-and-cover metro tunnel in a soft deposit". In: *Engineering Geology* 204, pp. 33–40. ISSN: 0013-7952. DOI: 10.1016/j.enggeo.2016.01.016. URL: https://www.sciencedirect.com/science/article/pii/S0013795216300163 (visited on 03/27/2025).

Ding, Qile et al. (Dec. 2024). "Subsurface Geological Profile Interpolation Using a Fractional Kriging Method Enhanced by Random Forest Regression". In: *Fractal and Fractional* 8.12. Number: 12 Publisher: Multidisciplinary Digital Publishing Institute, p. 717. ISSN: 2504-3110. DOI: 10.3390/fractalfract8120717. URL: https://www.mdpi.com/2504-3110/8/12/717 (visited on 02/11/2025).

Egeli, Isfendiyar and Nisa Kartaltepe (Dec. 1, 2012). "Preliminary design of an immersed tunnel in Izmir". In: *Gradjevinar* 64, pp. 1029–1040. DOI: 10.14256/JCE.817.2012.

Failmezger, R.A., D. Rom, and S.B. Ziegler (1999). *SPT? – A better approach to site characterization of residual soils using other In-Situ tests*. ASCE.

Fehmarn Belt fixed link (Feb. 21, 2024). In: *Wikipedia*. Page Version ID: 1209339946. URL: https://en.wikipedia.org/w/index.php?title=Fehmarn_Belt_fixed_link&oldid=1209339946 (visited on 03/11/2024).

FEHY (May 2013). *Fehmarnbelt Fixed Link EIA. Marine Soil – Impact Assessment. Seabed Morphology of the Fehmarnbelt Area.* E1TR0059 - Volume I. Hørsholm, Denmark: FEHY consortium, p. 140. URL: https://vvmdokumentation.femern.dk/da/ (visited on 10/31/2024).

Feng, W.-Q. and J.-H. Yin (2017). "A new simplified Hypothesis B method for calculating consolidation settlements of double soil layers exhibiting creep". In: *International Journal for Numerical and Analytical Methods in Geomechanics* 41.6. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nag.2635, pp. 899–917. ISSN: 1096-9853. DOI: 10.1002/nag.2635. (Visited on 11/08/2024).

Feng, Wei-Qiang et al. (Jan. 1, 2020). "A new simplified method for calculating consolidation settlement of multi-layer soft soils with creep under multi-stage ramp loading". In: *Engineering Geology* 264, p. 105322. ISSN: 0013-7952. DOI: 10.1016/j.enggeo.2019.105322. URL: https://www.sciencedirect.com/science/article/pii/S0013795219305058 (visited on 10/31/2024).

Fraser, R. A. and L. J. Wardle (Dec. 1976). "Numerical analysis of rectangular rafts on layered foundations". In: *Géotechnique* 26.4. Publisher: ICE Publishing, pp. 613–630. ISSN: 0016-8505. DOI: 10.1680/geot.1976.26.4.613. URL: https://www.icevirtuallibrary.com/doi/abs/10.1680/geot.1976.26.4.613 (visited on 04/11/2025).

Fu, Bai-yong et al. (Jan. 2020). "Summary of the Development of New Technologies for Submarine Immersed Tunnel Foundation Reinforcement and Settlement Control". In: *IOP Conference Series: Materials Science and Engineering* 741.1. Publisher: IOP Publishing, p. 012052. ISSN: 1757-899X. DOI: 10.1088/1757-899X/741/1/012052. URL: https://dx.doi.org/10.1088/1757-899X/741/1/012052 (visited on 02/19/2024).

Gavin, K G et al. (2019). "Investigation of the Remaining Life of an Immersed Tube Tunnel in the Netherlands". In: *Tunnels and Underground Cities: Engineering and Innovation meet Archaeology, Architecture and Art*.

Gibson, R. E. (Dec. 1958). "The Progress of Consolidation in a Clay Layer Increasing in Thickness with Time". In: *Géotechnique* 8.4. Publisher: ICE Publishing, pp. 171–182. ISSN: 0016-8505. DOI: 10.1680/geot.1958.8.4.171. URL: https://www.icevirtuallibrary.com/doi/abs/10.1680/geot.1958.8.4.171 (visited on 04/18/2025).

Grantz, Walter C (July 1, 2001). "Immersed tunnel settlements. Part 1: nature of settlements". In: *Tunnelling and Underground Space Technology* 16.3, pp. 195–201. ISSN: 0886-7798. DOI: 10.1016/S0886-7798(01)

00039-6. URL: https://www.sciencedirect.com/science/article/pii/S0886779801000396 (visited on 11/26/2024).

Hakimian, R. (Apr. 17, 2024). *18km trench dredging complete and Danish portal submerged*. New Civil Engineer. URL: https://www.newcivilengineer.com/latest/fehmarnbelt-tunnel-18km-trench-dredging-complete-and-danish-portal-submerged-17-04-2024/ (visited on 11/29/2024).

Hamza, Adnan (2016). "Integrated Finite Element Analysis and Design of Structures ANALYSIS REFERENCE". In: *Open Journal of Civil Engineering* 6.4. URL: https://www.academia.edu/8413025/Integrated_Finite_Element_Analysis_and_Design_of_Structures_ANALYSIS_REFERENCE (visited on 10/02/2024).

Hassan, Mohammed A (2017). "Modified Schmertmann,s Method (1978)for Calculating Settlement In Sand Soils By Using Integration". In: 7.8.

Haukaas, Terje (2023). "Euler-Bernoulli Beams". In: URL: terje.civil.ubc.ca.

Hawlader, B.C., B. Muhunthan, and G. Imai (Sept. 2003). "Viscosity Effects on One-Dimensional Consolidation of Clay". In: *https://ascelibrary-org.tudelft.idm.oclc.org/* 3.1. ISSN: 1532-3641. DOI: 10.1061/(ASCE)1532-3641(2003)3:1(99). URL: https://ascelibrary.org/doi/epdf/10.1061/%28ASCE%291532-3641%282003%293%3A1%2899%29 (visited on 04/10/2025).

Heijden, Marc van der (2023). *Performance of immersed tunnels subjected to settlements*. COB. URL: https://www.cob.nl/document/performance-of-immersed-tunnels-subjected-to-settlements/ (visited on 03/27/2025).

Hirai, Hiroyoshi (2008). "Settlements and stresses of multi-layered grounds and improved grounds by equivalent elastic method". In: *International Journal for Numerical and Analytical Methods in Geomechanics* 32.5. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/nag.636, pp. 523–557. ISSN: 1096-9853. DOI: 10.1002/nag.636. URL: https://onlinelibrary.wiley.com/doi/abs/10.1002/nag.636 (visited on 04/11/2025).

Hirai, Hiroyoshi and Takeshi KAMEI (July 30, 2004). "A METHOD TO CALCULATE SETTLEMENT, STRESS, FAILURE AND ALLOXVABLE STRESS OF MULTI-LAYERED GROUND BY EQUIVALENT THICKNESS THEORY". In: *Journal of Structural and Construction Engineering (Transactions of AIJ)* 69, pp. 79–86. DOI: 10.3130/aijs.69.79_3.

Hsu, Tung-Wen and Shuan-Chai Lu (Apr. 1, 2006). "Behavior of One-Dimensional Consolidation under Time-Dependent Loading". In: *Journal of Engineering Mechanics* 132.4. Publisher: American Society of Civil Engineers, pp. 457–462. ISSN: 0733-9399. DOI: 10.1061/(ASCE)0733-9399(2006)132:4(457). URL: https://ascelibrary.org/doi/10.1061/%28ASCE%290733-9399%282006%29132%3A4%28457%29 (visited on 04/18/2025).

Hu, Zhinan et al. (Sept. 1, 2018). "Advantages and potential challenges of applying semi-rigid elements in an immersed tunnel: A case study of the Hong Kong-Zhuhai-Macao Bridge". In: *Tunnelling and Underground Space Technology* 79, pp. 143–149. ISSN: 0886-7798. DOI: 10.1016/j.tust.2018.05.004. URL: https://www.sciencedirect.com/science/article/pii/S0886779817308593 (visited on 03/07/2024).

Joseph, Paul G. (Sept. 2014). "Viscosity and secondary consolidation in one-dimensional loading". In: *Geotechnical Research* 1.3, pp. 90–98. ISSN: 2052-6156. DOI: 10.1680/gr.14.00008. URL: https://www.icevirtuallibrary.com/doi/10.1680/gr.14.00008 (visited on 04/10/2025).

Kaczmarek, Łukasz and Paweł Dobak (Oct. 22, 2024). "(PDF) Contemporary overview of soil creep phenomenon". In: *ResearchGate*. DOI: 10.1515/ctg-2017-0003. URL: https://www.researchgate.net/publication/317399783_Contemporary_overview_of_soil_creep_phenomenon (visited on 02/11/2025).

Kammer, Jens et al. (Jan. 1, 2012). *FEHMARNBELT FIXED LINK. GEOTECHNICAL INVESTIGATIONS*.

Kou, Lei et al. (Apr. 12, 2024). *Long-term settlement analysis of immersed tube tunnel in operation considering undisturbed clayey soil foundation*. DOI: 10.21203/rs.3.rs-4223004/v1. URL: https://www.researchsquare.com/article/rs-4223004/v1 (visited on 04/14/2025).

Krogsbøll, A, O Hededal, and N Foged (2012). "Deformation properties of highly plastic fissured Palaeogene clay – Lack of stress memory?" In: *Proceedings of 16th Nordic geotechnical meeting NGM 2012*. Nordic Geotechnical Meeting. Vol. 1. Copenhagen, Denmark, pp. 133–140.

Łotysz, Sławomir (Jan. 1, 2010). "Immersed tunnel technology: A brief history of its development". In: *Civil and Environmental Engineering Reports*, pp. 97–110.

Martins, I.S.M. (1985). "A Theory of consolidation with secondary compression". In: 11th International Conference on Soil Mechanics and Foundation Engineering. San Francisco, pp. 567–570. ISBN: 90 6191 564 3. URL: https://www.issmge.org/uploads/publications/1/34/1985_02_0037.pdf.

Morfidis, K. (Aug. 1, 2007). "Exact matrices for beams on three-parameter elastic foundation". In: *Computers & Structures* 85.15, pp. 1243–1256. ISSN: 0045-7949. DOI: 10.1016/j.compstruc.2006.11.030. URL: https://www.sciencedirect.com/science/article/pii/S0045794906003981 (visited on 10/31/2024).

NEN (1997). *Geotechnical design of strutures - Part 1: General rules*. URL: https://www.nen.nl/en/nen-9997-1-2023-ontw-nl-317797.

Noor, M.A.M. and Mohd Effendi Daud (Apr. 20, 2016). "Determination of soil thickness based on natural frequency using microtremor measurement". In: *ARPN Journal of Engineering and Applied Sciences* 11.8, pp. 5342–5346.

Nova, Roberto and Tomasz Hueckel (Sept. 1981). "An engineering theory of soil behaviour in unloading and reloading". In: *Meccanica* 16.3, pp. 136–148. ISSN: 0025-6455, 1572-9648. DOI: 10.1007/BF02128442. URL: http://link.springer.com/10.1007/BF02128442 (visited on 04/09/2025).

Odemark, N. (1949). "INVESTIGATIONS AS TO THE ELASTIC PROPERTIES OF SOILS AND DESIGN OF PAVEMENTS ACCORDING TO THE THEORY OF ELASTICITY". In: *Statens Vaginstitut /Sweden/*. Number: Meddelande 77. URL: https://trid.trb.org/View/97660 (visited on 04/11/2025).

Olsen, T., T. Kasper, and J. de. Wit (Mar. 1, 2022). "Immersed tunnels in soft soil conditions experience from the last 20 years". In: *Tunnelling and Underground Space Technology* 121, p. 104315. ISSN: 0886-7798. DOI: 10.1016/j.tust.2021.104315. URL: https://www.sciencedirect.com/science/article/pii/S088677982100506X (visited on 03/04/2024).

Olson, R.E. (1989). "Secondary Consolidation". In: *Advanced Soil Mechanics*.

Olson, Roy E. and Charles C. Ladd (Jan. 1, 1979). "One-Dimensional Consolidation Problems". In: *Journal of the Geotechnical Engineering Division* 105.1. Publisher: American Society of Civil Engineers, pp. 11–30. DOI: 10.1061/AJGEB6.0000749. URL: https://ascelibrary.org/doi/10.1061/AJGEB6.0000749 (visited on 04/18/2025).

Pantelidis, Lysandros (Dec. 15, 2019). "The equivalent modulus of elasticity of layered soil mediums for designing shallow foundations with the Winkler spring hypothesis: A critical review". In: *Engineering Structures* 201, p. 109452. ISSN: 0141-0296. DOI: 10.1016/j.engstruct.2019.109452. URL: https://www.sciencedirect.com/science/article/pii/S0141029619311952 (visited on 04/04/2025).

PPIAF (Nov. 30, 2020). *The Øresund Fixed Link*. URL: https://www.gihub.org/connectivity-across-borders/case-studies/the-%C3%B8resund-fixed-link/ (visited on 05/19/2025).

Proceedings of the Fifth International Conference on Soil Mechanics and Foundation Engineering, Held in Paris, 17-22 July, 1961 (1961). Dunod. book.

Raychaudhuri, Samik (2008). "INTRODUCTION TO MONTE CARLO SIMULATION". In: *Proceedings of the 2008 Winter Simulation Conference*. Winter simulation conference. Broomfield, Colorado: Oracle Crystal Ball Global Business Unit.

Sakaeda, Hideki (Nov. 1, 2005). "Marmaray project: Tunnels and stations in BC contract". In: *Tunnelling and Underground Space Technology* 20.6, pp. 612–616. ISSN: 0886-7798. DOI: 10.1016/j.tust.2005.08.007. URL: https://www.sciencedirect.com/science/article/pii/S0886779805000714 (visited on 03/07/2024).

Savage, W.Z., M.M. Morrissey, and R.L. Baum (2000). *Open-File Report*. Open-File Report. Series: Open-File Report.

Schiffman, Robert L. and Jack R. Stein (July 1, 1970). "One-Dimensional Consolidation of Layered Systems". In: *Journal of the Soil Mechanics and Foundations Division* 96.4. Publisher: American Society of Civil Engineers, pp. 1499–1504. DOI: 10.1061/JSFEAQ.0001453. URL: https://ascelibrary.org/doi/10.1061/JSFEAQ.0001453 (visited on 04/18/2025).

SEISMIC REFRACTION SURVEY, LAC DE MONTIGNY (Oct. 2005). ALEXANDRIA MINERALS CORPORATION. URL: https://gq.mines.gouv.qc.ca/documents/examine/GM62150/GM62150.pdf.

Selvadurai, A P S. (1979). *Elastic analysis of soil-foundation interaction*. Developments in geotechnical engineering vol. 17. Section: XIV, 543 p. : ill. ; 25 cm. Amsterdam etc.: Elsevier. 543 pp.

Shen, Mingde, Zhiwei Zhou, and Wei Ma (Aug. 1, 2023). "Experimental and Theoretical Investigation on the Unloading Creep Behaviors of Frozen Soil". In: *Rock Mechanics and Rock Engineering* 56.8, pp. 5833–

5859. ISSN: 1434-453X. DOI: 10.1007/s00603-023-03369-1. URL: https://doi.org/10.1007/s00603-023-03369-1 (visited on 04/09/2025).

Sridharan, A., N. S. V. V. S. J. Gandhi, and S. Suresh (Apr. 1, 1990). "Stiffness Coefficients of Layered Soil Systems". In: *Journal of Geotechnical Engineering* 116.4. Publisher: American Society of Civil Engineers, pp. 604–624. ISSN: 0733-9410. DOI: 10.1061/(ASCE)0733-9410(1990)116:4(604). URL: https://ascelibrary.org/doi/10.1061/%28ASCE%290733-9410%281990%29116%3A4%28604%29 (visited on 04/11/2025).

Szavits-Nossan, Vlasta (Jan. 1, 2015). *Consolidation and creep: Hypotheses A and B revisited, 16th ECSMGE, Edinburgh (2015)*.

Takeda, Toshihiko et al. (Aug. 2013). "INITIAL RATE OF SECONDARY COMPRESSION IN ONE-DIMENSIONAL CONSOLIDATION ANALYSIS". In: *Journal of GeoEngineering* 8.2, pp. 55–60.

Tang, Cong et al. (Oct. 1, 2023). "Enhanced elastic beam model with BADS integrated for settlement assessment of immersed tunnels". In: *Underground Space* 12, pp. 79–88. ISSN: 2467-9674. DOI: 10.1016/j.undsp.2023.02.005. URL: https://www.sciencedirect.com/science/article/pii/S2467967423000491 (visited on 03/18/2024).

Terzaghi, Karl, Ralph B. Peck, and Gholamreza Mesri (Feb. 7, 1996). *Soil Mechanics in Engineering Practice*. Google-Books-ID: XjH6DwAAQBAJ. John Wiley & Sons. 597 pp. ISBN: 978-0-471-08658-1.

Testbook (Nov. 19, 2024). *Soil Mechanics: Soil Formation, Compaction, Shear Strength & Consolidation*. Testbook. URL: https://testbook.com/civil-engineering/soil-mechanics-and-foundations (visited on 04/08/2025).

Ueshita, K. and G.G. Meyerhof (Sept. 1967). "Deflection of Multilayer Soil Systems | Journal of the Soil Mechanics and Foundations Division | Vol 93, No 5". In: *Journal of the Soil Mechanics and Foundations Division* 93.5. URL: https://ascelibrary-org.tudelft.idm.oclc.org/doi/abs/10.1061/JSFEAQ.0001023 (visited on 04/11/2025).

Ural, Nazile and Mansoor Zoveidavianpoor (Sept. 12, 2018). "The Importance of Clay in Geotechnical Engineering". In: *Current Topics in the Utilization of Clay in Industrial and Medical Applications*. IntechOpen. ISBN: 978-1-78923-729-0. DOI: 10.5772/intechopen.75817. URL: https://www.intechopen.com/chapters/60931 (visited on 04/08/2025).

Uzielli, Marco et al. (Dec. 1, 2006). *Soil Variability Analysis for Geotechnical Practice*. Vol. 3. Journal Abbreviation: Characterisation and Engineering Properties of Natural Soils Publication Title: Characterisation and Engineering Properties of Natural Soils. ISBN: 978-0-415-42691-6. DOI: 10.1201/NOE0415426916.ch3.

Wang, Xiaowei (June 1, 2021). "Empirical Probability Distribution Models for Soil-Layer Thicknesses of Liquefiable Ground". In: *Journal of Geotechnical and Geoenvironmental Engineering* 147. DOI: 10.1061/(ASCE)GT.1943-5606.0002537.

Wang, Yan-Ning, Le-Chen Wang, and Lin-Shuang Zhao (Dec. 2023). "Settlement characteristics of immersed tunnel of Hong Kong–Zhuhai–Macau Bridge project". In: *Proceedings of the Institution of Civil Engineers - Geotechnical Engineering* 176.6. Publisher: ICE Publishing, pp. 605–617. ISSN: 1353-2618. DOI: 10.1680/jgeen.22.00200. URL: https://www.icevirtuallibrary.com/doi/10.1680/jgeen.22.00200 (visited on 04/14/2025).

Wang, Yanning et al. (Sept. 1, 2023). "Deformation analysis for an immersed tunnel considering cyclic degradation of substratum soil". In: *Applied Ocean Research* 138, p. 103681. ISSN: 0141-1187. DOI: 10.1016/j.apor.2023.103681. URL: https://www.sciencedirect.com/science/article/pii/S0141118723002225 (visited on 07/29/2024).

Wei, Gang, Hui Jie Qiu, and Xin Jiang Wei (Nov. 2012). "Analysis of Settlement Reasons and Mechanism in Immersed Tunnel". In: *Applied Mechanics and Materials* 238, pp. 803–807. ISSN: 1662-7482. DOI: 10.4028/www.scientific.net/AMM.238.803. URL: https://www.scientific.net/AMM.238.803 (visited on 02/19/2024).

Worku, Asrat (Aug. 1, 2013). "Calibrated Analytical Formulas for Foundation Model Parameters". In: *International Journal of Geomechanics* 13.4. Publisher: American Society of Civil Engineers, pp. 340–347. ISSN: 1943-5622. DOI: 10.1061/(ASCE)GM.1943-5622.0000214. URL: https://ascelibrary-org.tudelft.idm.oclc.org/doi/10.1061/%28ASCE%29GM.1943-5622.0000214 (visited on 09/04/2024).

Wu, Xinhang (2017). "Impact of spatial variability of subsoil stiffness on immersed tunnels". In: URL: https://repository.tudelft.nl/islandora/object/uuid%3A959f435e-b404-46c4-87e6-cf36797b9df3 (visited on 02/15/2024).

Yamashita, Naoyuki et al. (June 1, 2024). "National-scale mapping of soil-thickness probability in hilly and mountainous areas of Japan using legacy and modern soil survey". In: *Geoderma* 446, p. 116896. ISSN: 0016-7061. DOI: 10.1016/j.geoderma.2024.116896. URL: https://www.sciencedirect.com/science/article/pii/S0016706124001253 (visited on 04/13/2025).

Yan, Tingting et al. (Feb. 1, 2021). "Spatial distribution characteristics of the soil thickness on different land use types in the Yimeng Mountain Area, China". In: *Alexandria Engineering Journal* 60.1, pp. 511–520. ISSN: 1110-0168. DOI: 10.1016/j.aej.2020.09.024. URL: https://www.sciencedirect.com/science/article/pii/S1110016820304804 (visited on 04/13/2025).

Yin, Jian-Hua and Wei-Qiang Feng (Mar. 2017). "A new simplified method and its verification for calculation of consolidation settlement of a clayey soil with creep". In: *Canadian Geotechnical Journal* 54.3. Publisher: NRC Research Press, pp. 333–347. ISSN: 0008-3674. DOI: 10.1139/cgj-2015-0290. URL: https://cdnsciencepub.com/doi/full/10.1139/cgj-2015-0290 (visited on 11/11/2024).

Yumpu.com (May 2011). *Ground Investigation Report May 2011 - Femern*. yumpu.com. URL: https://www.yumpu.com/en/document/read/20787278/ground-investigation-report-may-2011-femern (visited on 04/05/2024).

Zhang, X. and W. Broere (2019). "Settlements of immersed tunnel on soft ground: World Tunnel Congress, WTC 2019 and the 45th General Assembly of the International Tunnelling and Underground Space Association, ITA-AITES 2019". In: *Tunnels and Underground Cities*. Ed. by Daniele Peila, Giulia Viggiani, and Tarcisio Celestino. Publisher: CRC Press / Balkema - Taylor & Francis Group, pp. 1234–1241. ISSN: 9781138388659. DOI: 10.1201/9780429424441-132. URL: http://www.scopus.com/inward/record.url?scp=85068390822&partnerID=8YFLogxK (visited on 02/15/2024).

Zhou, Huanzhu et al. (June 1, 2025). "Improved Deformation Calculation Method for Immersed Tunnels Considering Siltation Effects: A Case Study of the Yongjiang Immersed Tunnel in Ningbo, Zhejiang Province, China". In: *International Journal of Geomechanics* 25.6. Publisher: American Society of Civil Engineers, p. 04025105. DOI: 10.1061/IJGNAI.GMENG-10631. URL: https://ascelibrary.org/doi/10.1061/IJGNAI.GMENG-10631 (visited on 04/14/2025).

Zhu, G. and Jianhua Yin (June 1, 1999). "Consolidation of double soil layers under depth-dependent ramp load". In: *ICE Publishing* 49, pp. 415–421. ISSN: 0016-8505.

A

Appendix A

All formulas, parameters and matrices are retrieved from Morfidis 2007.

Stiffness matrix $[K] = [Q][R]^{-1}$ for the most common solution case

R *Matrix*

$$
\begin{array}{lll}
R_{11} = R_{12} = A_1, & R_{13} = R_{15} = A_3, & R_{14} = R_{16} = A_2 \\
R_{21} = R_{22} = A_4, & R_{23} = R_{25} = A_5, & R_{24} = R_{26} = A_6 \\
R_{31} = R_{32} = 1, & R_{33} = R_{35} = 1, & R_{34} = R_{36} = 0 \\
R_{41} = A_1 f_{1L}, & R_{42} = A_1 f_{2L}, & R_{43} = A_3 f_{3L} + A_2 f_{4L} \\
R_{44} = A_3 f_{4L} - A_2 f_{3L}, & R_{45} = A_3 f_{5L} - A_2 f_{6L}, & R_{46} = A_3 f_{6L} + A_2 f_{5L} \\
R_{51} = A_4 f_{1L}, & R_{52} = A_4 f_{2L}, & R_{53} = A_5 f_{3L} - A_6 f_{4L} \\
R_{54} = A_6 f_{3L} + A_5 f_{4L}, & R_{55} = A_5 f_{5L} - A_6 f_{6L}, & R_{56} = A_6 f_{5L} - A_5 f_{6L} \\
R_{61} = f_{1L}, & R_{62} = f_{2L}, & R_{63} = f_{3L} \\
R_{64} = f_{4L}, & R_{65} = f_{5L}, & R_{66} = f_{6L}
\end{array}
$$

Q *Matrix*

$$
\begin{array}{lll}
Q_{11} = Q_{12} = d_1, & Q_{13} = Q_{15} = d_2, & Q_{14} = Q_{16} = d_3 \\
Q_{21} = Q_{22} = (EI)(A_4 R_1), & Q_{23} = Q_{25} = (EI) b_4, & Q_{26} = Q_{24} = (EI) b_3 \\
Q_{31} = Q_{32} = G R_1, & Q_{35} = Q_{33} = GR, & Q_{34} = Q_{36} = GQ \\
Q_{41} = d_1 f_{1L}, & Q_{42} = d_1 f_{2L}, & Q_{43} = d_2 f_{3L} + d_3 f_{4L} \\
Q_{44} = d_3 f_{3L} - d_2 f_{4L}, & Q_{45} = d_2 f_{5L} + d_3 f_{6L}, & Q_{46} = d_3 f_{5L} + d_2 f_{6L} \\
Q_{51} = (EI) A_4 R_1 f_{1L}, & Q_{52} = (EI) A_4 R_1 f_{2L}, & Q_{53} = (EI)(b_4 f_{3L} - b_3 f_{4L}) \\
Q_{54} = (EI)(b_3 f_{3L} + b_4 f_{4L}), & Q_{55} = (EI)(b_4 f_{5L} + b_3 f_{6L}), & Q_{56} = (EI)(b_4 f_{6L} - b_3 f_{5L}) \\
Q_{61} = G R_1 f_{1L}, & Q_{62} = G R_1 f_{2L}, & Q_{63} = G(R f_{3L} - Q f_{4L}) \\
Q_{64} = G(R f_{4L} + Q f_{3L}), & Q_{65} = G(R f_{5L} + Q f_{6L}), & Q_{66} = G(R f_{6L} + Q f_{5L})
\end{array}
$$

Functions and Parameters

$$
\begin{array}{llll}
f_{1L} & = e^{R_1 L}, & f_{2L} & = e^{-R_1 L} \\
f_{3L} & = e^{RL}\cos(QL), & f_{4L} & = e^{RL}\sin(QL) \\
f_{5L} & = e^{-RL}\cos(QL), & f_{6L} & = e^{-RL}\sin(QL) \\
b_1 & = (R^2 - Q^2)A_5 - 2RQA_6, & b_2 & = (R^2 - Q^2)A_6 + 2RQA_5 \\
b_3 & = A_6 R + A_5 Q, & b_4 & = A_5 R - A_6 Q \\
d_1 & = EI(A_4 R_1^2), & d_2 & = EIb_1 \\
d_3 & = EIb_2, & A_1 & = 1 + \frac{k}{c} + \frac{G}{c}R_1^2 \\
A_2 & = 2RQ\frac{G}{c}, & A_3 & = 1 + \frac{k}{c} + \frac{G}{c}(R^2 - Q^2) \\
A_4 & = \frac{UR_1}{EIR_1^2 + U}, & A_5 & = \frac{a_1 b_1 + a_2 b_2}{b_1^2 + b_2^2}U \\
A_6 & = \frac{a_1 b_2 - a_2 b_1}{b_1^2 + b_2^2}U, & a_1 & = A_3 R + A_2 Q \\
a_2 & = A_2 R - A_3 Q, & b_1 & = EI(R^2 - Q^2) + U \\
b_2 & = 2(EI)RQ, & R_1 & = \sqrt[3]{\frac{b + \sqrt{D}}{3}} + \sqrt[3]{\frac{b - \sqrt{D}}{3}} \\
R & = \sqrt{\frac{m + \sqrt{m^2 + n^2}}{2}}, & Q & = \sqrt{\frac{m - \sqrt{m^2 + n^2}}{2}} \\
m & = \frac{1}{2}\left(\sqrt[3]{b + \sqrt{D}} + \sqrt[3]{b - \sqrt{D}} + \frac{2J_1}{3}\right), & n & = \frac{\sqrt{3}}{2}\left(\sqrt[3]{b + \sqrt{D}} - \sqrt[3]{b - \sqrt{D}}\right) \\
a & = \frac{1}{3}\left(\frac{J_1^2}{3} + J_2\right), & b & = \frac{1}{2}\left(\frac{2J_1^3}{27} - \frac{J_1 J_2}{3} + J_3\right) \\
J_1 & = k + \frac{c}{G} + \frac{c}{U}, & J_2 & = \frac{cU}{EI} + \frac{kG}{U}, \\
J_3 & = \frac{kc}{EIG}, & D = a^3 + b^2 & \\
\Phi & = G_B F_0 &&
\end{array}
$$

Transfer matrix **F** for the most common solution case.

$$
\begin{aligned}
f_{11}(x) &= f_{44}(x) = \tfrac{A_1 b_3}{D_1} m_1(x) + \tfrac{A_3 b_3 - A_2(A_4 R_1 - b_4)}{D_1} m(x)\bar{m}(x) + \tfrac{A_2 b_3 + A_3(A_4 R_1 - b_4)}{D_1} n(x)\bar{n}(x) \\
f_{12}(x) &= f_{54}(x) = \tfrac{A_1(d_2 Q - d_3 R)}{D_2} n_1(x) - \tfrac{A_3(d_1 Q - d_3 R_1) + A_2(d_1 R - d_2 R_1)}{D_2} n(x)\bar{m}(x) - \tfrac{A_3(d_2 R_1 - d_1 R) + A_2(d_1 Q - d_3 R_1)}{D_2} m(x)\bar{n}(x) \\
f_{13}(x) &= f_{64}(x) = \tfrac{A_1(A_3 b_3 + A_2 b_4)}{D_1} m_1(x) - \tfrac{A_1(A_3 b_3 + A_2 b_4)}{D_1} m(x)\bar{m}(x) - \tfrac{A_4 R_1(A_2^2 + A_3^2) + A_1(A_2 b_3 - A_3 b_4)}{D_1} n(x)\bar{n}(x) \\
f_{14}(x) &= \tfrac{A_1(A_5 Q - A_6 R)}{D_2} n_1(x) - \tfrac{A_3(A_6 R_1 - A_4 Q) + A_2(A_5 R_1 - A_4 R)}{D_2} n(x)\bar{m}(x) + \tfrac{A_2(A_6 R_1 + A_4 Q) + A_3(A_5 R_1 - A_4 R)}{D_2} m(x)\bar{n}(x) \\
f_{15}(x) &= f_{24}(x) = \tfrac{A_1 A_2}{EID_1} m_1(x) - \tfrac{A_1 A_2}{EID_1} m(x)\bar{m}(x) + \tfrac{A_3(A_1 - A_3)A_2^2}{EID_1} n(x)\bar{n}(x) \\
f_{16}(x) &= f_{34}(x) = \tfrac{(A_5 Q - A_6 R)}{D_2} n_1(x) + \tfrac{(A_4 Q - A_6 R_1)}{D_2} n(x)\bar{m}(x) - \tfrac{(A_4 R - A_5 R_1)}{D_2} m(x)\bar{n}(x) \\
f_{31}(x) &= f_{46}(x) = \tfrac{b_3}{D_1} m_1(x) + \tfrac{b_3}{D_1} m(x)\bar{m}(x) + \tfrac{A_4 R_1 - b_4}{D_1} n(x)\bar{n}(x) \\
f_{32}(x) &= f_{56}(x) = \tfrac{d_2 Q - d_3 R}{D_2} n_1(x) - \tfrac{d_1 Q - d_3 R_1}{D_2} n(x)\bar{m}(x) + \tfrac{d_1 R - d_2 R_1}{D_2} m(x)\bar{n}(x) \\
f_{33}(x) &= f_{66}(x) = \tfrac{A_3 b_3 + A_2 b_4}{D_1} m_1(x) - \tfrac{A_1 b_3 + A_2 A_4 R_1}{D_1} m(x)\bar{m}(x) - \tfrac{A_1 b_4 + A_3 A_4 R_1}{D_1} n(x)\bar{n}(x) \\
f_{35}(x) &= f_{26}(x) = \tfrac{A_2}{EID_1} m_1(x) - \tfrac{A_2}{EID_1} m(x)\bar{m}(x) + \tfrac{A_1 A_3}{EID_1} n(x)\bar{n}(x) \\
f_{36}(x) &= \tfrac{A_5 d_3 - A_6 d_2}{GD_2} n_1(x) + \tfrac{A_4 d_3 - A_6 d_1}{GD_2} n(x)\bar{m}(x) - \tfrac{A_4 d_2 - A_5 d_1}{GD_2} m(x)\bar{n}(x) \\
f_{21}(x) &= f_{45}(x) = \tfrac{A_4 b_3}{D_1} n_1(x) + \tfrac{A_5 b_3 + A_6(A_4 R_1 - b_4)}{D_1} n(x)m(x) - \tfrac{A_6 b_3 + A_5(b_4 - A_4 R_1)}{D_1} m(x)n(x) \\
f_{22}(x) &= f_{55}(x) = \tfrac{A_2 A_4 R_1}{D_1} m_1(x) - \tfrac{(A_1 - A_3) b_3 A_2 b_4}{D_1} m(x)\bar{m}(x) - \tfrac{(A_1 - A_3) b_4 + A_2 b_3}{D_1} n(x)\bar{n}(x) \\
f_{23}(x) &= f_{65}(x) = \tfrac{G R_1 A_2}{EID_1} n_1(x) + \tfrac{G[(A_1 - A_3) Q A_2 R]}{EID_1} n(x)\bar{m}(x) + \tfrac{G[(A_1 - A_3) R + A_2 Q]}{EID_1} m(x)\bar{n}(x) \\
f_{25}(x) &= \tfrac{A_2 A_4}{EID_1} n_1(x) - \tfrac{A_2 A_5 (A_1 - A_3) A_6}{EID_1} n(x)\bar{m}(x) + \tfrac{A_2 A_6 + A_5(A_1 - A_3)}{EID_1} m(x)\bar{n}(x) \\
f_{42}(x) &= f_{51}(x) = \tfrac{EIR_1 A_4 b_3}{D_1} m_1(x) + \tfrac{EIR_1 A_4 b_3}{D_1} m(x)\bar{m}(x) - \tfrac{EI[b_3^2 + b_4(b_4 - A_4 R_1)]}{D_1} n(x)\bar{n}(x) \\
f_{41}(x) &= \tfrac{b_3 d_1}{D_1} n_1(x) + \tfrac{EI[A_4 R_1 b_2 Q(R^2 + Q^2)(A_5^2 + A_6^2)]}{D_1} n(x)\bar{m}(x) + \tfrac{EI[A_4 R_1 b_1 R(R^2 + Q^2)(A_5^2 + A_6^2)]}{D_1} m(x)\bar{n}(x) \\
f_{43}(x) &= f_{61}(x) = \tfrac{G b_3 R_1}{D_1} n_1(x) - \tfrac{G[b_3 R Q(b_4 - A_4 R_1)]}{D_1} n(x)m(x) + \tfrac{G[b_3 Q + R(b_4 - A_4 R_1)]}{D_1} m(x)n(x) \\
f_{52}(x) &= \tfrac{EIR_1 A_4 (d_3 R - d_2 Q)}{D_2} n_1(x) - \tfrac{EI[b_3(d_1 R - d_2 R_1) - b_4(d_1 Q - d_3 R_1)]}{D_2} n(x)\bar{m}(x) + \tfrac{EI[b_3(d_1 Q - d_3 R_1) + b_4(d_1 R - d_2 R_1)]}{D_2} m(x)\bar{n}(x) \\
f_{53}(x) &= f_{62}(x) = \tfrac{G R_1 (d_3 R - d_2 Q)}{D_2} m_1(x) + \tfrac{G R_1 (d_3 R - d_2 Q)}{D_2} m(x)\bar{m}(x) - \tfrac{G[R_1(d_2 R + d_3 Q) - d_1(R^2 + Q^2)]}{D_2} n(x)\bar{n}(x) \\
f_{63}(x) &= \tfrac{G R_1 (A_3 b_3 + A_2 b_4)}{D_1} n_1(x) + \tfrac{G[A_1(R b_3 - Q b_4) + A_4 R_1(A_2 R + A_3 Q)]}{D_1} n(x)\bar{m}(x) - \tfrac{G[A_1(Q b_3 + R b_4) + A_4 R_1(Q A_2 - R A_3)]}{D_1} m(x)\bar{n}(x) \\
f_{25}(x) &= \tfrac{A_2 A_4}{EID_1} n_1(x) - \tfrac{A_2 A_5 (A_1 - A_3) A_6}{EID_1} n(x)\bar{m}(x) + \tfrac{A_2 A_6 + A_5(A_1 - A_3)}{EID_1} m(x)\bar{n}(x) \\
f_{42}(x) &= f_{51}(x) = \tfrac{EIR_1 A_4 b_3}{D_1} m_1(x) + \tfrac{EIR_1 A_4 b_3}{D_1} m(x)\bar{m}(x) - \tfrac{EI[b_3^2 + b_4(b_4 - A_4 R_1)]}{D_1} n(x)\bar{n}(x) \\
f_{41}(x) &= \tfrac{b_3 d_1}{D_1} n_1(x) + \tfrac{EI[A_4 R_1 b_2 Q(R^2 + Q^2)(A_5^2 + A_6^2)]}{D_1} n(x)\bar{m}(x) + \tfrac{EI[A_4 R_1 b_1 R(R^2 + Q^2)(A_5^2 + A_6^2)]}{D_1} m(x)\bar{n}(x) \\
f_{43}(x) &= f_{61}(x) = \tfrac{G b_3 R_1}{D_1} n_1(x) - \tfrac{G[b_3 R Q(b_4 - A_4 R_1)]}{D_1} n(x)\bar{m}(x) + \tfrac{G[b_3 Q + R(b_4 - A_4 R_1)]}{D_1} m(x)\bar{n}(x) \\
f_{52}(x) &= \tfrac{EIR_1 A_4 (d_3 R - d_2 Q)}{D_2} n_1(x) - \tfrac{EI[b_3(d_1 R - d_2 R_1) - b_4(d_1 Q - d_3 R_1)]}{D_2} n(x)\bar{m}(x) + \tfrac{EI[b_3(d_1 Q - d_3 R_1) + b_4(d_1 R - d_2 R_1)]}{D_2} m(x)\bar{n}(x) \\
f_{53}(x) &= f_{62}(x) = \tfrac{G R_1 (d_3 R - d_2 Q)}{D_2} m_1(x) + \tfrac{G R_1 (d_3 R - d_2 Q)}{D_2} m(x)\bar{m}(x) - \tfrac{G[R_1(d_2 R + d_3 Q) - d_1(R^2 + Q^2)]}{D_2} n(x)\bar{n}(x) \\
f_{63}(x) &= \tfrac{G R_1 (A_3 b_3 + A_2 b_4)}{D_1} n_1(x) + \tfrac{G[A_1(R b_3 - Q b_4) + A_4 R_1(A_2 R + A_3 Q)]}{D_1} n(x)\bar{m}(x) - \tfrac{G[A_1(Q b_3 + R b_4) + A_4 R_1(Q A_2 - R A_3)]}{D_1} m(x)\bar{n}(x) \\
D_1 &= b_3(A_3 - A_1) + A_2(b_4 - A_4 R_1) \\
D_2 &= Q(A_4 d_2 - A_5 d_1) + R(A_6 d_1 - A_4 d_3) + R_1(A_5 d_3 - A_6 d_2) \\
m_1(x) &= \cosh(R_1 x) \\
n_1(x) &= \sinh(R_1 x) \\
m(x) &= \cos(Q x) \\
n(x) &= \sin(Q x) \\
\bar{m}(x) &= \cosh(R x) \\
\bar{n}(x) &= \sinh(R x)
\end{aligned}
$$

Trapezoidal load at a random portion of the element

$$
\begin{aligned}
V_0^{(q)} &= f_V^{(q)} D \\
M_0^{(q)} &= f_M^{(q)} D \\
V_{G0}^{(q)} &= f_{VG}^{(q)} D \\
V_L^{(q)} &= \bar{f}_V^{(q)} D \\
M_L^{(q)} &= \bar{f}_M^{(q)} D \\
V_{GL}^{(q)} &= \bar{f}_{VG}^{(q)} D
\end{aligned}
$$

Load vector elements

$$
\begin{aligned}
f_V &= a(f_{25L}f_{34L} + f_{35L}f_{15L}) - b(f_{35L}^2 + f_{25L}f_{36L}) + c(f_{15L}f_{36L} - f_{34L}f_{35L}) \\
f_{VG} &= a(f_{15L}^2 + f_{25L}f_{14L}) + b(f_{25L}f_{34L} + f_{15L}f_{35L}) + c(f_{14L}f_{35L} - f_{15L}f_{34L}) \\
f_M &= c(f_{34L}^2 - f_{14L}f_{36L}) + b(f_{34L}f_{35L} - f_{15L}f_{36L}) + a(f_{15L}f_{34L} - f_{14L}f_{35L}) \\
D &= f_{25L}(f_{34L}^2 - f_{14L}f_{36L}) + f_{35L}(f_{14L}f_{35L} - f_{15L}f_{34L}) - f_{15L}(f_{34L}f_{35L} - f_{15L}f_{36L}) \\
a &= \begin{cases} -I_{wk} \\ -\bar{I}_{wk} \end{cases} \\
b &= \begin{cases} -I_{w} \\ -\bar{I}_{w} \end{cases} \\
c &= \begin{cases} -I_{G} \\ -\bar{I}_{G} \end{cases}
\end{aligned}
$$

Elements of Transfer Matrix

$$
\begin{aligned}
I_W &= \left[\tfrac{A_1(A_5Q-A_6R)}{D_2}\right] I_1 + \left[\tfrac{A_3(A_6R_1-A_4Q)+A_2(A_5R_1-A_4R)}{D_2}\right] I_2 - \left[\tfrac{A_2(A_6R_1+A_4Q)+A_3(A_5R_1-A_4R)}{D_2}\right] I_3 \\
I_{Wk} &= \left(\tfrac{A_5Q-A_6R}{D_2}\right) I_1 - \left(\tfrac{A_4Q-A_6R_1}{D_2}\right) I_2 + \left(\tfrac{A_4R-A_5R_1}{D_2}\right) I_3 \\
I_G &= \left(\tfrac{A_1A_2}{EID_1}\right)(I_5 - I_4) + \left[\tfrac{A_3(A_1-A_3)A_2^2}{EID_1}\right] I_6 \\
\bar{I}_W &= \left[\tfrac{A_1(A_5Q-A_6R)}{D_2}\right] I_1 + \left[\tfrac{A_3(A_6R_1-A_4Q)+A_2(A_5R_1-A_4R)}{D_2}\right] I_2 - \left[\tfrac{A_2(A_6R_1+A_4Q)+A_3(A_5R_1-A_4R)}{D_2}\right] I_3 \\
\bar{I}_{Wk} &= \left(\tfrac{A_5Q-A_6R}{D_2}\right) \bar{I}_1 - \left(\tfrac{A_4Q-A_6R_1}{D_2}\right) \bar{I}_2 + \left(\tfrac{A_4R-A_5R_1}{D_2}\right) \bar{I}_3 \\
\bar{I}_G &= \left(\tfrac{A_1A_2}{EID_1}\right)(\bar{I}_5 - \bar{I}_4) + \left[\tfrac{A_3(A_1-A_3)-A_2^2}{EID_1}\right] \bar{I}_6 \\
I_1 &= \bar{q}_a \left(\tfrac{m_{1a}-m_{1b}}{R_1}\right) + \left[\tfrac{q_a-q_b}{(b-a)L}\right](n_{1b} - n_{1a}) + \left[\tfrac{R_1L(am_{1a}-bm_{1b})}{R_1^2}\right] \\
I_2 &= \bar{q}_a \left[\tfrac{R(\bar{m}_a m_a - \bar{m}_b m_b) + Q(\bar{n}_a n_a - \bar{n}_b n_b)}{R^2+Q^2}\right] \\
&\quad + \left[\tfrac{q_a-q_b}{(b-a)L}\right]\left[\tfrac{1}{(R^2+Q^2)^2}\right]\Big\{(R^2+Q^2)\big[aL(R\bar{m}_a m_a + Q\bar{n}_a n_a) - bL(R\bar{m}_b m_b + Q\bar{n}_b n_b) \\
&\quad + 2RQ(\bar{n}_b m_b - \bar{n}_a m_a) + (R^2-Q^2)(n_b\bar{m}_b - n_a\bar{m}_a)\big]\Big\} \\
I_3 &= \bar{q}_a \left[\tfrac{R(\bar{n}_a n_a - \bar{n}_b n_b) - Q(\bar{m}_a m_a - m_b\bar{m}_b)}{R^2+Q^2}\right] \\
&\quad + \left[\tfrac{q_a-q_b}{(b-a)L}\right]\left[\tfrac{1}{(R^2+Q^2)^2}\right]\Big[(R^2+Q^2)\big[aL(R\bar{n}_a n_a - Q\bar{m}_a m_a) + bL(Q\bar{m}_b m_b - R\bar{n}_b n_b) \\
&\quad + 2RQ(n_a\bar{m}_a - n_b - \bar{m}_b) + (R^2-Q^2)(\bar{n}_b m_b - \bar{n}_a m_a)\big]\Big] \\
I_4 &= \bar{q}_a \left(\tfrac{n_{1b}-n_{1a}}{R_1}\right) + \left[\tfrac{q_a-q_b}{(b-a)L}\right]\left[\tfrac{(m_{1a}-m_{1b})+R_1L(bn_{1b}-an_{1a})}{R_1^2}\right] \\
I_5 &= \bar{q}_a \left[\tfrac{Q(m_b\bar{n}_b - m_a\bar{n}_a) + R(\bar{m}_b n_b - \bar{m}_a n_a)}{R^2+Q^2}\right] \\
&\quad + \left[\tfrac{q_a-q_b}{(b-a)L}\right]\left[\tfrac{1}{(R^2+Q^2)^2}\right]\Big\{(R^2+Q^2)\big[-aL(R\bar{m}_a n_a + Q\bar{m}_a n_a) + bL(R\bar{m}_b n_b + Q\bar{n}_b m_b) \\
&\quad + (R^2-Q^2)(m_a\bar{m}_a - m_b\bar{m}_b) + 2RQ(n_a\bar{n}_a - n_b\bar{n}_b)\big]\Big\} \\
I_6 &= \bar{q}_a \left[\tfrac{Q(\bar{m}_a n_a - \bar{m}_b n_b) + R(m_b\bar{n}_b - m_a\bar{n}_a)}{R^2+Q^2}\right] \\
&\quad + \left[\tfrac{q_a-q_b}{(b-a)L}\right]\left[\tfrac{1}{(R^2+Q^2)^2}\right]\Big\{(R^2+Q^2)\big[aL(Q\bar{m}_a n_a - R\bar{n}_a m_a) + bL(R\bar{n}_b m_b - Q\bar{m}_b n_b) \\
&\quad + (R^2-Q^2)(\bar{n}_a n_a - \bar{n}_b n_b) + 2RQ(m_b\bar{m}_b - m_a\bar{m}_a)\big]\Big\}
\end{aligned}
$$

Terms

$$
\begin{aligned}
\bar{q}_a &= q_a - \left[\tfrac{q_b - q_a}{(b-a)L}\right] \\
m_{1a} &= \cosh[R_1 L(a-1)] \\
n_{1a} &= \sinh[R_1 L(a-1)] \\
m_a &= \cosh[RL(a-1)] \\
n_a &= \sinh[RL(a-1)] \\
m_{1b} &= \cosh[R_1 L(b-1)] \\
n_{1b} &= \sinh[R_1 L(b-1)] \\
m_b &= \cosh[RL(b-1)] \\
n_b &= \sinh[RL(b-1)] \\
\bar{n}_a &= \sin[QL(a-1)] \\
\bar{n}_b &= \sin[QL(b-1)] \\
\bar{m}_a &= \cos[QL(a-1)] \\
\bar{m}_b &= \cos[QL(b-1)]
\end{aligned}
$$

The above relations for integrals $I_1 - I_6$ can also be used for the calculation of integrals $\bar{I}_1 - \bar{I}_6$, provided that the terms (a-1) and (b-1) are replaced by (-a) and (-b) Morfidis 2007.

B

Appendix B

The following figure shows the generation of excess pore pressures after a certain time in days and gives an indication of the location of permeable and impermeable boundaries in the soil profiles.

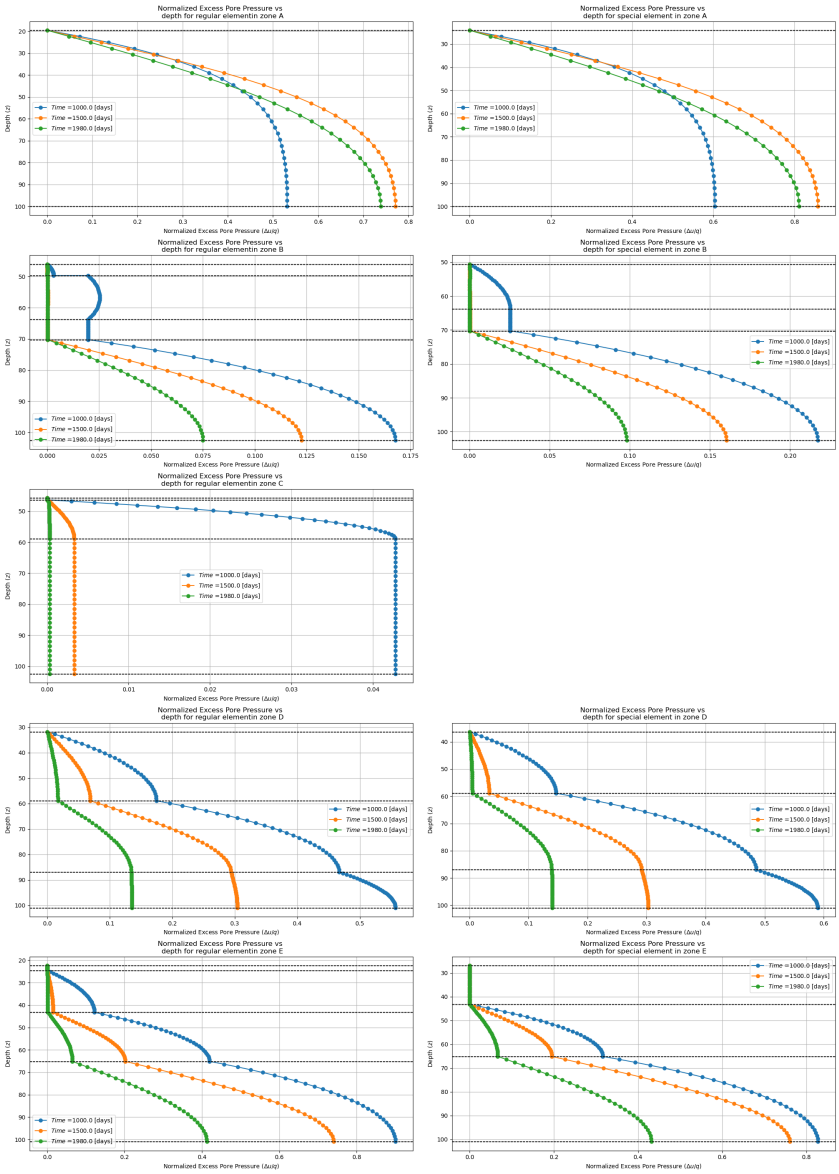


Figure B.1: Excess pore pressure profile per zone and time

Appendix C

The ranges in values of the primary consolidation are shown in figures C.1, C.2, C.3 and C.4, and with standard deviations for the layer thicknesses of 0.5, 1, 2 and 5 meters respectively.

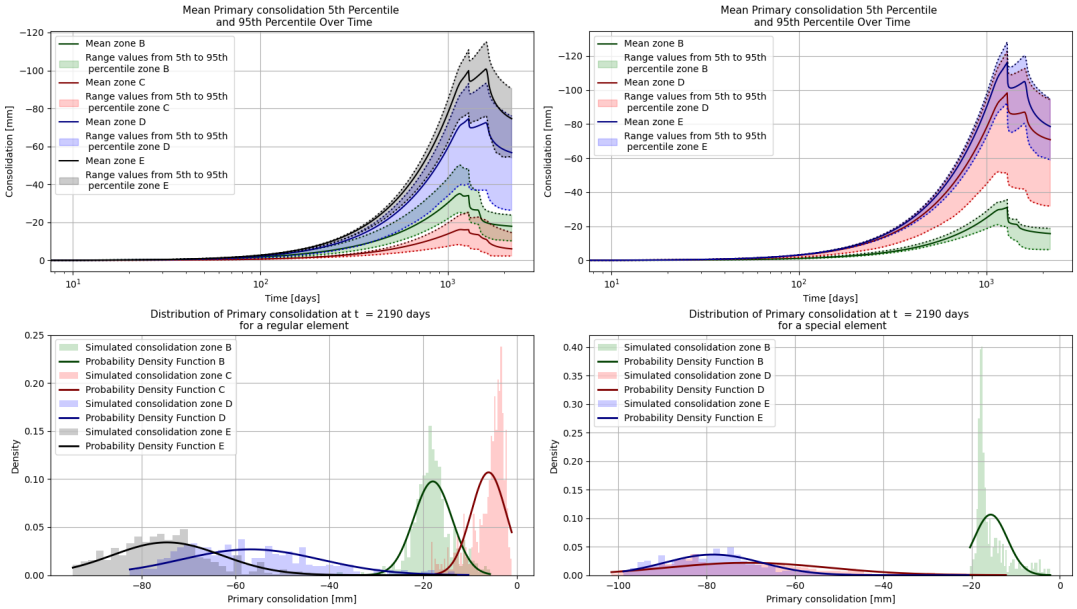


Figure C.1: Range of primary consolidation per defined zone for a standard deviation of 0.5 meter

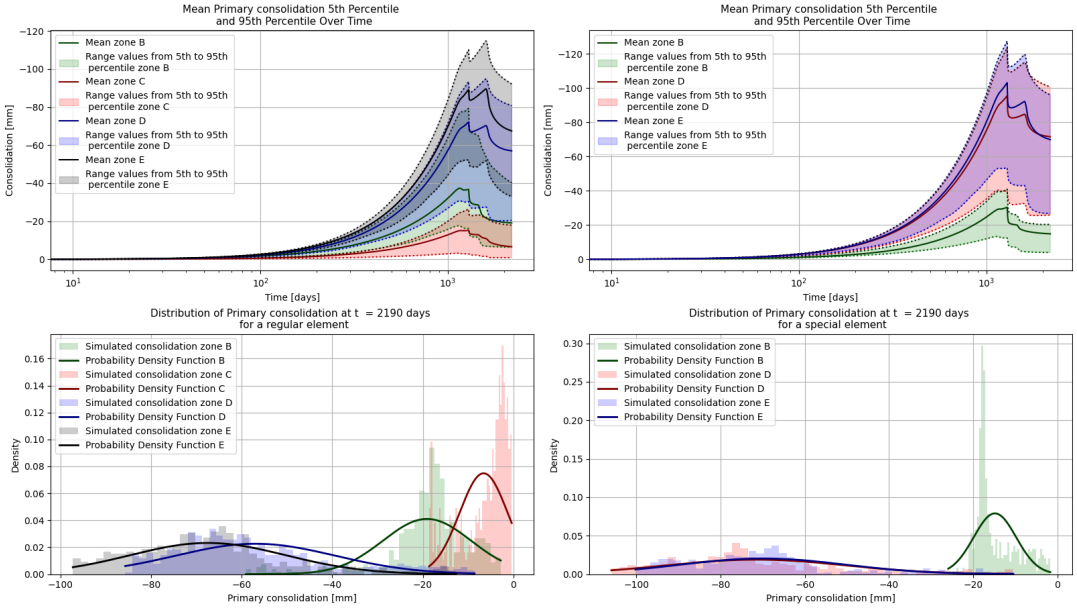


Figure C.2: Range of primary consolidation per defined zone for a standard deviation of 1 meter

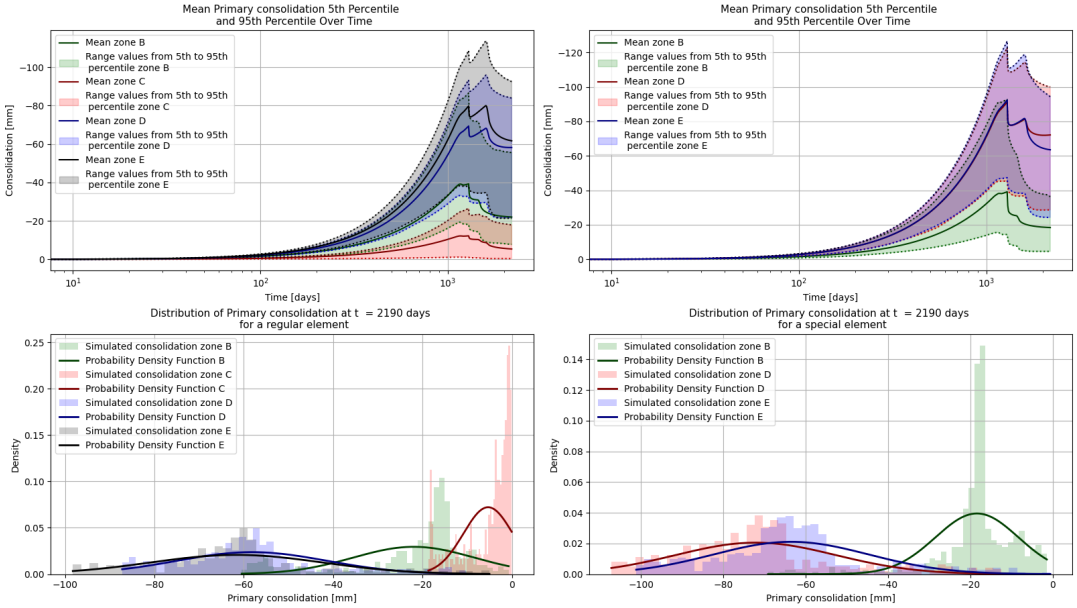


Figure C.3: Range of primary consolidation per defined zone for a standard deviation of 2 meter

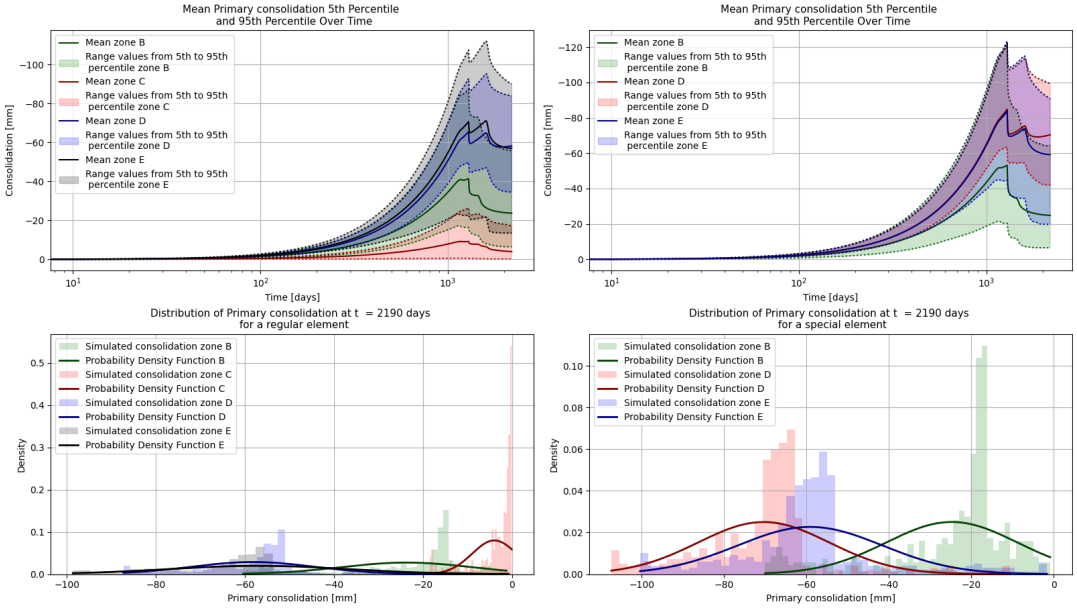


Figure C.4: Range of primary consolidation per defined zone for a standard deviation of 5 meter

The ranges in values of the secondary consolidation are shown in figures C.5, C.6, C.7 and C.8, with standard deviations for the layer thicknesses of 0.5, 1, 2 and 5 meters respectively.

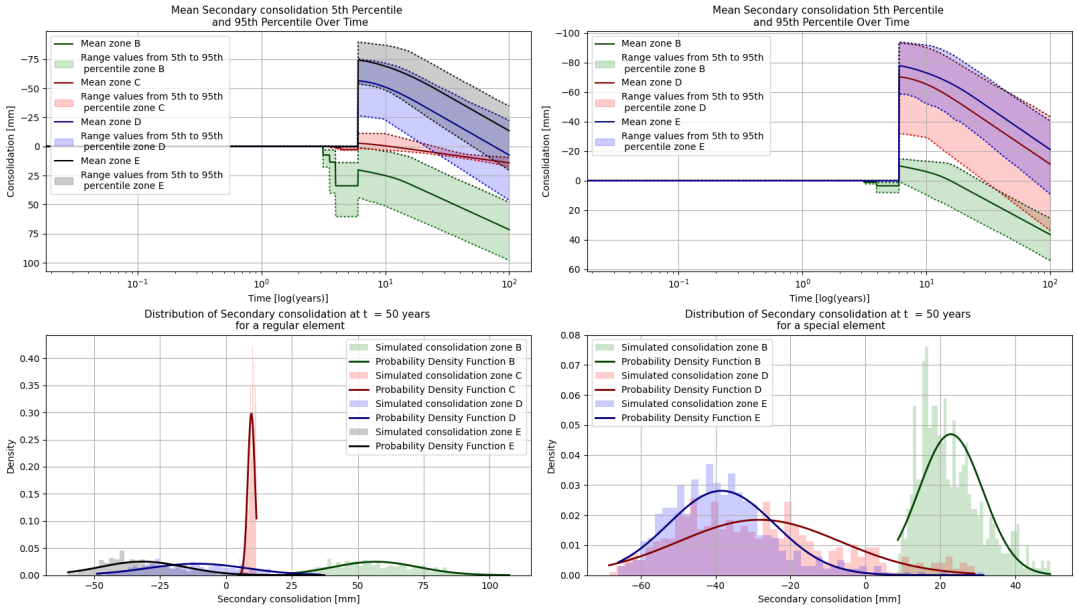


Figure C.5: Range of secondary consolidation per defined zone for a standard deviation of 0.5 meter

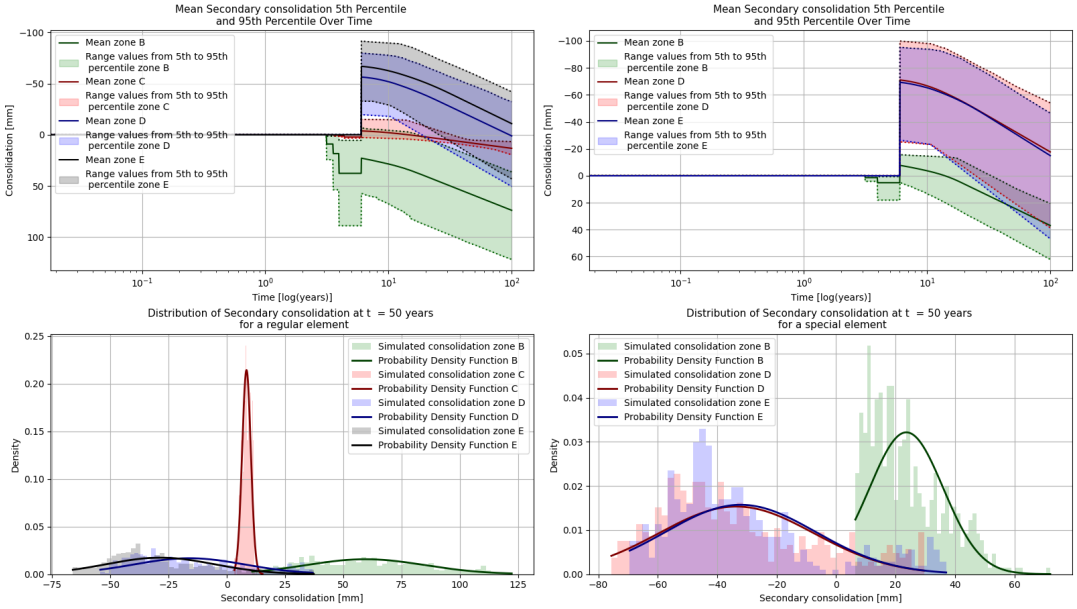


Figure C.6: Range of secondary consolidation per defined zone for a standard deviation of 1 meter

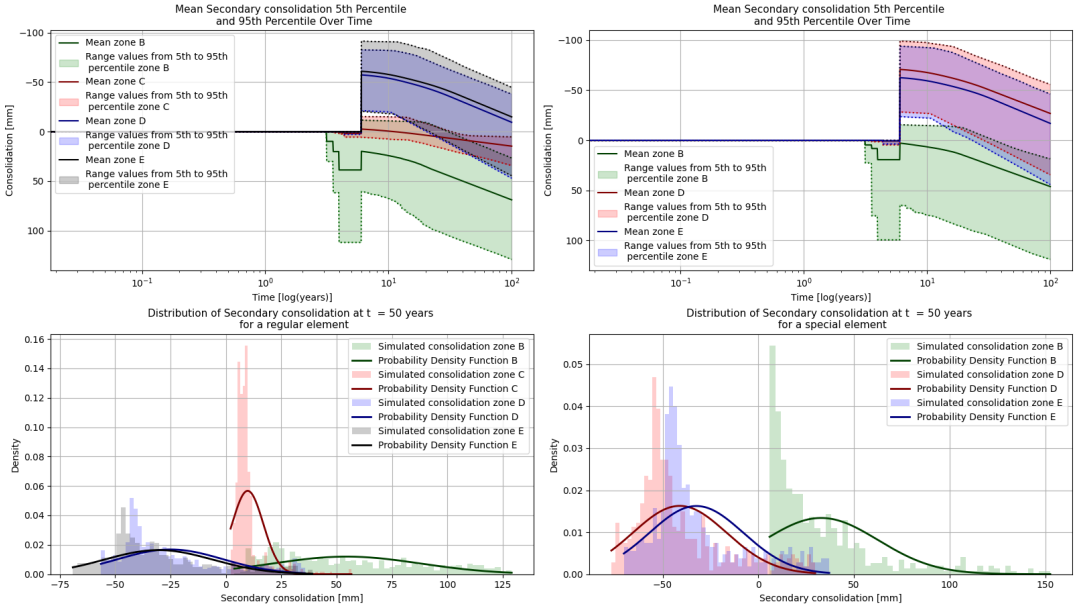


Figure C.7: Range of secondary consolidation per defined zone for a standard deviation of 2 meter

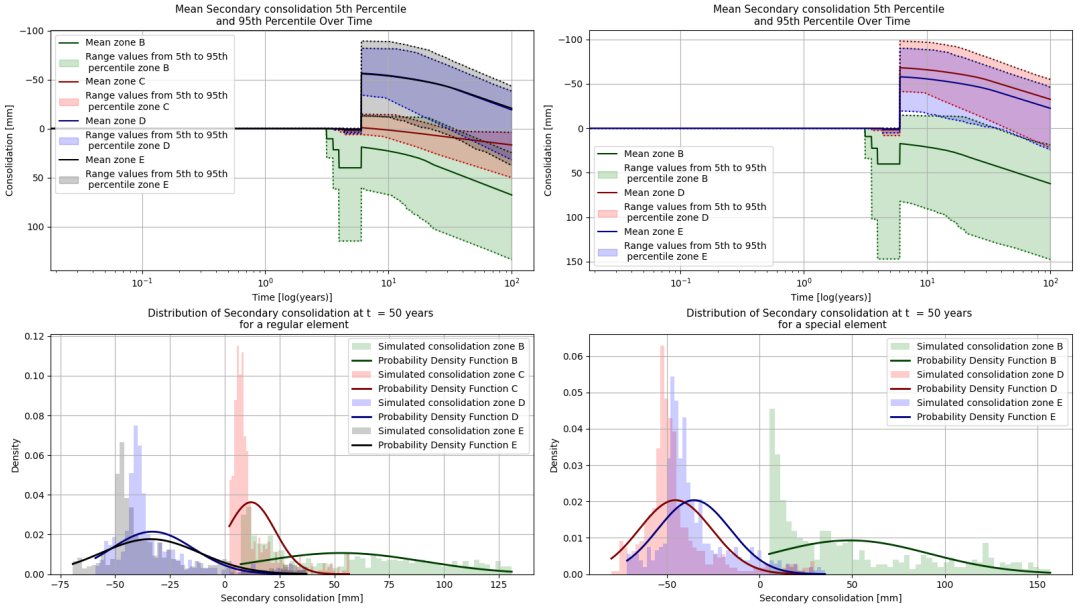


Figure C.8: Range of secondary consolidation per defined zone for a standard deviation of 5 meter

The ranges in values of the total deformation of the entire tunnel are shown in figures below (ordered per zone), with standard deviations for the layer thicknesses of 0.5, 1, 2, and 5 meters respectively.

A. Zone B

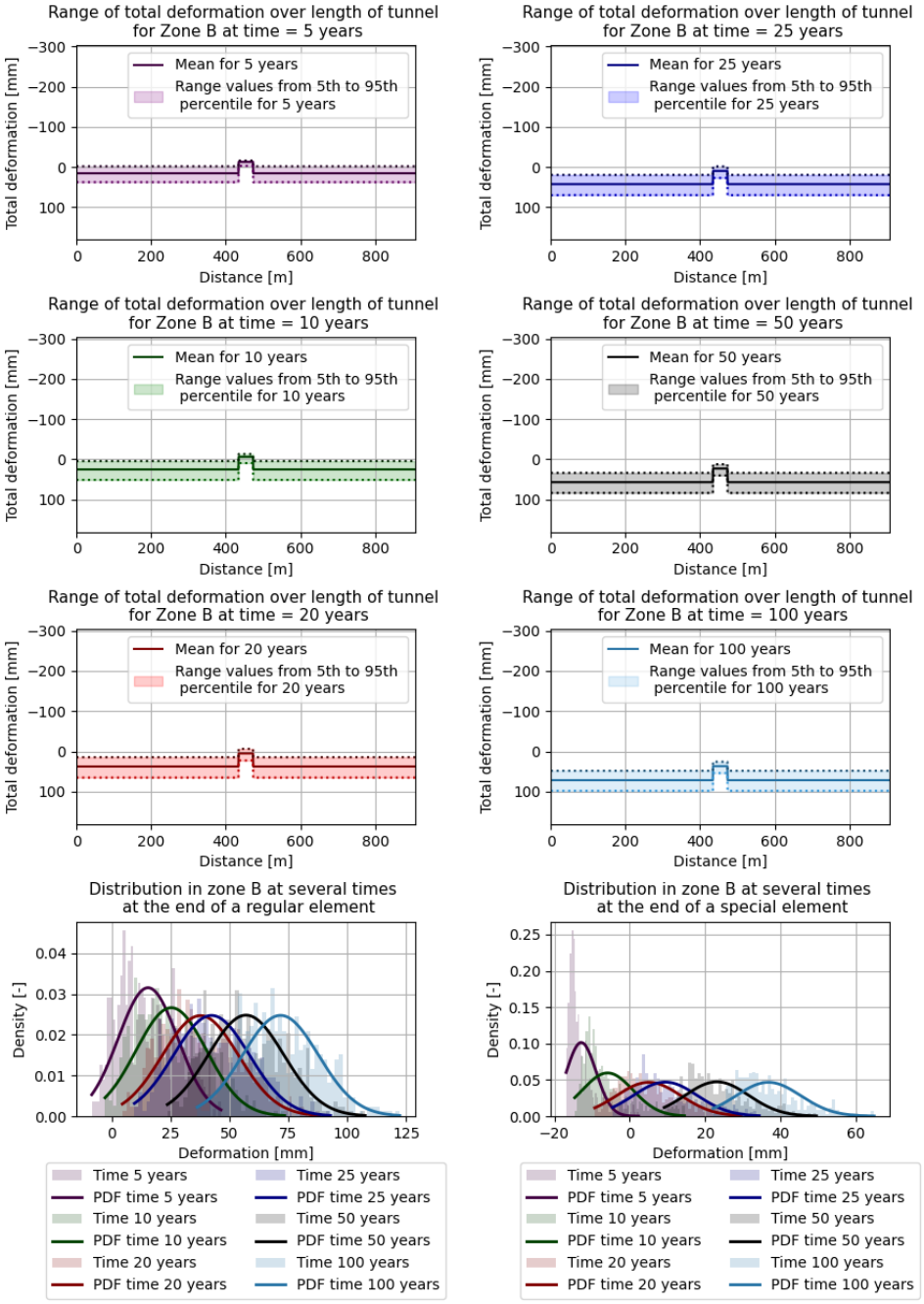


Figure C.9: Range of total deformation entire length of tunnel in Zone B with standard deviation of 0.5 meter

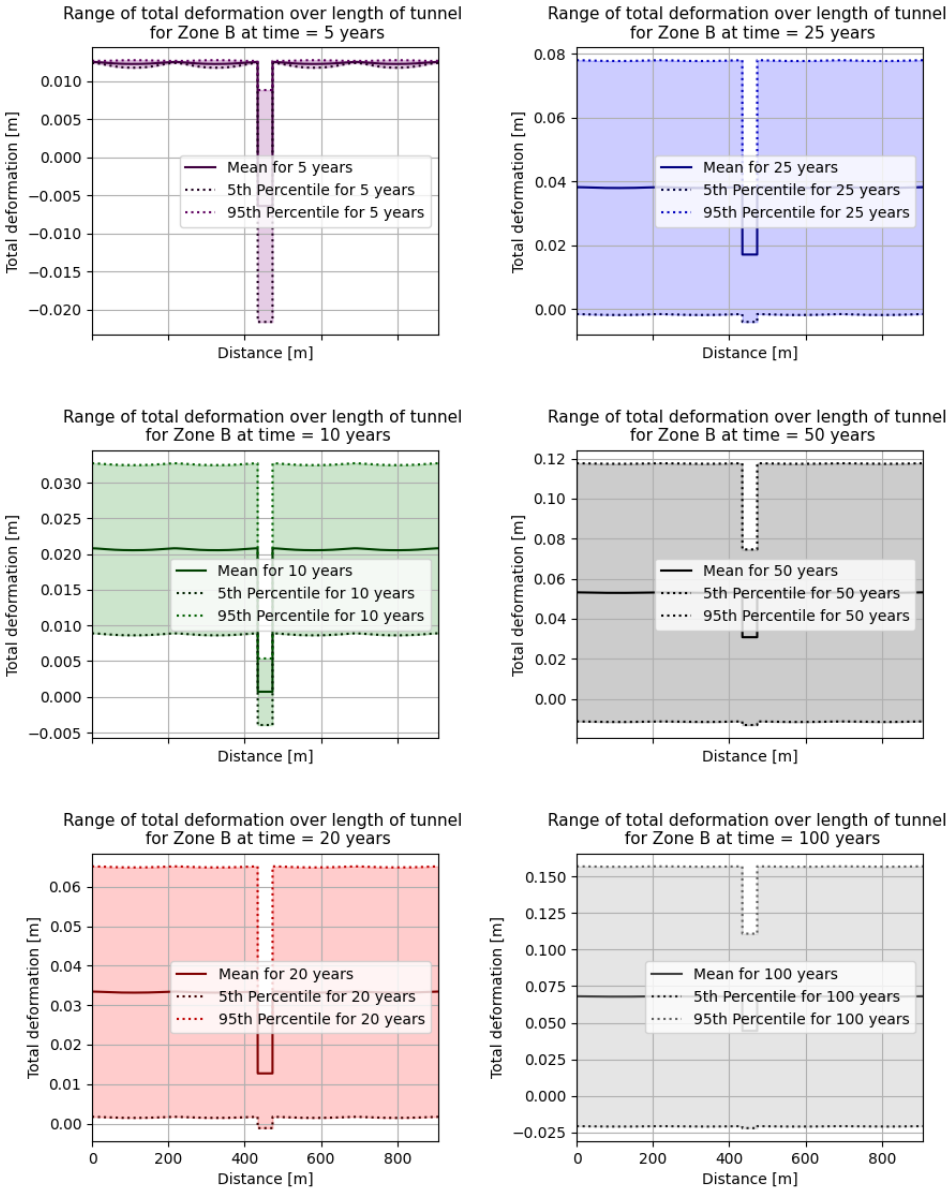


Figure C.10: Range of total deformation entire length of tunnel in Zone B with standard deviation of 1 meter

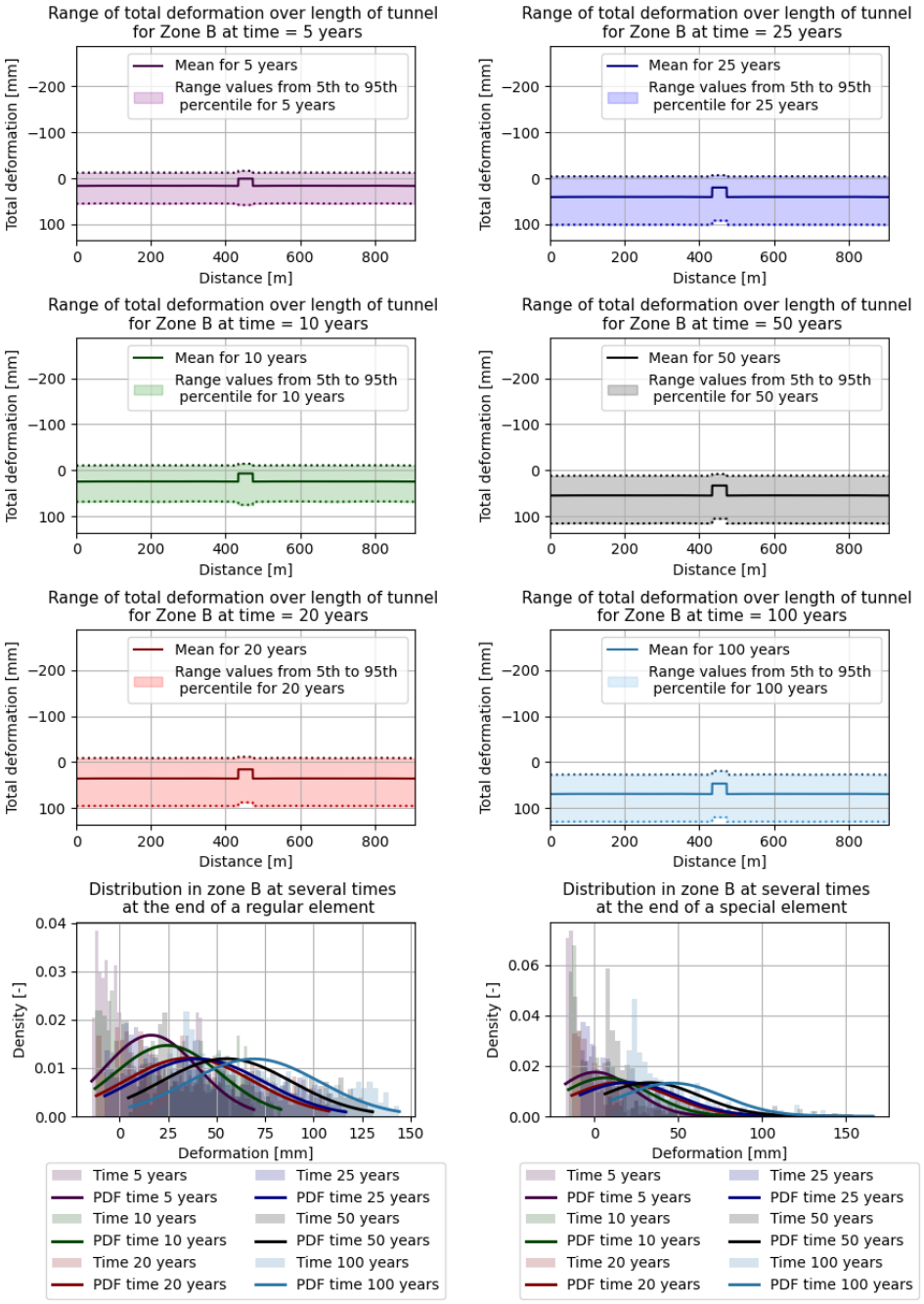


Figure C.11: Range of total deformation entire length of tunnel in Zone B with standard deviation of 2 meter

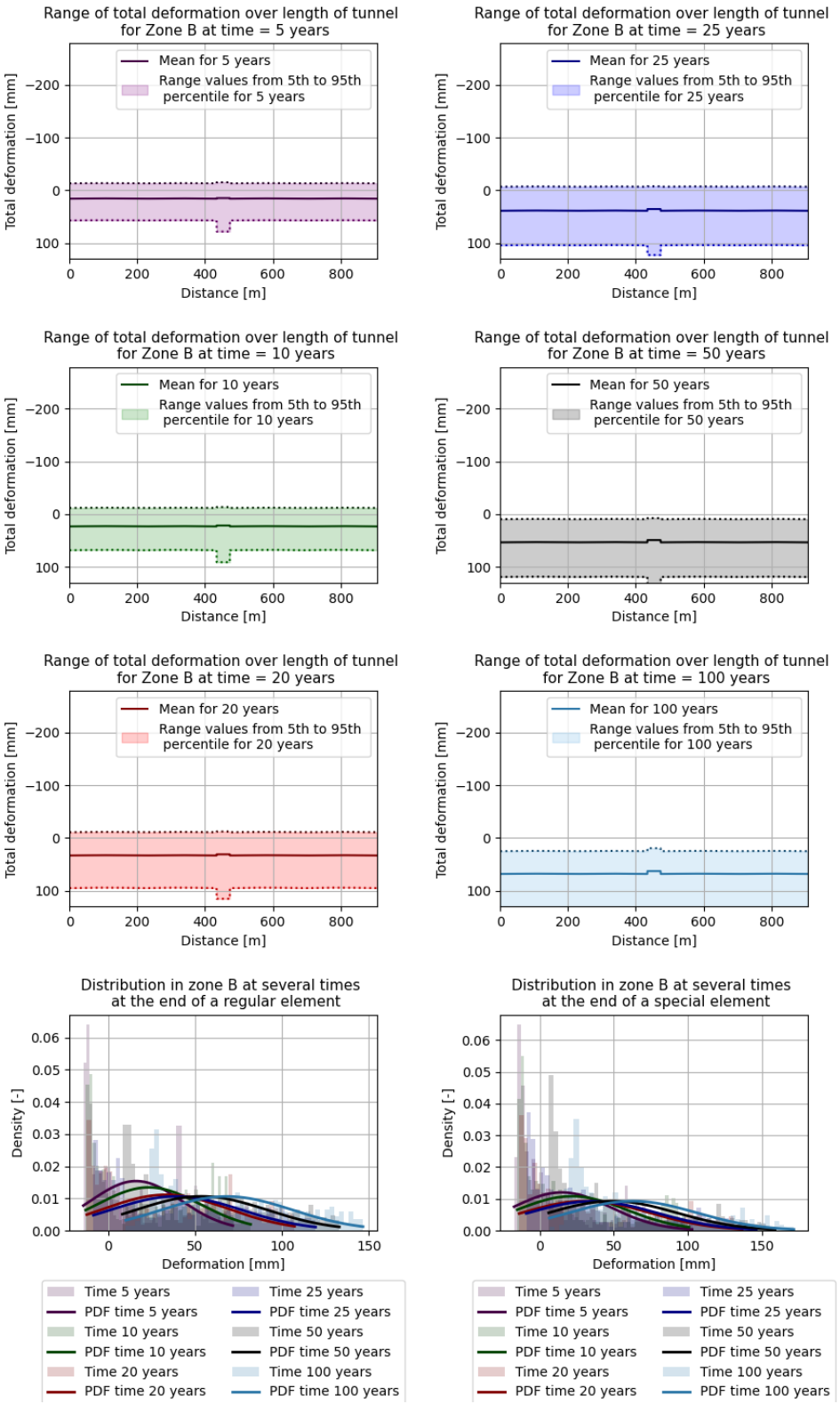


Figure C.12: Range of total deformation entire length of tunnel in Zone B with standard deviation of 5 meter

B. Zone C

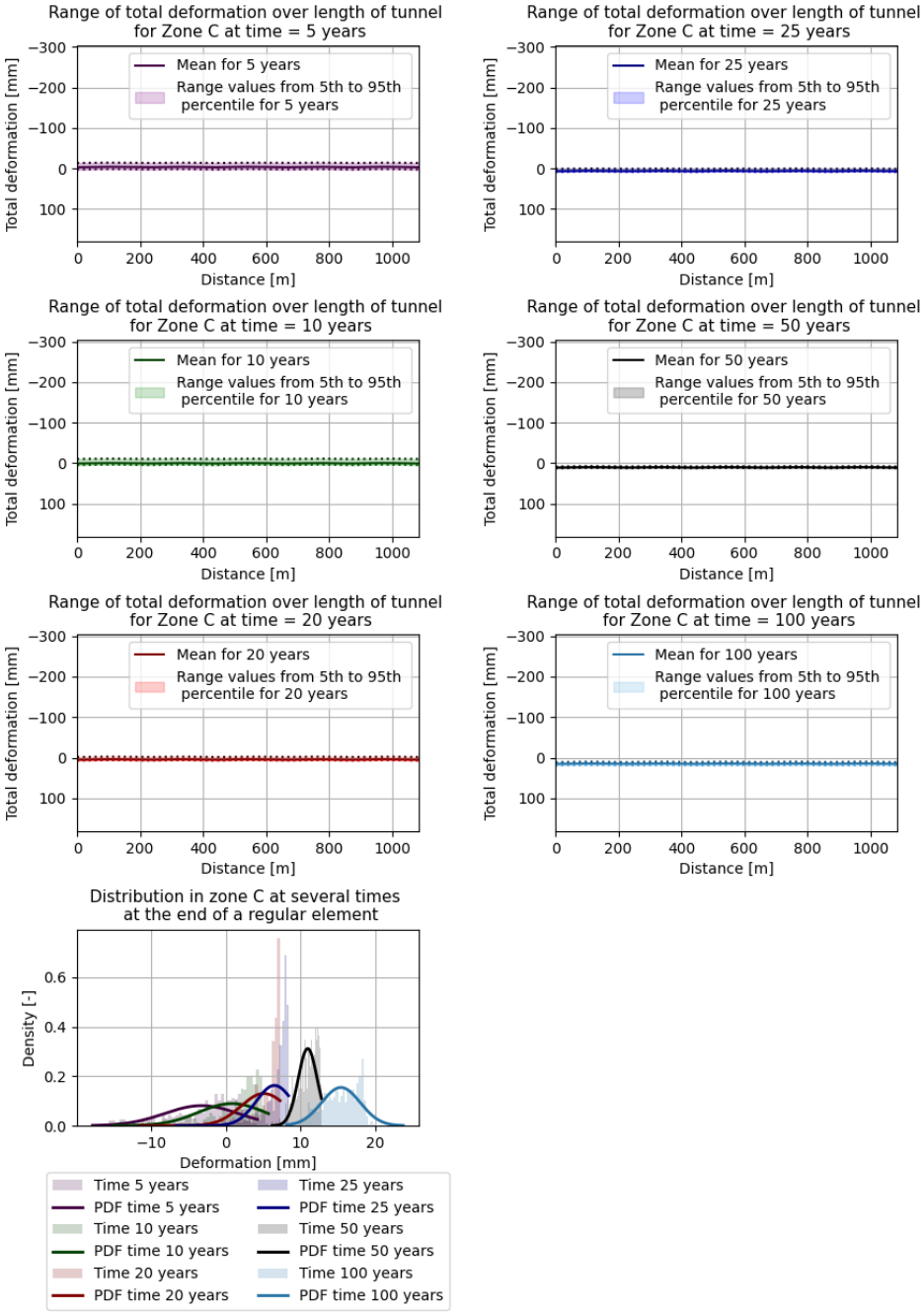


Figure C.13: Range of total deformation entire length of tunnel in Zone C with standard deviation of 0.5 meter

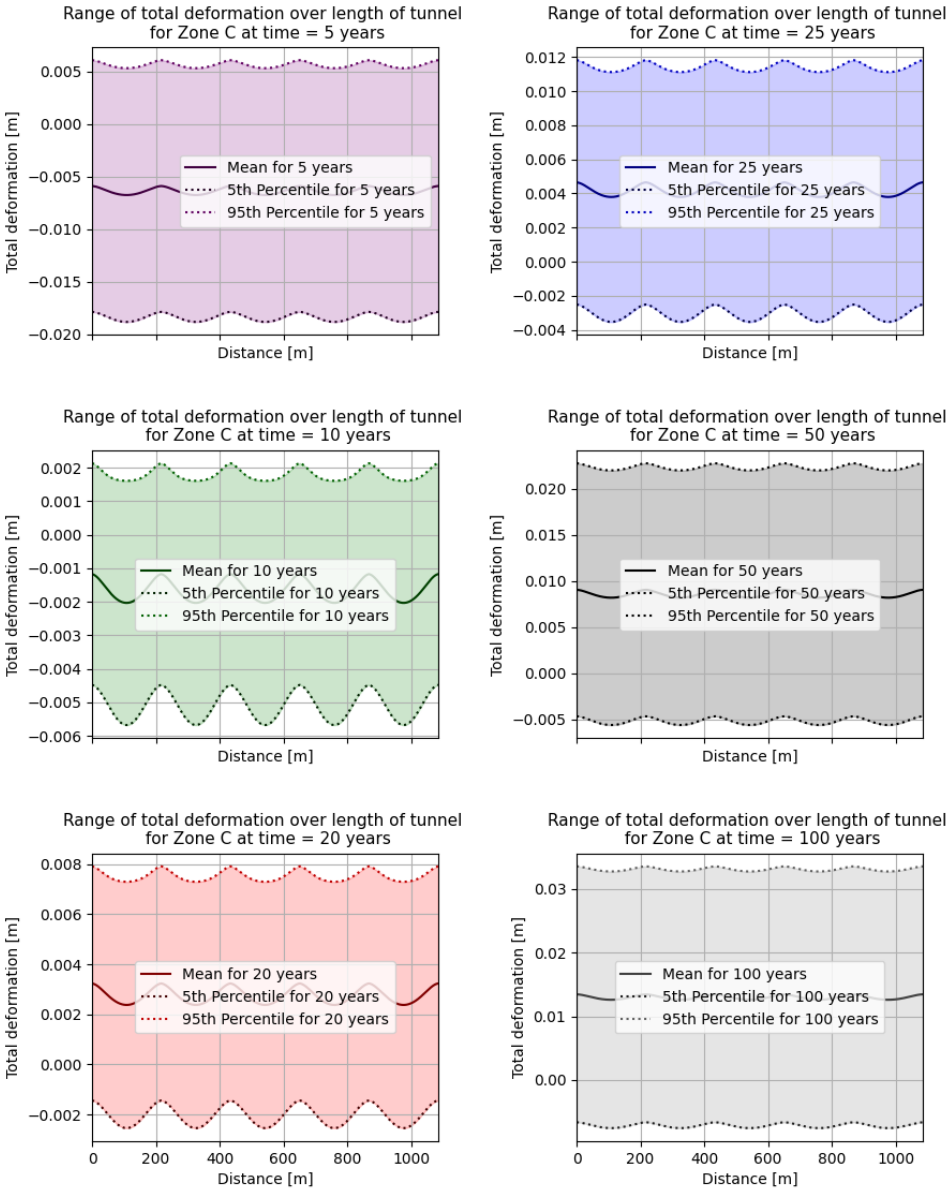


Figure C.14: Range of total deformation entire length of tunnel in Zone C with standard deviation of 1 meter

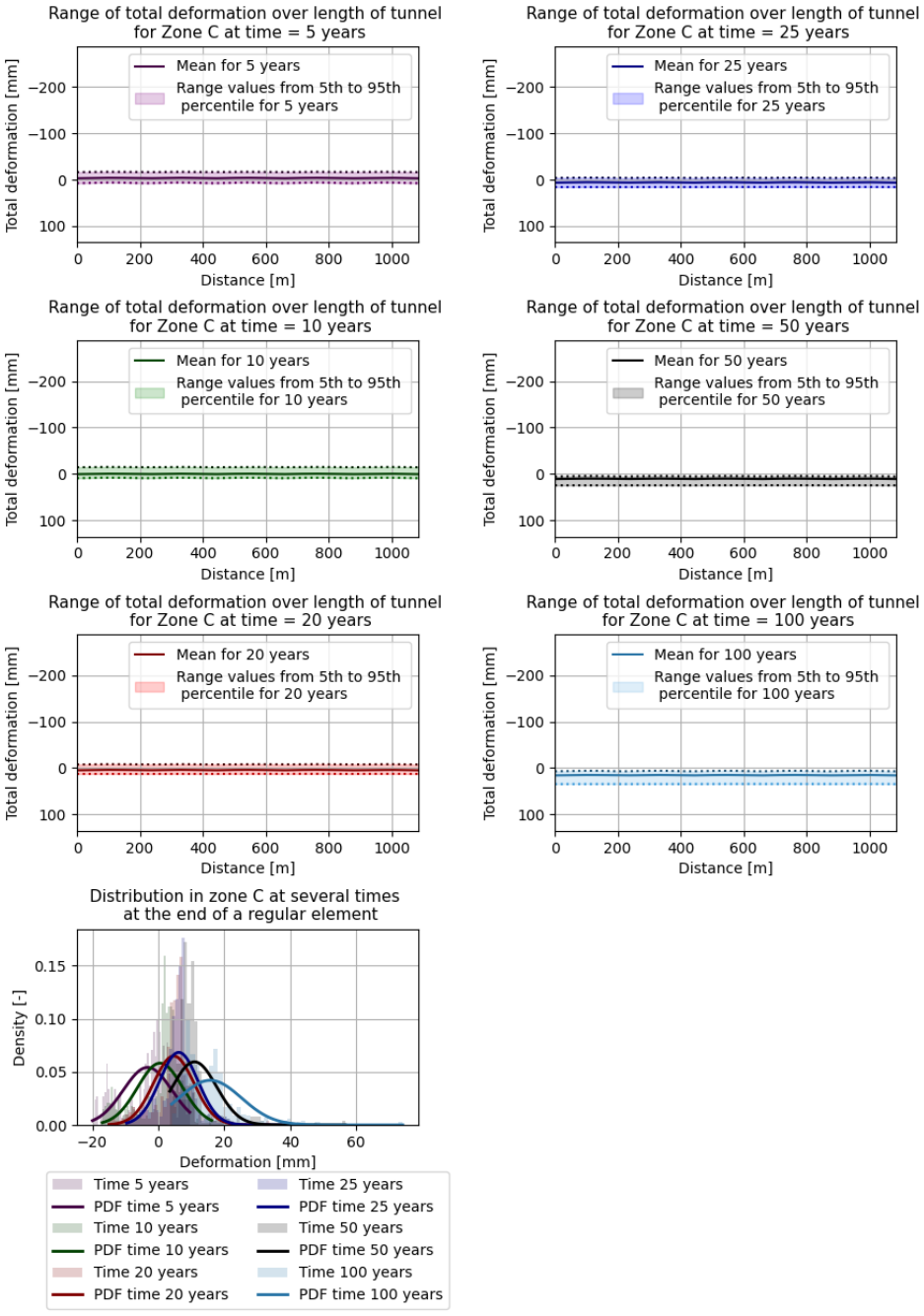


Figure C.15: Range of total deformation entire length of tunnel in Zone C with standard deviation of 2 meter

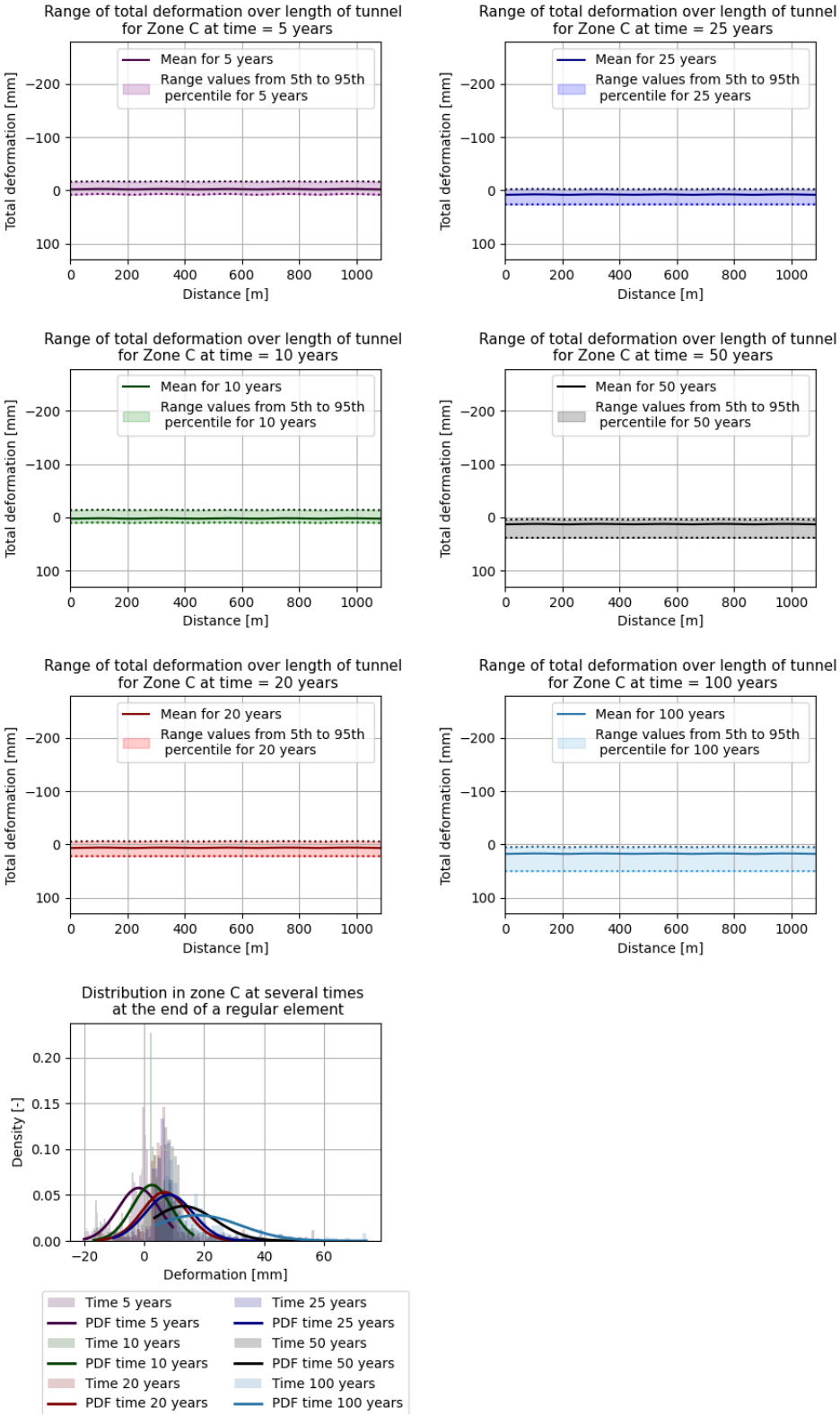


Figure C.16: Range of total deformation entire length of tunnel in Zone C with standard deviation of 5 meter

C. Zone D

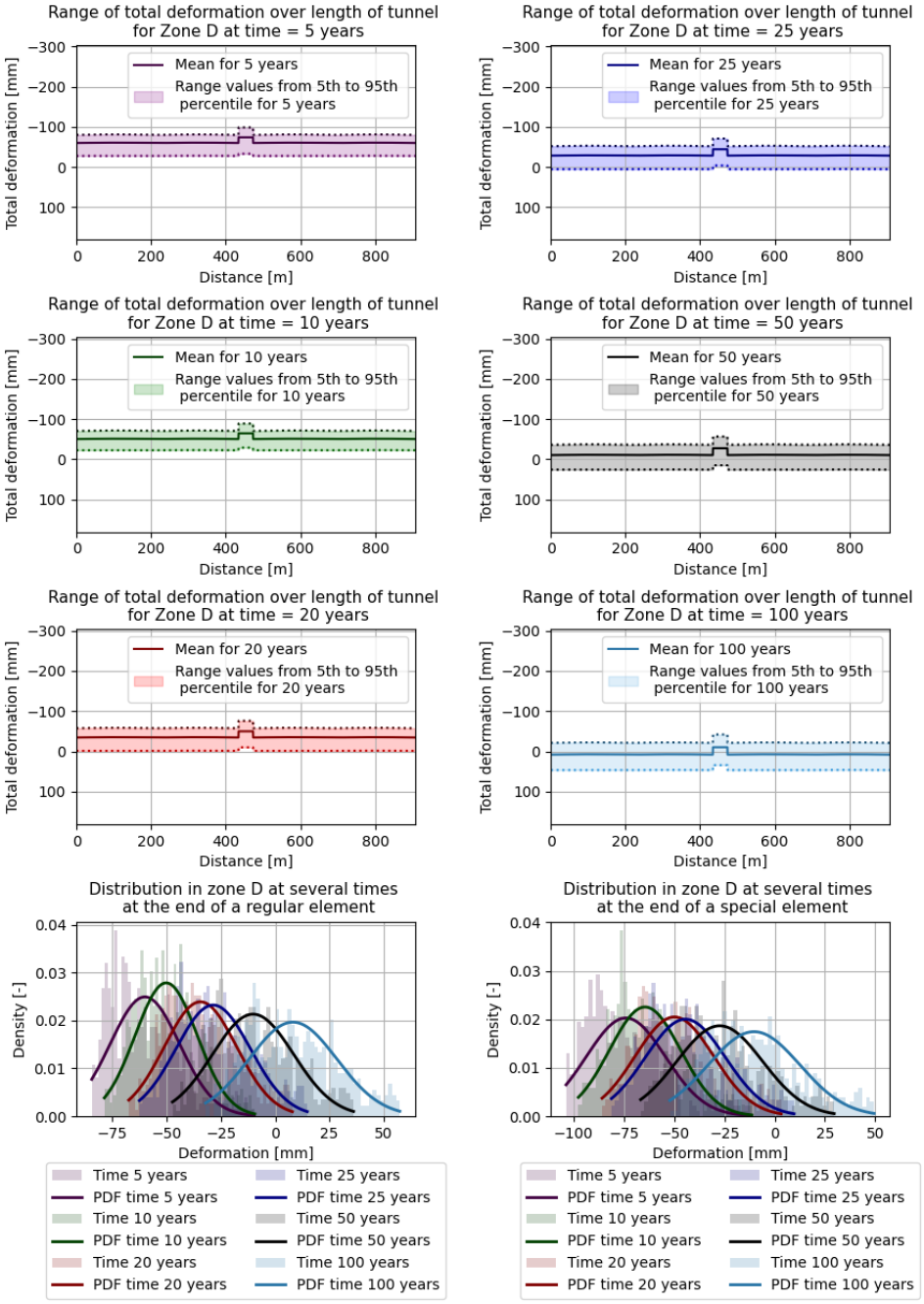


Figure C.17: Range of total deformation entire length of tunnel in Zone D with standard deviation of 0.5 meter

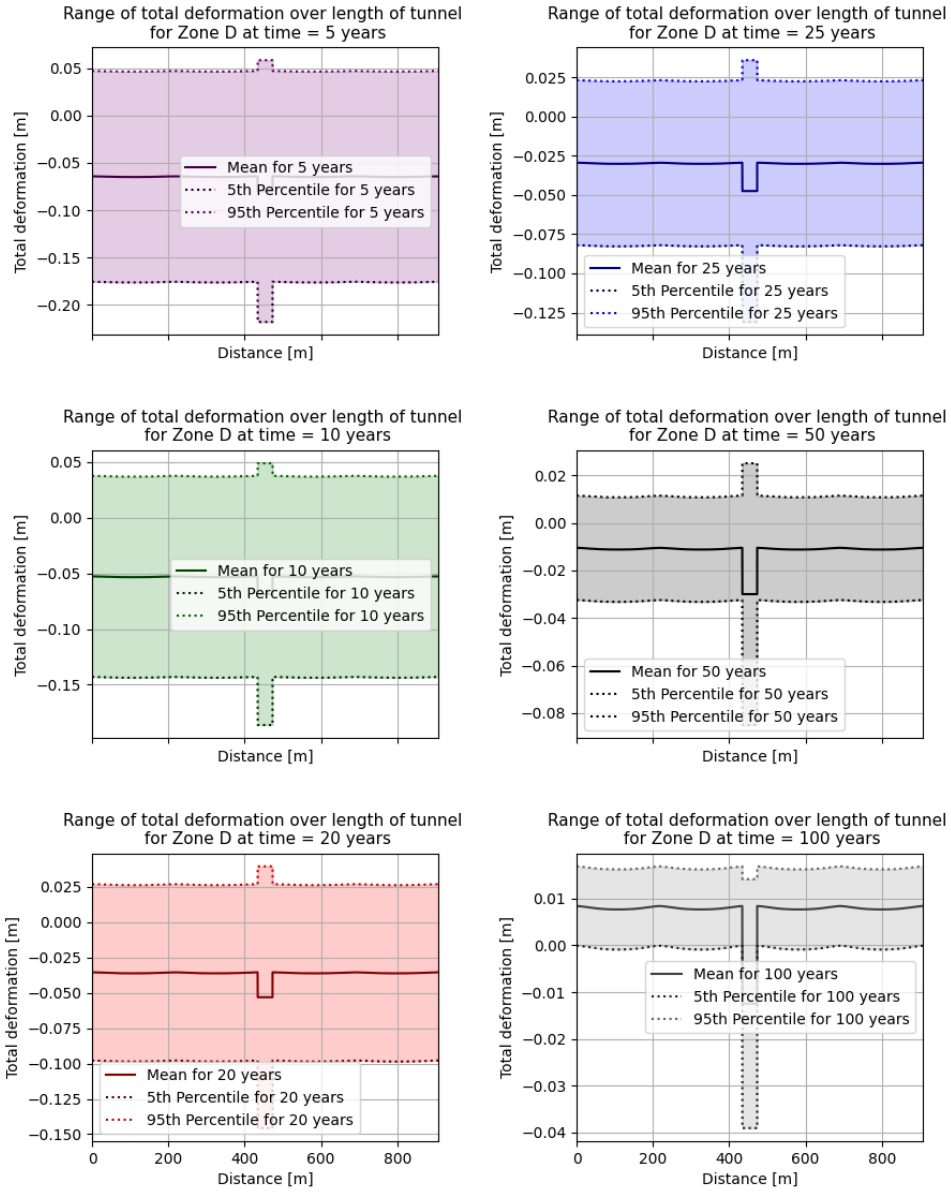


Figure C.18: Range of total deformation entire length of tunnel in Zone D with standard deviation of 1 meter

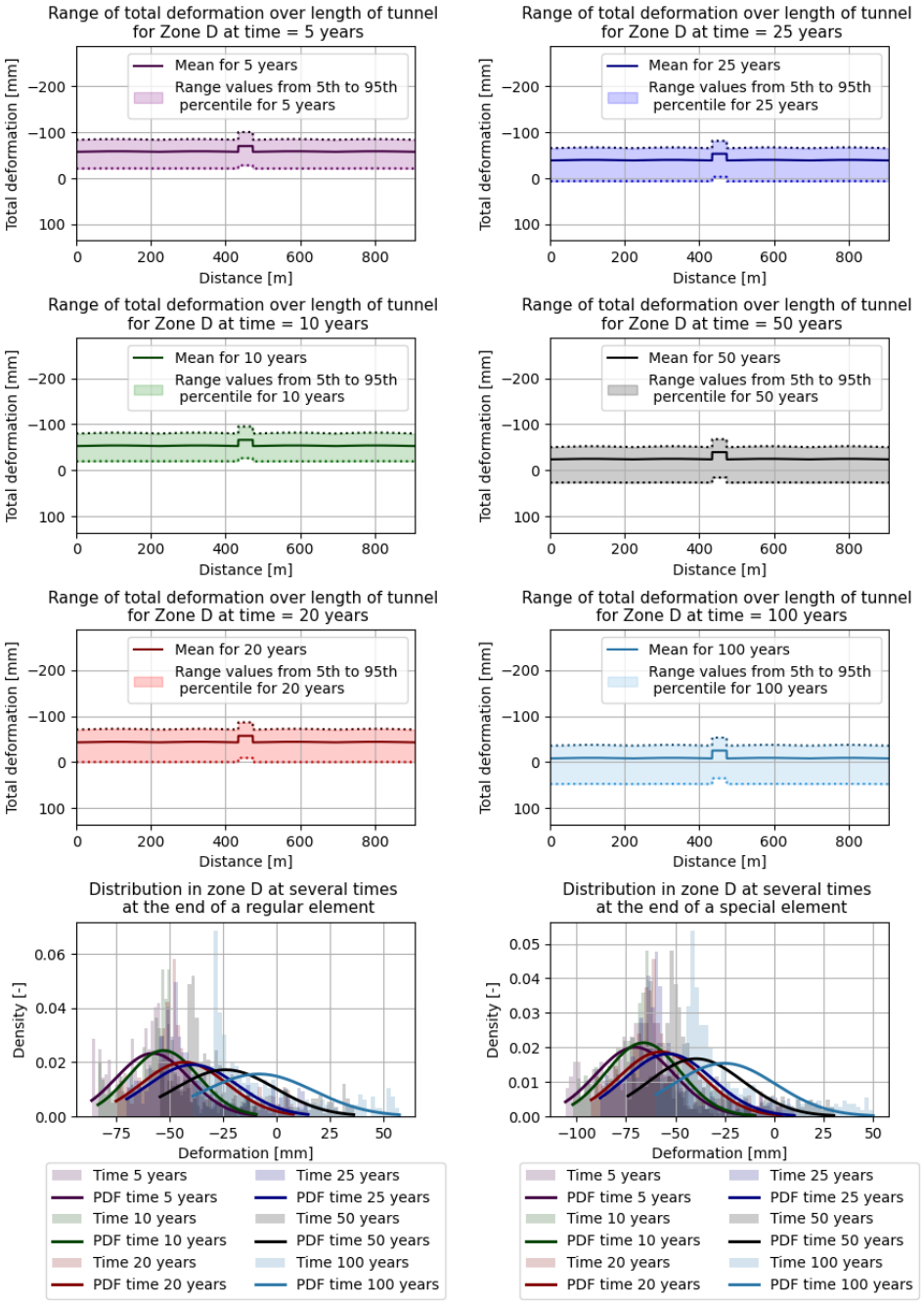


Figure C.19: Range of total deformation entire length of tunnel in Zone D with standard deviation of 2 meter

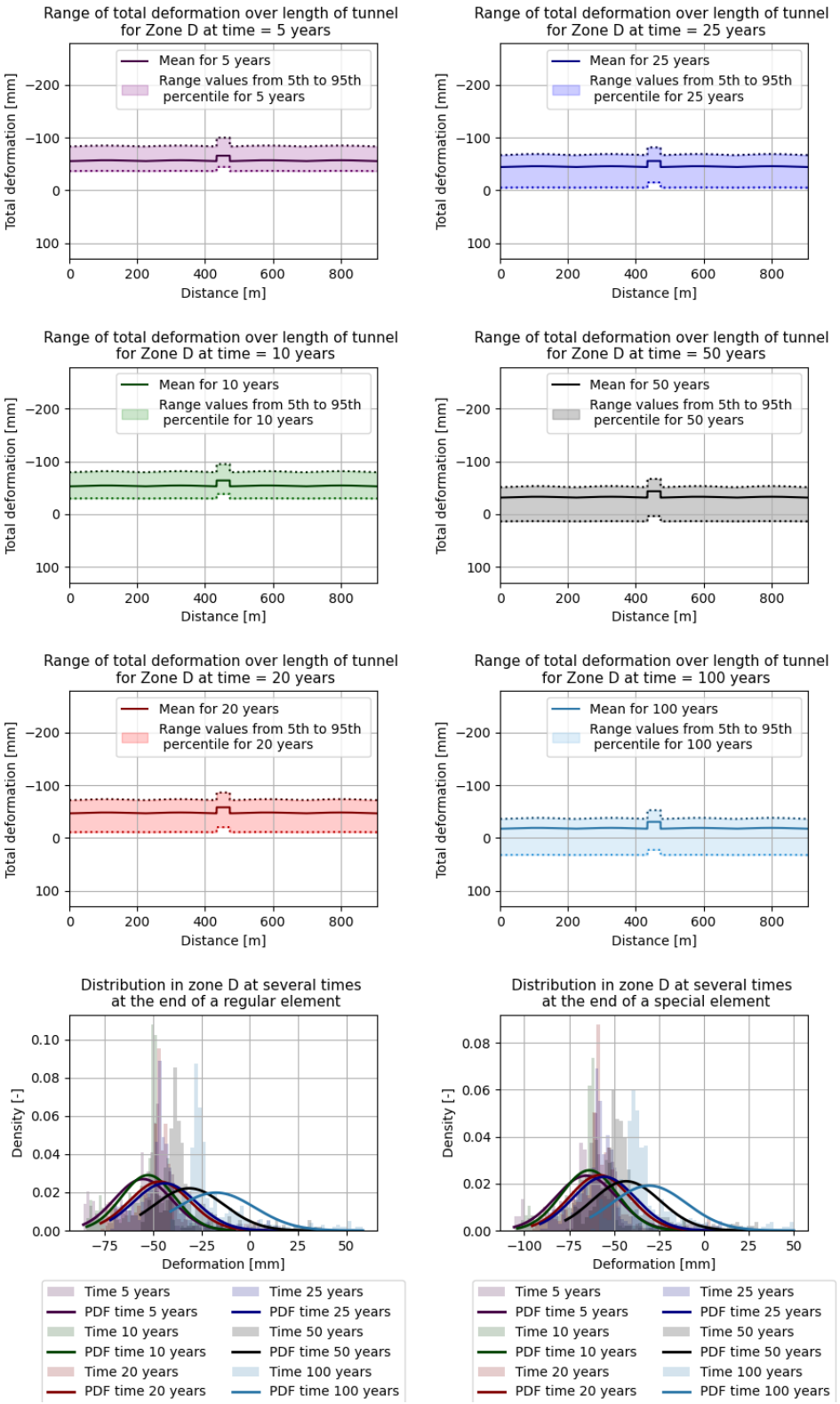


Figure C.20: Range of total deformation entire length of tunnel in Zone D with standard deviation of 5 meter

D. Zone E

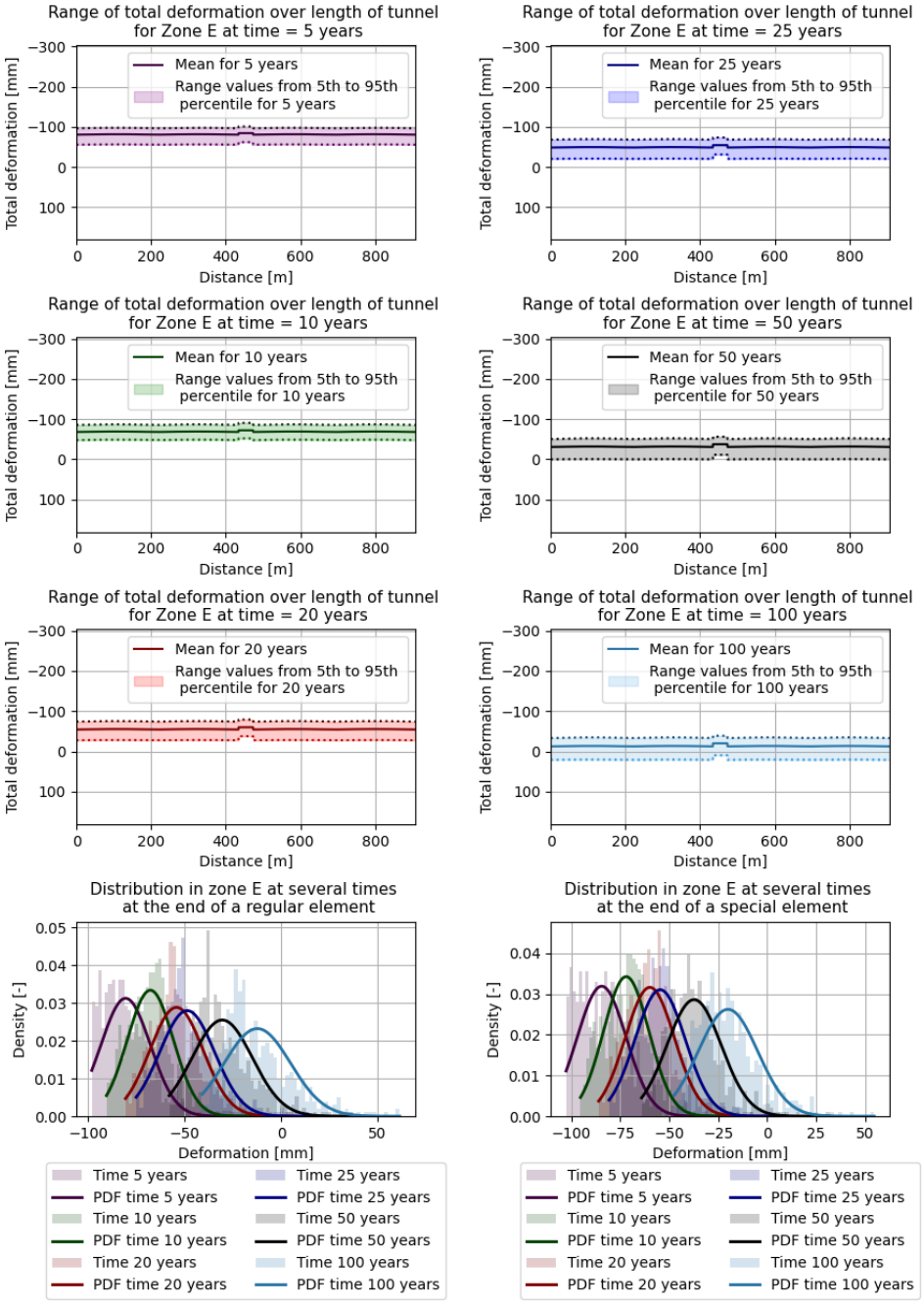


Figure C.21: Range of total deformation entire length of tunnel in Zone E with standard deviation of 0.5 meter

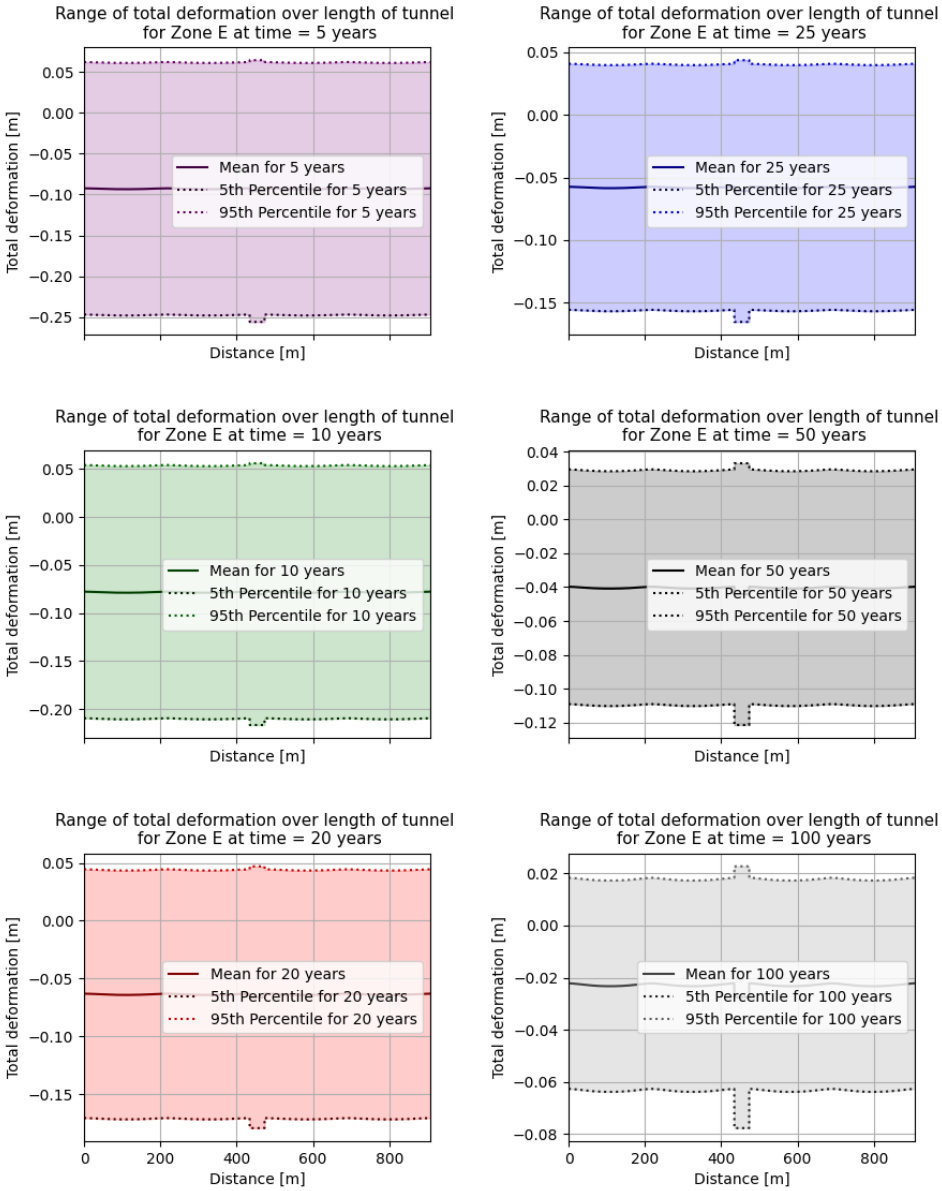


Figure C.22: Range of total deformation entire length of tunnel in Zone E with standard deviation of 1 meter

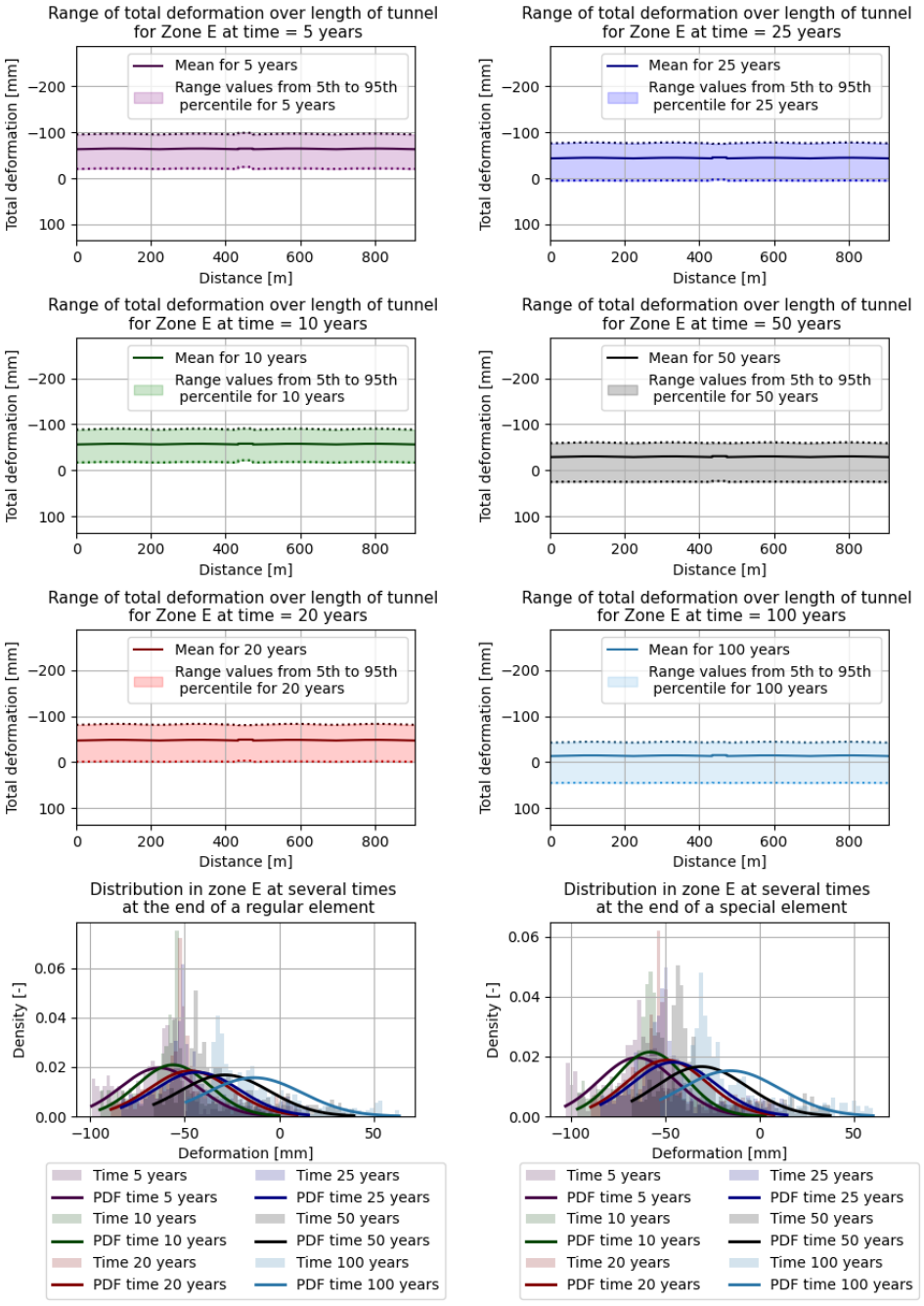


Figure C.23: Range of total deformation entire length of tunnel in Zone E with standard deviation of 2 meter

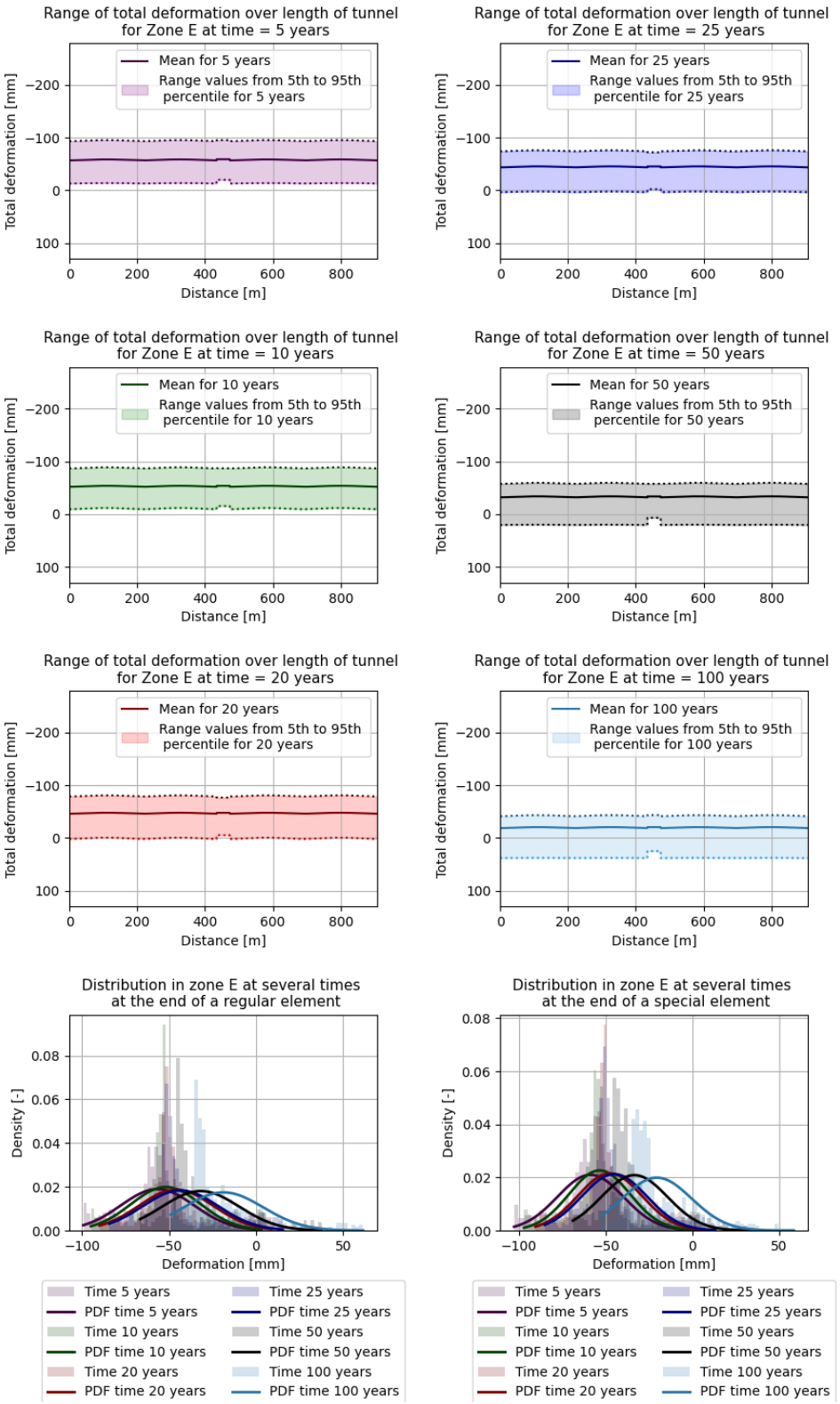


Figure C.24: Range of total deformation entire length of tunnel in Zone E with standard deviation of 5 meter

D

Equations

$$e = V_v / V_s \tag{2.1}$$

$$n = \frac{e}{(1+e)} \tag{2.2}$$

$$\sigma = \frac{F_{ex}}{A} \tag{2.3}$$

$$\varepsilon = \frac{\Delta L}{L_0} \tag{2.4}$$

$$OCR = \frac{\sigma_p^{'}}{\sigma_0^{'}} \tag{2.5}$$

$$E_{oed} = \frac{\Delta \sigma}{\Delta \epsilon} \tag{2.6}$$

$$\frac{\delta u}{\delta t} = C_v * \frac{\delta^2 u}{\delta z^2} \tag{2.7}$$

$$C_v = \frac{k_h}{\gamma_w * m_v} \tag{2.8}$$

$$U = \frac{2}{\sqrt{\pi}} \sqrt{\frac{C_v t}{H_d^2}} \tag{2.9}$$

$$S_s = C_{\alpha e} * H * log\left(\frac{t_2}{t_1}\right) \tag{2.10}$$

$$C_c = \frac{e_1 - e_2}{log\sigma_2^{'} - log\sigma_1^{'}} = \frac{e_1 - e_2}{log\left(\frac{\sigma_2^{'}}{\sigma_1^{'}}\right)} \quad \text{(Das 2016)} \tag{2.11}$$

$$C_e = \frac{e_3 - e_4}{log\left(\frac{\sigma_4^{'}}{\sigma_3^{'}}\right)} \quad \text{(Das 2016)} \tag{2.12}$$

$$C_{\alpha e} = \frac{\Delta e}{log t_2 - log t_1} = \frac{\Delta e}{log \frac{t_2}{t_1}} \quad \text{(Das 2016)}$$

(2.13)

$$E_{eq} = \frac{\sum_{i-1}^{n} h_i E_{oed,i}}{\sum_{i-1}^{n} h_i} \tag{2.14}$$

$$\nu_s = \frac{\sum_{i-1}^{n} h_i \nu_{s,i}}{\sum_{i-1}^{n} h_i} \tag{2.15}$$

$$F_b = \rho * V_w * g \tag{4.1}$$

$$V(x) = \frac{dM(x)}{dx} \tag{4.2}$$

$$\frac{d^2M(x)}{dx^2} = -q(x) \tag{4.3}$$

$$M(x) = EI\kappa = -EI * \frac{d\phi(x)}{dx} = -EI\frac{d^2w}{dx^2} + EI\frac{d\gamma}{dx} \tag{4.4}$$

$$V(x) = GA\gamma/f_s = GA * \left(\frac{dw(x)}{dx} - \phi(x)\right)/f_s \tag{4.5}$$

$$\phi(x) = \frac{dw(x)}{dx} - \frac{1}{\kappa GA} + \frac{dM(x)}{dx} \tag{4.6}$$

$$\frac{d^2w(x)}{dx^2} + \frac{dM(x)}{dx} - \frac{d}{dx}\left(\frac{V(x)}{\kappa GA}\right) = 0 \tag{4.7}$$

$$M(x) = -\iint q(x)dxdx + C_1x + C_2 \tag{4.8}$$

$$EIxw(x) = -\iint \left[\frac{EI}{\kappa GA}q(x) + M(x)\right]dxdx + C_3x + C_4 \tag{4.9}$$

$$c(w - w_k) - \Phi\frac{d}{dx}\left(\frac{dw}{dx} - \psi\right) - q(x) = 0 \tag{4.10}$$

$$w = \left(1 + \frac{k}{c}\right)w_k - \left(\frac{G}{c}\right)\frac{d^2w_k}{dx^2} \tag{4.11}$$

$$-EI\frac{d^2\psi}{dx^2} + \Phi\left(\psi - \frac{dw}{dx}\right) \tag{4.12}$$

$$-\frac{EIG}{c}\frac{d^6w_k}{dx^6} + \left[EI\left(1 + \frac{k}{c}\right) + G\frac{EI}{\Phi}\right]\frac{d^4w_k}{dx^4} - \left[G + k\frac{EI}{\Phi}\right]\frac{d^2w_k}{dx^2} + kw_k + \left(\frac{EI}{\Phi}\right)\frac{d^2q}{dx2} - q = 0 \tag{4.13}$$

$$-\frac{EIG}{c}\frac{d^6\psi}{dx^6} + \left[EI\left(1 + \frac{k}{c}\right) + G\frac{EI}{\Phi}\right]\frac{d^4\psi}{dx^4} - \left[G + k\frac{EI}{\Phi}\right]\frac{d^2\psi}{dx^2} + k\psi - \left(1 + \frac{k}{c}\right)\frac{dq}{dx} + \frac{G}{c}\frac{d^3q}{dx^3} = 0 \tag{4.14}$$

$$w_k(x) = \sum_{i=1}^{6} C_i f_i \tag{4.15a}$$

$$\psi(x) = \sum_{i=1}^{6} C_i^{II} f_i \tag{4.15b}$$

$$w(x) = \sum_{i=1}^{6} C_i^{III} f_i \tag{4.15c}$$

$$M(x) = -EI\frac{d\psi}{dx} \tag{4.16a}$$

$$V(x) = -EI\frac{d^2\psi}{dx^2}\partial t \tag{4.16b}$$

$$V_G(x) = G\frac{dw_k(x)}{dx} \tag{4.16c}$$

$$G_s = \frac{E_s}{2 * (1 - \nu_s)} \tag{4.17a}$$

$$G = \frac{4 * H_s * G_s * B}{9} \tag{4.17b}$$

$$k_s = \frac{4 * E_s * B}{H_s} \tag{4.18}$$

$$c = \frac{4 * E_s * B}{3 * H_s} \tag{4.19}$$

$$u = \{w_1, \psi_1, w_{k1}, w_2, \psi_2, w_{k2}\} \text{ from (Morfidis 2007)} \tag{4.20}$$

$$s = \{V_1, M_1, V_{G1}, V_2, M_2, V_{G2}\} \text{ from (Morfidis 2007)} \tag{4.21}$$

$$c = R^{-1}u \text{ and } s = QR^{-1}u \text{ so } s = Ku \text{ and } K = QR^{-1} \tag{4.22}$$

$$y = F * y_0 + y_p \tag{4.23}$$

$$y = y(x) = \{w(x)\ \ \psi(x)\ \ w_k(x)\ \ V(x)\ \ M(x)\ \ V_G(x)\} \tag{4.24}$$

$$F = F(x) = [f_{ij}(x)], \quad i, j = 1 - 6 \tag{4.25}$$

$$y_0 = \{w_0\ \ \psi_0\ \ w_{k0}\ \ V_0\ \ M_0\ \ V_{G0}\} \tag{4.26}$$

$$y_p = y_p(x) = \{w_p(x)\ \ \psi_p(x)\ \ w_{kp}(x)\ \ V_p(x)\ \ M_p(x)\ \ V_{Gp}(x)\} \tag{4.27}$$

$$y_p^q(x) = \begin{cases} \{0\}, & x < aL \\ \displaystyle\int_{aL}^{x} q(u) f_{i4}(x-u)du (i = 1-6), & aL \le x \le bL, \\ & q(u) = q_a + \left[\frac{q_b - q_a}{(b-a)L}\right][u - (aL)] \\ \displaystyle\int_{aL}^{bL} q(u) f_{i4}(x-u)du (i = 1-6), & x > bL \end{cases} \tag{4.28}$$

$$\begin{bmatrix} u_L \\ s_L \end{bmatrix} = \begin{bmatrix} F_{uu} & F_{us} \\ F_{su} & F_{ss} \end{bmatrix} \begin{bmatrix} u_0 \\ s_0 \end{bmatrix} + \begin{bmatrix} u_{pl} \\ s_{pl} \end{bmatrix} \tag{4.29}$$

$$S_{c(p)} = \int \varepsilon_z dz \quad \text{(Das 2016)} \tag{4.30}$$

$$S_{c(p)} = \frac{C_c H_c}{1+e_0} \log \frac{\sigma_0^{'} + \Delta\sigma_{av}^{'}}{\sigma_0^{'}} \quad \text{(Das 2016)} \tag{4.31}$$

$$S_{c(p)} = \frac{C_s H_c}{1+e_0} \log \frac{\sigma_0^{'} + \Delta\sigma_{av}^{'}}{\sigma_0^{'}} \quad \text{(Das 2016)} \tag{4.32}$$

$$S_{c(p)} = \frac{C_s H_c}{1+e_0} \log \frac{\sigma_c^{'}}{\sigma_0^{'}} + \frac{C_c H_c}{1+e_0} \log \frac{\sigma_0^{'} + \Delta\sigma_{av}^{'}}{\sigma_0^{'}} \quad \text{(Das 2016)} \tag{4.33}$$

$$U = \frac{S_{c(t)}}{S_{c(max)}} = 1 - \sum_{m=0}^{m=\infty} \left(\frac{2}{M^2}\right) e^{-M^2 T_v} \quad \text{(Das 2016)} \tag{4.34}$$

$$\begin{aligned} T_v &= \frac{C_v t}{H_d^2} \quad \text{(Das 2016)} \\ C_v &= \frac{k}{m_v \gamma_w} = \frac{k}{\frac{\Delta e}{\Delta\sigma^{'}(1+e_{av})}\gamma_w} \quad \text{(Das 2016)} \end{aligned} \tag{4.35}$$

$$\nu\beta \frac{\partial u}{\partial t} - \frac{\partial \epsilon_z}{\partial t} + \frac{\partial v_z}{\partial z} = 0 \tag{4.36}$$

$$c_v \frac{\partial^2 u}{\partial z^2} = \frac{1}{\eta} \frac{\partial u}{\partial t} - \frac{d\sigma}{dt} \tag{4.37}$$

$$\sigma(t) = A cos(\omega t) + B sin(\omega t) \tag{4.38}$$

$$\frac{d\sigma}{dt} = -A\omega sin(\omega t) + B\omega cos(\omega t) \tag{4.39}$$

$$\sigma(t) = A_0 + \sum_{k=1}^{inf} [A_k cos(\omega_k t) + B_k sin(\omega_k t)] \tag{4.40}$$

$$\frac{\partial u}{\partial z}(o,t) = 0 \qquad \text{Impervious base} \tag{4.41a}$$

$$u(H,t) = 0 \qquad \text{Fully permeable upper surface} \tag{4.41b}$$

$$u(z,0) = 0 \qquad \text{Initial condition} \tag{4.41c}$$

$$u(z,t) = \int_0^t \frac{\partial \sigma(\tau)}{\partial \tau} \bar{u}(z, t-\tau) d\tau \tag{4.42}$$

$$u_\omega(z,t) = \sum_{j=1}^{\inf} \left[Y_j cos\left(\frac{(2j-1)\pi z}{2H}\right) exp\left(-\eta \left(\frac{(2j-1)\pi}{2H}\right)^2 T_v\right)\right] \tag{4.43}$$

$$s_\omega(t) = m_v H \left[A cos(\omega t) + B sin(\omega t) - \sum_{j=1}^{\inf} \frac{2}{\eta \chi^2} Y_j \right] \tag{4.44}$$

$$u(z,t) = \sum_{k=1}^{M} u_k(z,t) \tag{4.45}$$

$$s(t) = m_v H \left[\sigma_0 + \sum_{k=1}^{M} s_k(t) \right] \tag{4.46}$$

$$S_{totalB} = \sum_{j=1}^{m} (S_{consolidation,j} + S_{creep,j}) \tag{4.47}$$

$$S_{creep,j} = \sum_{k=1}^{n} (S_{creep,fj,k} + S_{creep,dj,k}) \tag{4.48}$$

$$S_{creep,fj} = \sum_{k=1}^{n} (\varepsilon_{creep,fj,k}) * h_k \tag{4.49}$$

$$S_{creep,dj} = \sum_{k=1}^{n} (\varepsilon_{creep,dj,k}) * h_k \tag{4.50}$$

$$\varepsilon_{creep,fj,k} = \frac{C_{\alpha e}}{(1+e_0)} log \left(\frac{t_0 + t_{ej,k}}{t_0 + \Delta t_{ej,k}} \right) \tag{4.51}$$

$$\Delta t_{ej,k} = t_0 * 10^{\left((\varepsilon_{ej,k} - \varepsilon_{zp(j-1),k}) \frac{(1+e_0)}{C_{\alpha e}} \right)} \left(\frac{\sigma'_{zj,k}}{\sigma'_{zp(j-1),k}} \right)^{-\frac{C_c}{C_{\alpha e}}} - t_0 \tag{4.52}$$

$$t_{ej,k} = t - \sum_{j=1}^{j-1} t_j - \frac{t_{cj}}{2} - t_0 + \Delta t_{ej,k} \tag{4.53}$$

$$\varepsilon_{creep,dj,k} = \frac{C_{\alpha e}}{(1+e_0)} log \left(\frac{t_0 + t_{ej,k}}{t_0} \right) \tag{4.54}$$

$$t_{ej,k} = t - \sum_{j=1}^{j-1} t_j - \frac{t_{cj}}{2} - t_0 \tag{4.55}$$

$$\varepsilon_{creep,dj,k} = \frac{C_{\alpha e}}{(1+e_0)} * log \left(\frac{t_0 + t_{ej,k}}{\Delta t_{ej,k} + t_{EOP}} \right) \tag{4.56}$$

$$\varepsilon_{creep,dj,k} = \frac{C_{\alpha e}}{(1+e_0)} * log \left(\frac{t_0 + t_{ej,k}}{t_{EOP}} \right) \tag{4.57}$$

$$\sigma'_{zp(j-1),k} = 10^{(\varepsilon_{z(j-2),t1,k} - \varepsilon_{zp(j-2),k}) * \frac{(1+e_0)}{(C_c - C_e)}} * (\sigma'_{z(j-1),k})^{-\frac{C_e}{(C_c - C_e)}} * \sigma'_{zp(j-2),k} \frac{C_c}{(C_c - C_e)} \tag{4.58}$$

Appendix E

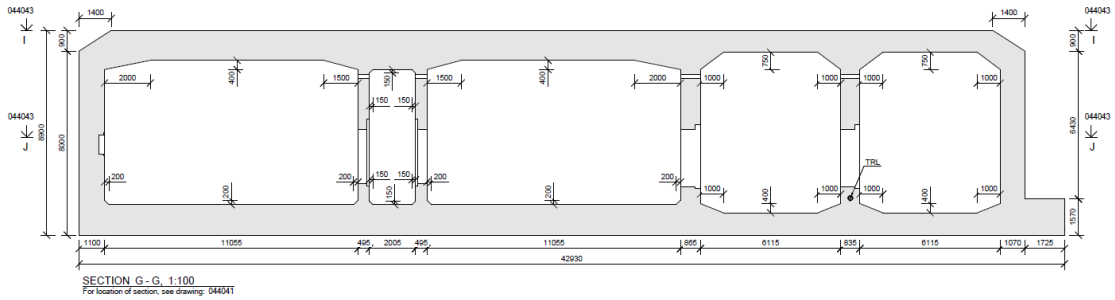
The Python script for the bouyancy calculation of the regular element is shown here.

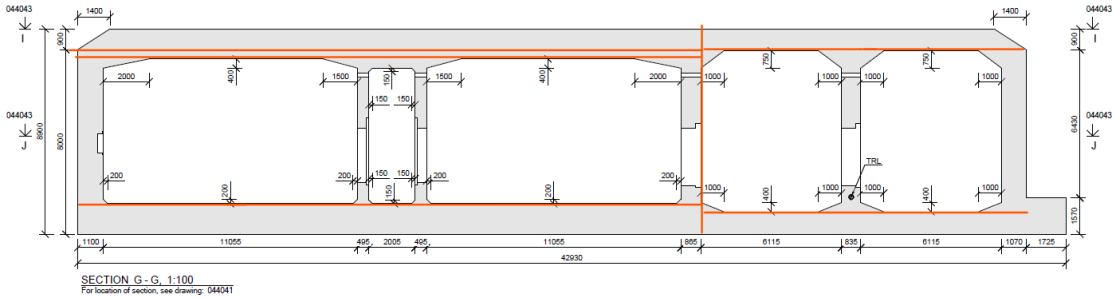
Regular_element_buoyancy_calculation_Fehmarn

April 29, 2025

[ ]:
```
from sympy import *
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import os
import math
import matplotlib.pyplot as plt
```

0.0.1 Geometrical properties - structural concrete





Cross section A (Regular element)

[2]:
```
#Roof left
h_r_l = 1.25            #height roof left side [m]
w_r_l = 27.070          #width roof left side [m]
a_r_l = h_r_l*w_r_l     #area roof left side [m2]
```

```
#Roof right
h_r_r = 0.900           #height roof right side [m]
w_r_r = 14.135          #width roof right side [m]
a_r_r = h_r_r * w_r_r   #area roof right side [m2]

#Outer walls (2x)
h_ow_l = 6.280              #height outer walls [m]
w_ow_l = 1.100              #width outer walls [m]
a_ow_l = h_ow_l*w_ow_l      #area outer walls [m2]

h_ow_r = 6.980              #height outer walls [m]
w_ow_r = 1.070              #width outer walls [m]
a_ow_r = h_ow_r*w_ow_r      #area outer walls [m2]

#Inner walls thin (2x)
h_iw_thin = 6.280                       #height inner walls [m]
w_iw_thin = 0.495                       #width inner walls [m]
a_iw_thin = 2*h_iw_thin*w_iw_thin       #area inner walls [m2]
a_iw_top = 0.400*2.500                  #area of the top concrete slap␣
 ↪underneath the roof [m2]

#Inner wall thick left
h_iw_thick_l = 6.230                            #height inner walls [m]
w_iw_thick_l = 0.865                            #width inner walls [m]
a_iw_thick_l = h_iw_thick_l*w_iw_thick_l        #area inner walls [m2]

#Inner wall thick right
h_iw_thick_r = 6.980                            #height inner walls [m]
w_iw_thick_r = 0.835                            #width inner walls [m]
a_iw_thick_r = h_iw_thick_r*w_iw_thick_r        #area inner walls [m2]

#Floor left
h_f_l = 1.320             #height floor [m]
w_f_l = 27.070            #width floor [m]
a_f_l = h_f_l*w_f_l       #area floor [m2]

#Floor right
h_f_r = 0.970             #height floor [m]
w_f_r = 14.135            #width floor [m]
a_f_r = h_f_r*w_f_r       #area floor [m2]

#Footing right side
h_fo = 1.570            #height footing left [m]
w_fo = 1.725            #width footing left [m]
a_fo = h_fo * w_fo      #area footing left [m2]

#Protection layer -----  Still need to figure out ------
```

```
h_pl = 0.150            #height portection layer [m]
w_pl = 42.280           #width portection layer [m]
a_pl = h_pl*w_pl        #area portection layer [m2]

#Intermediate floor   ------ does not exist in this model ------
#h_if = 0.500            #height intermediate floor [m]
#w_if = 5.100            #width intermediate floor [m]
#a_if = h_if*w_if        #area intermediate floor [m2]

#------------------ Totals -----------------
w_r = w_r_l + w_r_r                                 #Total width of roof [m]
a_r = a_r_l + a_r_r + a_iw_top                      #Total area roof [m2]
a_ow = a_ow_l + a_ow_r                              #Total area outer walls (2
 ↪walls) [m2]
a_iw = a_iw_thin + a_iw_thick_r + a_iw_thick_r      #Total area inner walls (4
 ↪walls) [m2]
a_f = a_f_l + a_f_r                                 #Total area floors [m2]
```

Envelopes

[3]:
```
#Main carriageway (2x)
h_mc = 6.280            #height main carriageway [m]
w_mc = 11.055           #width main carriageway [m]
a_mc = 2*h_mc * w_mc    #area main carriageway [m2]

#Service gallery
h_sg = 5.880            #height service gallery [m]
w_sg = 2.005            #width service gallery [m]
a_sg = h_sg * w_sg      #area service gallery [m2]

#Train carriageway (2x)
h_tc = 6.980            #height train gallery [m]
w_tc = 11.055           #width train gallery [m]
a_tc = 2 * h_tc * w_tc  #area train gallery [m2]
```

Add haunges (2x)

[4]:
```
#--------- no concrete ------------
#Top outside
h_to = 1.400                #Height top outside [m]
w_to = 0.900                #Width top outside [m]
a_to = h_to*w_to*0.5*2      #Area top outside [m2]

#--------- extra concrete ------------
#Top inside main carriageway
h_ti = 1.500                #Height top inside [m]
w_ti = 0.400                #Width top inside [m]
a_ti = h_ti*w_ti*0.5*4      #Area top inside [m2]
```

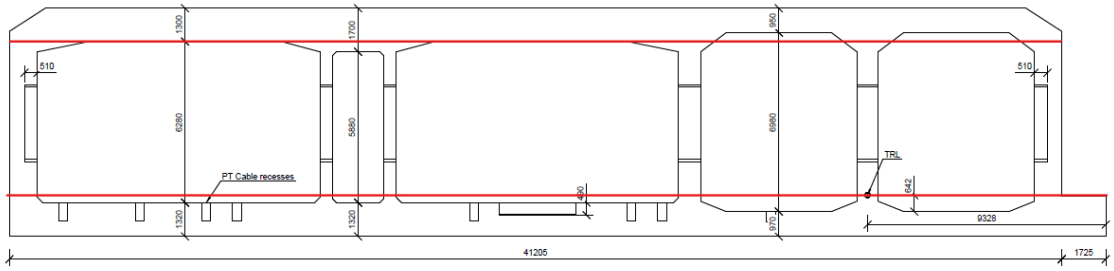
```
#Bottom inside main carriage way
h_bi = 0.200                 #Height bottom inside [m]
w_bi = 0.200                 #Width bottom inside [m]
a_bi = h_bi*w_bi*0.5*4       #Area bottom inside [m2]

#Top inside main carriageway
h_ti_mc = 1.500                        #Height top inside main carriageway [m]
w_ti_mc = 0.400                        #Width top inside main carriageway [m]
a_ti_mc = h_ti_mc*w_ti_mc*0.5*4        #Area top inside main carriageway [m2]

#Bottom inside main carriage way
h_bi_mc = 0.200                        #Height bottom inside main carriageway [m]
w_bi_mc = 0.200                        #Width bottom inside main carriageway [m]
a_bi_mc = h_bi_mc*w_bi_mc*0.5*4        #Area bottom inside main carriageway [m2]

#Gallery top
h_gt = 0.150                           #Height gallery top [m]
w_gt = 0.150                           #Width gallery top [m]
a_gt = h_gt*w_gt*0.5*2                 #Area gallery top [m2]

#Gallery bottom
h_gb = 0.150                           #Height gallery bottom [m]
w_gb = 0.150                           #Width gallery bottom [m]
a_gb = h_gb*w_gb*0.5*2                 #Area gallery bottom [m2]

#Top inside train carriage way
h_ti_tc = 0.750                        #Height top inside train carriageway [m]
w_ti_tc = 1.000                        #Width top inside train carriageway [m]
a_ti_tc = h_ti_tc*w_ti_tc*0.5*4        #Area top inside train carriageway [m2]

#Bottom inside train carriage way
h_bi_tc = 0.400                        #Height bottom inside train carriageway␣
 ↪[m]
w_bi_tc = 1.000                        #Width bottom inside train carriageway [m]
a_bi_tc = h_bi_tc*w_bi_tc*0.5*4        #Area bottom inside train carriageway [m2]
```

0.0.2 Main

[5]:
```
#Area of concrete of cross section [m2]
A_con_csA = a_r + a_ow + a_iw + a_f + a_pl + a_ti_mc + a_bi_mc + a_ti_tc +␣
 ↪a_bi_tc + a_gt + a_gb - a_to + a_bi  # + a_if

#Length of concrete of cross section [m]
l_con_csA = 217

#Volume of concrete of cross section [m3]
```

```
V_con_csA = l_con_csA * A_con_csA

#print(f'Total volume of the concrete cross section = {V_con_csA:.2f}␣
 ↪[m\N{SUPERSCRIPT THREE}]')
```

0.0.3 Geometrical properties - buoyancy



[6]:
```
#Roof
h_buo_r = 1.30                  #height roof [m]
w_buo_r = 41.205                #width roof [m]
a_buo_r = h_buo_r*w_buo_r       #area roof [m2]

#Walls
h_buo_w = 5.988                 #height walls [m]
w_buo_w = 41.205                #width walls [m]
a_buo_w = h_buo_w*w_buo_w       #area walls [m2]

#Floor
h_buo_f = 1.612                 #height floor [m]
w_buo_f = 42.930                #width floor [m]
a_buo_f = h_buo_f*w_buo_f       #area floor [m2]

#Protection layer
h_buo_pl = 0.150                #height portection layer [m]
w_buo_pl = 41.205               #width portection layer [m]
a_buo_pl = h_buo_pl*w_buo_pl    #area portection layer [m2]


#Buoyancy area of cross-section [m2]
A_buo_csA = a_buo_r + a_buo_w + a_buo_f  + a_buo_pl - a_to

#Length of cross section [m]
l_buo_csA = 217

#Buoyancy volume of cross section [m3]
V_buo_csA = l_buo_csA * A_buo_csA
```

```
#print(f'The buoyancy volume of the cross section is {V_buo_csA:.2f}␣
 ↪[m\N{SUPERSCRIPT THREE}]')
```

0.0.4 Summary geometrical properties

[7]:
```
L_TE = 217              #Total length of tunnel element [m]
V_con = V_con_csA       #Volume of structural concrete [m3]
V_buo = V_buo_csA       #Volume of buoyancy [m3]
v_st = 0                #Volume of steel membrane [m3]

#Quick chekc on freeboard (g_wat = 10 [kN/m3] and g_con = 24.5 [kN/m3])
Check = (V_buo*10 - V_con*24.5)/(l_buo_csA*10*w_r) #[m]

#print(f'{Check:.2f}')
```

0.1 Material

0.1.1 Material properties

[8]:
```
#Water
g_wat_min = 10.00         #[kN/m3]
g_wat_max = 10.35         #[kN/m3]

#Reinforced structural concrete
g_st_con_min = 24.00      #[kN/m3]
g_st_con_max = 25.25      #[kN/m3]

#Ballast concrete
g_b_con_min = 23.50       #[kN/m3]
g_b_con_max = 23.50       #[kN/m3]

#Steel sheet
g_st_min = 78.00          #[kN/m3]
g_st_max = 78.00          #[kN/m3]
```

0.1.2 Bulkheads

[9]:
```
#Water in front of bulkheads
A_bulk = A_buo_csA-A_con_csA      #Area bulkhead [m2]
d_wat = 0.550                     #Distance of water in front of bulkhead [m]

#Weight of primary bulkhead [kN]
F_bh = A_bulk*2*3.8

#print(f'Weight of the primary bulkhead is {F_bh:.2f} [kN]')
```

0.1.3 Ballast

[10]:
```
#Ballast concrete
t_bal_con = 0.850                                    #Average thickness of ballast concrete [m]
t_bal_con_serv = 0.850                               #Thickness of ballast concrete at gallery [m]
t_bal_con_train = 0.850                              #Thickness of ballast concrete at train gallery [m]
w_bal_con = 2 * w_mc                                 #Width of carriageway [m]
w_bal_con_serv = w_sg                                #Width of gallery [m]
w_bal_con_train = 2 * w_tc                           #Width of train gallery [m]
V_bal_con = ((w_bal_con_serv*t_bal_con_serv+w_bal_con*t_bal_con+w_bal_con_train*t_bal_con_train)-(a_bi_
    #Volume of ballast concrete [m3]

#print(f'The volume of the ballast concrete is {V_bal_con:.2f} [m\N{SUPERSCRIPT THREE}]')

#Ballast tanks
n_bal_tank = 4                          #Number of ballast tanks [-]
l_bal_tank = 50.000                     #length of ballast tanks [m]
w_bal_tank = w_mc                       #Width of ballast tanks [m]
h_bal_tank = 7                          #Maximum height of ballast tanks [m]
V_bal_tank = n_bal_tank*l_bal_tank*w_bal_tank*h_bal_tank                          #Volume of ballast tanks [m3]

#print(f'The volume of the ballast tanks is {V_bal_tank:.2f} [m\N{SUPERSCRIPT THREE}]')
```

0.1.4 Deck layout

[11]:
```
w_pin_con = 10                               #Weight pin construction [kN]
w_catch_con = 10                             #Weight catch construction [kN]
w_survey_tower = 1000                        #Weight survey tower [kN]
```

```
w_ac_shaft = 50                                                                 ␣
 ↪                    #Weight access shaft [kN]
w_B = 30                                                                        ␣
 ↪                    #Weight bollards [kN]
w_sus_lugs = 10                                                                 ␣
 ↪                    #Weight suspension lugs [kN]
w_bal_pip = 50                                                                  ␣
 ↪                    #Weight ballast piping [kN]

G_d_lay = w_pin_con + w_catch_con + w_survey_tower + w_ac_shaft + w_B +␣
 ↪w_sus_lugs + w_bal_pip      #Total weight of deck layout [kN]
```

0.1.5 Protection

- The rock protection will not be included in the buoyancy calculation
- The protection concrete will not be included in the buoyancy calculation and will be included as a reserve safety factor

0.2 Checks

0.2.1 Criterion 1: Minimum contact pressure in casting basin

[12]:
```
Per_bal_tank_1 = 51                                                             #Percentage␣
 ↪of water in ballast tanks [%]

Buo_1 = -V_buo_csA * g_wat_max                                                  #Buoyancy␣
 ↪[kN]
#print(f'Buoyancy = {Buo_1:.0f} [kN]')

Str_con_1 = V_con_csA * g_st_con_min                                            #Structural␣
 ↪concrete [kN]
#print(f'Structural concrete = {Str_con_1:.0f} [kN]')

Bul_1 = F_bh                                                                    #Bulkheads␣
 ↪[kN]
Bul_wat_1 = (A_bulk*d_wat)*g_wat_min                                            #Water in␣
 ↪front of bulkhead [kN]
Bal_wat_1 = (Per_bal_tank_1/100)*V_bal_tank*g_wat_min                           #Water in␣
 ↪ballast tanks [kN]

F_res_1 = Buo_1+Str_con_1+Bul_1+Bul_wat_1+Bal_wat_1
#print(f'The resulting force in criterion 1 is {F_res_1:.0f} [kN]')

#FoS
FoS_req = 1.015                                                                 #Required␣
 ↪factor of safety [-]
```

```
FoS_1 = (Str_con_1+Bul_1+Bul_wat_1+Bal_wat_1)/np.abs(Buo_1)          #Factor of␣
 ↪safety [-]

#print(f'The required factor of safety is {FoS_req:.3f}')
#print(f'The factor of safety is {FoS_1:.3f}')
```

0.2.2 Criterion 2: Freeboard after float-up

[13]:
```
Per_bal_tank_2 = 0                                                   #Percentage␣
 ↪of water in ballast tanks [%]

Buo_2 = -V_buo_csA * g_wat_min                                       #Buoyancy␣
 ↪[kN]
#print(f'Buoyancy = {Buo_2:.0f} [kN]')

Str_con_2 = V_con_csA * g_st_con_max                                 #Structural␣
 ↪concrete [kN]
#print(f'Structural concrete = {Str_con_2:.0f} [kN]')

Bul_2 = F_bh                                                         #Bulkheads␣
 ↪[kN]
Bul_wat_2 = (A_bulk*d_wat)*g_wat_min                                 #Water in␣
 ↪front of bulkhead [kN]
Bal_wat_2 = (Per_bal_tank_2/100)*V_bal_tank*g_wat_min                #Water in␣
 ↪ballast tanks [kN]
G_d_lay = G_d_lay                                                    #Total␣
 ↪weight of deck layout [kN]

F_res_2 = Buo_2+Str_con_2+Bul_2+Bul_wat_2+Bal_wat_2+G_d_lay
#print(f'The resulting force in criterion 2 is {F_res_2:.0f} [kN]')

h_hau = h_gt                                                         #Height of␣
 ↪haunch [m]
b_hau = w_gt                                                         #Width of␣
 ↪haunch [m]
b_elem = w_r                                                         #Width of␣
 ↪element [m]
b_t_r = b_elem - 2 * b_hau                                           #Width of␣
 ↪top roof [m]

#Freeboard
Frb_req = 0.150                                                      #Required␣
 ↪minimum freeboard [m]
Frb_2 = -F_res_2/(l_con_csA*g_wat_min*(b_t_r+b_hau/h_hau))           #Freeboard␣
 ↪[m]
```

```
#print(f'The required minimum freeboard is {Frb_req:.3f} [m]')
#print(f'The freeboard is {Frb_2:.3f} [m]')
```

0.2.3 Criterion 3: Final situation

[15]:
```
Per_bal_conc = 100                                                    #Percentage␣
 ↪of water in ballast tanks [%]

Buo_3 = -V_buo_csA * g_wat_max                                        #Buoyancy␣
 ↪[kN]
#print(f'Buoyancy = {Buo_3:.0f} [kN]')

Str_con_3 = V_con_csA * g_st_con_min                                  #Structural␣
 ↪concrete [kN]
#print(f'Structural concrete = {Str_con_3:.0f} [kN]')

Bal_con = (Per_bal_conc/100)*V_bal_con*g_b_con_min                    #Ballast␣
 ↪concrete [kN]
#print(f'Ballast concrete = {Bal_con:.0f} [kN]')

St_mem = v_st*g_st_min                                                #Steel␣
 ↪membrane [kN]
#print(f'Ballast concrete = {Bal_con:.0f} [kN]')

F_res_3_reg_first_guess = Buo_3+Str_con_3+Bal_con+St_mem
#print(f'The resulting force in criterion 3 is {F_res_3:.0f} [kN]')

#FoS
FoS_req = 1.060                                                       #Required␣
 ↪factor of safety [-]
FoS_1 = (Str_con_3+Bal_con+St_mem)/np.abs(Buo_3)                      #Factor of␣
 ↪safety [-]

Bal_con_req_reg = FoS_req * np.abs(Buo_3) - Str_con_3 - St_mem
F_res_3_reg = Buo_3 + Str_con_3 + Bal_con_req_reg + St_mem

#print(f'The required factor of safety is {FoS_req:.3f}')
#print(f'The factor of safety is {FoS_1:.3f}')
print(F_res_3_reg)
print(f'The required force of the ballast concrete needs to be:␣
 ↪{Bal_con_req_reg:.0f} [kN]')
```

```
50456.51774415001
The required force of the ballast concrete needs to be: 172487 [kN]
```

[ ]:

[ ]:

Appendix F

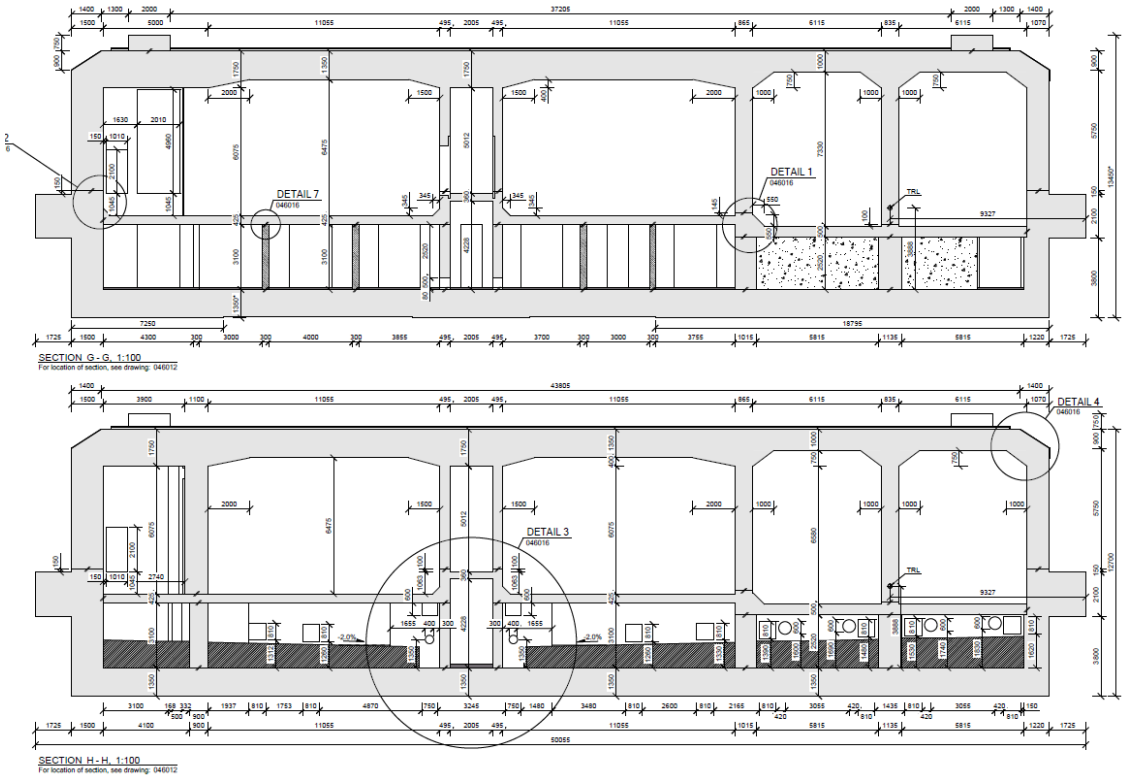
The Python script for the buoyancy calculation of the special element is shown here.

Special_element_buoyancy_calculation_Fehmarn

April 29, 2025

[1]:
```
from sympy import *
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import os
import math
import matplotlib.pyplot as plt
```

0.0.1 Geometrical properties - structural concrete



Cross section A (Regular element)

Floor

[2]:
```
w_f_d = 46.605                  #Width of complete floor at bottom element [m]
h_f_d = 1.350                   #Height of complete floor at bottom element [m]
a_f_d = w_f_d * h_f_d           #Element of complete floor at bottom element [m2]

w_f_l = 16.550                  #Width of floor underneath left main carriageway␣
 ↪[m]
h_f_l = 0.425                   #Height of floor underneath left main␣
 ↪carriageway [m]
a_f_l = w_f_l * h_f_l           #Area of floor underneath left main carriageway␣
 ↪[m2]

w_f_m = 12.415                  #Width of floor underneath right main␣
 ↪carriageway [m]
```

```
h_f_m = 0.425                   #Height of floor underneath right main␣
 ↪carriageway [m]
a_f_m = w_f_m * h_f_m           #Area of floor underneath right main carriageway␣
 ↪[m2]

w_f_r = 12.415                  #Width of floor underneath train carriageway [m]
h_f_r = 0.500                   #Height of floor underneath train carriageway [m]
a_f_r = w_f_r * h_f_r           #Area of floor underneath train carriageway [m2]

w_f_g = 2.005                   #Width of floor in gallery [m]
h_f_g = 0.360                   #Height of floor in gallery [m]
a_f_g = w_f_g * h_f_g           #Area of floor in gallery [m2]
```

Roof

[3]:
```
h_r_l = 1.750                   #Width of roof above left side leftt main␣
 ↪carriageway [m]
w_r_l = 6.500                   #Height of roof above left side lef main␣
 ↪carriageway [m]
a_r_l = h_r_l * w_r_l           #Area of roof above left side left main␣
 ↪carriageway [m2]

h_r_m = 1.350                   #Width of roof above main carriageway [m]
w_r_m = 25.970                  #Height of roof above main carriageway [m]
a_r_m = h_r_m * w_r_m           #Area of roof above main carriageway [m2]

h_r_r = 1.000                   #Width of roof above train carriageway [m]
w_r_r = 13.985                  #Height of roof above trian carriageway [m]
a_r_r = h_r_r * w_r_r           #Area of roof above train carriageway [m2]

h_r_g = 1.000                   #Width of intermediate layer service gallery [m]
w_r_g = 13.985                  #Height of intermediate layer service gallery [m]
a_r_g = h_r_g * w_r_g           #Area of intermediate layer service gallery [m2]
```

Walls

[4]:
```
w_w_l = 1.500                   #Width of wall on the left of the element [m]
h_w_l = 9.600                   #Height of wall on the left of the element [m]
a_w_l = w_w_l * h_w_l           #Area of wall on the left of the element [m2]

w_w_r = 1.220                   #Width of wall on the right of the element [m]
h_w_r = 10.350                  #Height of wall on the right of the element [m]
a_w_r = w_w_r * h_w_r           #Area of wall on the right of the element [m2]

w_w_g = 0.495                   #Width of walls of the service gallery [m]
h_w_g = (6.475+3.100)           #Height of walls of the service gallery [m]
a_w_g = 2 * w_w_g * h_w_g       #Area of walls of the service gallery [m2]
```

```
w_w_t1 = 1.015                  #Width of one of the walls of the train␣
 ↪carriageway [m]
w_w_t2 = 0.865                  #Width of one of the walls of the train␣
 ↪carriageway [m]
w_w_t3 = 1.135                  #Width of one of the walls of the train␣
 ↪carriageway [m]
w_w_t4 = 0.835                  #Width of one of the walls of the train␣
 ↪carriageway [m]
h_w_t1 = 2.520                  #Height of one of the walls of the train␣
 ↪carriageway [m]
h_w_t2 = 6.475                  #Height of one of the walls of the train␣
 ↪carriageway [m]
h_w_t3 = 2.520                  #Height of one of the walls of the train␣
 ↪carriageway [m]
h_w_t4 = 7.330                  #Height of one of the walls of the train␣
 ↪carriageway [m]
a_w_t1 = w_w_t1 * h_w_t1        #Area of one of the walls of the train␣
 ↪carriageway [m2]
a_w_t2 = w_w_t2 * h_w_t2        #Area of one of the walls of the train␣
 ↪carriageway [m2]
a_w_t3 = w_w_t3 * h_w_t3        #Area of one of the walls of the train␣
 ↪carriageway [m2]
a_w_t4 = w_w_t4 * h_w_t4        #Area of one of the walls of the train␣
 ↪carriageway [m2]
a_w_t = a_w_t1 + a_w_t2
+ a_w_t3 + a_w_t4               #Total area of the walls of the train␣
 ↪carriageway [m2]

w_w_cw1 = 0.900                  #Width of one of the walls of the service␣
 ↪gallery [m]
w_w_cw2 = 1.232                  #Width of one of the walls of the service␣
 ↪gallery [m]
h_w_cw1 = 3.100                  #Height of one of the walls of the service␣
 ↪gallery [m]
h_w_cw2 = 6.075                  #Height of one of the walls of the service␣
 ↪gallery [m]
a_w_cw1 = w_w_cw1 * h_w_cw1      #Area of one of the walls of the service␣
 ↪gallery [m2]
a_w_cw2 = w_w_cw2 * h_w_cw2      #Area of one of the walls of the service␣
 ↪gallery [m2]
a_w_cw = a_w_cw1 + a_w_cw2       #Total area of the walls of the service gallery␣
 ↪[m2]
```

Protection layer

[5]:
```
#-----  Still need to figure out ------
h_pl = 0.150             #Height portection layer [m]
w_pl = 43.805            #Width portection layer [m]
a_pl = h_pl*w_pl         #Area portection layer [m2]
```

Totals

[6]:
```
w_r = w_r_l + w_r_m + w_r_r                                  #Total width of roof␣
 ↪[m]
a_r = a_r_l + a_r_m + a_r_r + a_r_g                          #Total area roof [m2]
a_ow = a_w_l + a_w_r                                         #Total area outer␣
 ↪walls (2 walls) [m2]
a_iw = a_w_g + a_w_cw + a_w_t                                #Total area inner␣
 ↪walls (10 walls) [m2]
a_f = a_f_d + a_f_l + a_f_m + a_f_r + a_f_g                  #Total area floors␣
 ↪[m2]
```

Envelopes

[7]:
```
#Main carriageway (2x)
h_mc_u = 6.475                      #Height main carriageway upper part [m]
w_mc = 11.055                       #Width main carriageway upper part [m]
a_mc_u = 2*h_mc_u * w_mc            #Area main carriageway upper part [m2]

h_mc_l = 3.100                      #Height main carriageway lower part [m]
w_mc = 11.055                       #Width main carriageway lower part [m]
a_mc_l = 2*h_mc_l * w_mc            #Area main carriageway lower part [m2]

a_mc = a_mc_u + a_mc_l              #Total area main carriageway [m2]

#Service gallery
h_sgm_u = 5.012                     #Height middle service gallery upper part [m]
w_sgm_u = 2.005                     #Width middle service gallery upper part [m]
a_sgm_u = h_sgm_u * w_sgm_u         #Area middle service gallery upper part [m2]

h_sgm_l = 4.228                     #Height middle service gallery lower part [m]
w_sgm_l = 2.005                     #Width middle service gallery lower part [m]
a_sgm_l = h_sgm_l * w_sgm_l         #Area middle service gallery lower part [m2]

h_sgl_u = 6.075                     #Height middle left gallery upper part [m]
w_sgl_u = 3.900                     #Width middle left gallery upper part [m]
a_sgl_u = h_sgl_u * w_sgl_u         #Area left service gallery upper part [m2]

h_sgl_l = 3.100                     #Height left service gallery lower part [m]
w_sgl_l = 4.100                     #Width left service gallery lower part [m]
a_sgl_l = h_sgl_l * w_sgl_l         #Area left service gallery lower part [m2]
```

```
a_sg = a_sgm_l + a_sgm_u +  a_sgl_u + a_sgl_l                    #Total area␣
 ↪service gallery [m2]

#Train carriageway (2x)
h_tc_u = 7.330                      #Height train gallery upper part [m]
w_tc = 5.815                        #Width train gallery upper part [m]
a_tc_u = 2 * h_tc_u * w_tc          #Area train gallery upper part [m2]

h_tc_l = 2.500                      #Height train gallery lower part [m]
w_tc = 5.815                        #Width train gallery lower part [m]
a_tc_l = 2 * h_tc_l * w_tc          #Area train gallery lower part [m2]

a_tc = a_tc_l + a_tc_u              #Total area train gallery [m2]
```

Add haunges (2x)

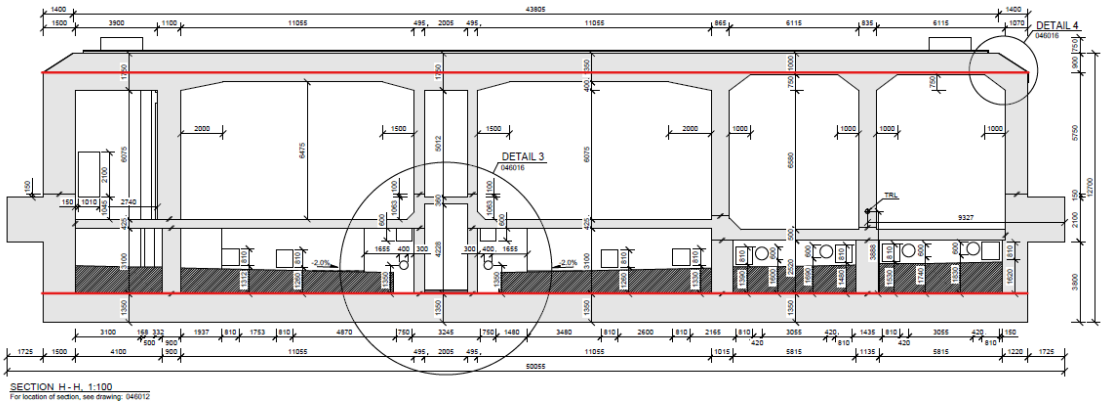
[8]:
```
#--------- no concrete ------------
#Top outside left
h_to_l = 0.900                        #Height top outside left [m]
w_to_l = 1.500                        #Width top outside left [m]
a_to_l= h_to_l*w_to_l*0.5             #Area top outside left [m2]

#Top outside right
h_to_r = 0.900                        #Height top outside right [m]
w_to_r = 1.400                        #Width top outside right [m]
a_to_r = h_to_r*w_to_r*0.5            #Area top outside right [m2]

a_to = a_to_l + a_to_r                #Total area of top outside haunges [m2]

#--------- extra concrete ------------

#Outside element (2x)
h_m = 2.100                           #Height concrete addition outside element␣
 ↪[m]
w_m = 1.725                           #Width concrete addition outside element␣
 ↪[m]
a_m = 2 *  h_m * w_m                  #Area concrete addition outside element␣
 ↪[m2]

#Top inside main carriageway (4 total)
h_ti_o = 0.400                        #Height top inside on the outer sides [m]
w_ti_o = 2.000                        #Width top inside on the outer sides [m]
a_ti_o = h_ti_o*w_ti_o*0.5*2          #Area top inside on the outer sides [m2]

h_ti_m = 0.400                        #Height top inside on the inner sides [m]
w_ti_m = 1.500                        #Width top inside on the inner sides [m]
a_ti_m = h_ti_m*w_ti_m*0.5*2          #Area top inside on the inner sides [m2]
```

```
a_ti = a_ti_m + a_ti_o                  #Total area of all haunges on top inside␣
 ↪the main carriageway [m2]

#Bottom inside main carriage way (2x)
h_bi = 0.345                            #Height bottom inside [m]
w_bi = 0.345                            #Width bottom inside [m]
a_bi_mc = h_bi*w_bi*0.5*2               #Area bottom inside [m2]

#Left gallery bottom (1x)
h_gb = 0.345                            #Height gallery bottom [m]
w_gb = 0.345                            #Width gallery bottom [m]
a_gb = h_gb*w_gb*0.5                    #Area gallery bottom [m2]

#Top inside train carriage way (4x)
h_ti_tc = 1.000                         #Height top inside train carriageway [m]
w_ti_tc = 0.750                         #Width top inside train carriageway [m]
a_ti_tc = h_ti_tc*w_ti_tc*0.5*4         #Area top inside train carriageway [m2]

#Bottom inside train carriage way (1x)
h_bi_tc = 0.550                         #Height bottom inside train carriageway [m]
w_bi_tc = 0.550                         #Width bottom inside train carriageway [m]
a_bi_tc = h_bi_tc*w_bi_tc*0.5           #Area bottom inside train carriageway [m2]
```

0.0.2 Main

[9]:
```
#Area of concrete of cross section [m2]
A_con_csA = a_r + a_ow + a_iw + a_f + a_pl + a_m + a_ti + a_bi_mc + a_ti_tc +␣
 ↪a_bi_tc + a_gb - a_to  # + a_gt + a_if + a_bo + a_ti_mc

#Length of concrete of cross section [m]
l_con_csA = 39

#Volume of concrete of cross section [m3]
V_con_csA = l_con_csA * A_con_csA

#print(f'Total volume of the concrete cross section = {V_con_csA:.2f}␣
 ↪[m\N{SUPERSCRIPT THREE}]')
```

0.0.3 Geometrical properties - buoyancy



[10]:
```
#Roof
h_buo_r = 1.000                 #Height roof [m]
w_buo_r = 46.605                #Width roof [m]
a_buo_r = h_buo_r*w_buo_r       #Area roof [m2]

#Walls
h_buo_w = 10.450                 #Height walls [m]
w_buo_w = 46.605                #Width walls [m]
a_buo_w = h_buo_w*w_buo_w       #area walls [m2]

#Floor
h_buo_f = 1.350                 #height floor [m]
w_buo_f = 46.605                #width floor [m]
a_buo_f = h_buo_f*w_buo_f       #area floor [m2]

#Protection layer
h_buo_pl = 0.150                #height portection layer [m]
w_buo_pl = 43.805               #width portection layer [m]
a_buo_pl = h_buo_pl*w_buo_pl    #area portection layer [m2]


#Buoyancy area of cross-section [m2]
A_buo_csA = a_buo_r + a_buo_w + a_buo_f  + a_buo_pl + a_m

#Length of cross section [m]
l_buo_csA = 39

#Buoyancy volume of cross section [m3]
V_buo_csA = l_buo_csA * A_buo_csA

#print(f'The buoyancy volume of the cross section is {V_buo_csA:.2f}␣
 ↪[m\N{SUPERSCRIPT THREE}]')
```

0.0.4 Summary geometrical properties

[11]:
```
L_TE = 39               #Total length of tunnel element [m]
V_con = V_con_csA       #Volume of structural concrete [m3]
V_buo = V_buo_csA       #Volume of buoyancy [m3]
v_st = 0                #Volume of steel membrane [m3]

#Quick check on freeboard (g_wat = 10 [kN/m3] and g_con = 24.5 [kN/m3])
Check = (V_buo*10 - V_con*24.5)/(l_buo_csA*10*w_r) #[m]

#print(f'{Check:.2f}')
```

0.1 Material

0.1.1 Material properties

[12]:
```
#Water
g_wat_min = 10.00        #[kN/m3]
g_wat_max = 10.35        #[kN/m3]

#Reinforced structural concrete
g_st_con_min = 24.00     #[kN/m3]
g_st_con_max = 25.25     #[kN/m3]

#Ballast concrete
g_b_con_min = 23.50      #[kN/m3]
g_b_con_max = 23.50      #[kN/m3]

#Steel sheet
g_st_min = 78.00         #[kN/m3]
g_st_max = 78.00         #[kN/m3]
```

0.1.2 Bulkheads

[13]:
```
#Water in front of bulkheads
A_bulk = A_buo_csA-A_con_csA     #Area bulkhead [m2]
d_wat = 0.550                    #Distance of water in front of bulkhead [m]

#Weight of primary bulkhead [kN]
F_bh = A_bulk*2*3.8

#print(f'Weight of the primary bulkhead is {F_bh:.2f} [kN]')
```

0.1.3 Ballast

[14]:
```
#Ballast concrete
t_bal_con = 1.300
 ↪                                          #Average thickness of ballast␣
 ↪concrete [m]
t_bal_con_serv = 1.380
 ↪                                          #Thickness of ballast concrete at␣
 ↪gallery [m]
t_bal_con_train = 1.380
 ↪                                          #Thickness of ballast concrete at␣
 ↪train gallery [m]
w_bal_con = 2 * w_mc
 ↪                                          #Width of carriageway [m]
w_bal_con_serv = w_sgl_l
 ↪                                          #Width of gallery [m]
w_bal_con_train = 2 * w_tc
 ↪                                          #Width of train gallery [m]
V_bal_con =␣
 ↪((w_bal_con_serv*t_bal_con_serv+w_bal_con*t_bal_con+w_bal_con_train*t_bal_con_train)-(2*0.
 ↪750))*l_con_csA    #Volume of ballast concrete [m3]

#print(f'The volume of the ballast concrete is {V_bal_con:.2f} [m\N{SUPERSCRIPT␣
 ↪THREE}]')

#Ballast tanks
n_bal_tank = 4
 ↪                #Number of ballast tanks [-]
l_bal_tank = 39.000
 ↪                #length of ballast tanks [m]
w_bal_tank = w_mc
 ↪                #Width of ballast tanks [m]
h_bal_tank = 3
 ↪                #Maximum height of ballast tanks [m]
V_bal_tank = n_bal_tank*l_bal_tank*w_bal_tank*h_bal_tank
 ↪                #Volume of ballast tanks [m3]

#print(f'The volume of the ballast tanks is {V_bal_tank:.2f} [m\N{SUPERSCRIPT␣
 ↪THREE}]')
```

0.1.4 Deck layout

[15]:
```
w_pin_con = 10
 ↪                    #Weight pin construction [kN]
w_catch_con = 10
 ↪                    #Weight catch construction [kN]
```

```
w_survey_tower = 1000                                                          ␣
 ↪                    #Weight survey tower [kN]
w_ac_shaft = 50                                                                ␣
 ↪                    #Weight access shaft [kN]
w_B = 30                                                                       ␣
 ↪                    #Weight bollards [kN]
w_sus_lugs = 10                                                                ␣
 ↪                    #Weight suspension lugs [kN]
w_bal_pip = 50                                                                 ␣
 ↪                    #Weight ballast piping [kN]

G_d_lay = w_pin_con + w_catch_con + w_survey_tower + w_ac_shaft + w_B +␣
 ↪w_sus_lugs + w_bal_pip      #Total weight of deck layout [kN]
```

0.1.5 Protection

- The rock protection will not be included in the buoyancy calculation
- The protection concrete will not be included in the buoyancy calculation and will be included as a reserve safety factor

0.2 Checks

0.2.1 Criterion 1: Minimum contact pressure in casting basin

[16]:
```
Per_bal_tank_1 = 51                                                #Percentage␣
 ↪of water in ballast tanks [%]

Buo_1 = -V_buo_csA * g_wat_max                                     #Buoyancy␣
 ↪[kN]
#print(f'Buoyancy = {Buo_1:.0f} [kN]')

Str_con_1 = V_con_csA * g_st_con_min                               #Structural␣
 ↪concrete [kN]
#print(f'Structural concrete = {Str_con_1:.0f} [kN]')

Bul_1 = F_bh                                                       #Bulkheads␣
 ↪[kN]
Bul_wat_1 = (A_bulk*d_wat)*g_wat_min                               #Water in␣
 ↪front of bulkhead [kN]
Bal_wat_1 = (Per_bal_tank_1/100)*V_bal_tank*g_wat_min              #Water in␣
 ↪ballast tanks [kN]

F_res_1 = Buo_1+Str_con_1+Bul_1+Bul_wat_1+Bal_wat_1
#print(f'The resulting force in criterion 1 is {F_res_1:.0f} [kN]')

#FoS
```

```
FoS_req = 1.015                                                    #Required␣
 ↪factor of safety [-]
FoS_1 = (Str_con_1+Bul_1+Bul_wat_1+Bal_wat_1)/np.abs(Buo_1)        #Factor of␣
 ↪safety [-]

#print(f'The required factor of safety is {FoS_req:.3f}')
#print(f'The factor of safety is {FoS_1:.3f}')
```

0.2.2 Criterion 2: Freeboard after float-up

[17]:
```
Per_bal_tank_2 = 0                                                 #Percentage␣
 ↪of water in ballast tanks [%]

Buo_2 = -V_buo_csA * g_wat_min                                     #Buoyancy␣
 ↪[kN]
#print(f'Buoyancy = {Buo_2:.0f} [kN]')

Str_con_2 = V_con_csA * g_st_con_max                               #Structural␣
 ↪concrete [kN]
#print(f'Structural concrete = {Str_con_2:.0f} [kN]')

Bul_2 = F_bh                                                       #Bulkheads␣
 ↪[kN]
Bul_wat_2 = (A_bulk*d_wat)*g_wat_min                               #Water in␣
 ↪front of bulkhead [kN]
Bal_wat_2 = (Per_bal_tank_2/100)*V_bal_tank*g_wat_min              #Water in␣
 ↪ballast tanks [kN]
G_d_lay = G_d_lay                                                  #Total␣
 ↪weight of deck layout [kN]

F_res_2 = Buo_2+Str_con_2+Bul_2+Bul_wat_2+Bal_wat_2+G_d_lay
#print(f'The resulting force in criterion 2 is {F_res_2:.0f} [kN]')

h_hau = 0.1                                                          #Height of␣
 ↪haunch [m]
b_hau = 0.1                                                          #Width of␣
 ↪haunch [m]
b_elem = w_r                                                       #Width of␣
 ↪element [m]
b_t_r = b_elem - 2 * b_hau                                         #Width of␣
 ↪top roof [m]

#Freeboard
Frb_req = 0.150                                                    #Required␣
 ↪minimum freeboard [m]
```

```
Frb_2 = -F_res_2/(l_con_csA*g_wat_min*(b_t_r+b_hau/h_hau))          #Freeboard
 ↪[m]

#print(f'The required minimum freeboard is {Frb_req:.3f} [m]')
#print(f'The freeboard is {Frb_2:.3f} [m]')
```

0.2.3 Criterion 3: Final situation

[19]:
```
Per_bal_conc = 100                                                  #Percentage
 ↪of water in ballast tanks [%]

Buo_3 = -V_buo_csA * g_wat_max                                      #Buoyancy
 ↪[kN]
#print(f'Buoyancy = {Buo_3:.0f} [kN]')

Str_con_3 = V_con_csA * g_st_con_min                                #Structural
 ↪concrete [kN]
#print(f'Structural concrete = {Str_con_3:.0f} [kN]')

Bal_con = (Per_bal_conc/100)*V_bal_con*g_b_con_min                  #Ballast
 ↪concrete [kN]
#print(f'Ballast concrete = {Bal_con:.0f} [kN]')

St_mem = v_st*g_st_min                                              #Steel
 ↪membrane [kN]
#print(f'Ballast concrete = {Bal_con:.0f} [kN]')

F_res_3_spec_first_guess = Buo_3+Str_con_3+Bal_con+St_mem
#print(f'The resulting force in criterion 3 is {F_res_3:.0f} [kN]')

#FoS
FoS_req = 1.060                                                     #Required
 ↪factor of safety [-]
FoS_1 = (Str_con_3+Bal_con+St_mem)/np.abs(Buo_3)                    #Factor of
 ↪safety [-]

Bal_con_req_spec = FoS_req * np.abs(Buo_3) - Str_con_3 - St_mem
F_res_3_spec = Buo_3 + Str_con_3 + Bal_con_req_spec + St_mem

#print(f'The required factor of safety is {FoS_req:.3f}')
#print(f'The factor of safety is {FoS_1:.3f}')

print(Bal_con_req_spec)
print(F_res_3_spec)
```

```
48456.87277274989
14782.302785250009
```

[ ]:

G

Appendix G

The Python script for the matrices used in the TBKF model is shown here.

Matrix

May 22, 2025

[1]:
```
import import_ipynb
import Matrix
import numpy as np
from scipy.integrate import quad_vec, quad
from numba import jit
import pandas as pd
```

```
Import Matrix was succesfull
```

[2]:
```
# Define a function to check if a matrix is symmetric
def check_symmetric(matrix, tol=1e-3):
    return np.allclose(matrix, matrix.T, atol=tol)
```

[3]:
```
def cuberoot(x):
    if x < 0:
        x = abs(x)
        cube_root = x**(1/3)*(-1)
    else:
        cube_root = x**(1/3)
    return cube_root
```

[ ]:
```
def PC(k_s, kappa, E_b, v_b, E_s, v_s, B, H, H_s):
    G_b = E_b/(2*(1-v_b))                                   # [kPa]
    G_s = E_s/(2*(1-v_s))                                   # [kPa]
    A = B*H                                                 # [m^2]
    I = (1/12)*B*H**3                                       # [m^4]
    G = ((4*H_s*G_s*B)/9)                                   # [kPa*m]       (Morfidis,␣
 ↪2003)
    k = (4*E_s*B)/(H_s)                                     # [kN/m^3*m]    (Morfidis,␣
 ↪2003)
    c = (4*E_s*B)/(3*H_s)                                   # [kN/m^3*m]    (Morfidis,␣
 ↪2003)
    EI = E_b*I                                              # [kNm^2]
    F_a = kappa*A                                           # [m^2]
    Phi = G_b * F_a                                         # [kN]

    # print(f"Moment of Inertia (I): {I:.2f} m^4")
    # print(f"Elastic Modulus of Beam (E_b): {E_b} kPa")
```

```
    # print(f'A = {A} m^2')
    # print(f'G_b = {G_b} [kPa]')
    # print(f"Spring Stiffness (k): {k:.2f} kN/m^3")
    # print(f"Damping Coefficient (c): {c:.2f} kN/m^3")
    # print(f"Shear modulus (G): {G:.2f} kPa")
    # print(f'EI = {EI}')

    return EI, k, c, G, Phi, A
```

[4]:
```
def K_M(EI, k, c, G, Phi, L_ele):
    #print(f'Check if this is the element length: {L_ele} m')
    # Calculations for J_1, J_2, J_3
    J_1 = -((k + c) / G + c / Phi)
    J_2 = (c / Phi) * (Phi / EI + k / G)
    J_3 = -((k * c) / (EI * G))

    # Calculations for alpha, beta, delta
    alpha = (1/3) * (-((J_1**2)/3) + J_2)
    beta = (1/2) * (((2 * J_1**3) / 27) - ((J_1 * J_2) / 3) + J_3)
    delta = alpha ** 3 + beta ** 2

    # Print delta with a descriptive message
    #print(f"Delta value: {delta:.13f}")
    if delta <= 0:
        raise Exception("Sorry, no numbers below zero and check your parameters!
")

    # Calculations for n, m, Q, R, R_1
    n = (np.sqrt(3)/2)*(Matrix.cuberoot(-beta+np.sqrt(delta))-(Matrix.
cuberoot(-beta-np.sqrt(delta))))
    m = -(1/2)*((Matrix.cuberoot(-beta+np.sqrt(delta)))+(Matrix.
cuberoot(-beta-np.sqrt(delta)))+(2*J_1/3))
    Q = np.sqrt((np.sqrt(m**2+n**2)-m)/2)
    R = np.sqrt((np.sqrt(m**2+n**2)+m)/2)
    R_1 = np.sqrt((Matrix.cuberoot(-beta+np.sqrt(delta)))+(Matrix.
cuberoot(-beta-np.sqrt(delta)))-(J_1/3))

    # Calculations for A_1, A_2, A_3
    A_1 = (1+k/c)-(G/c)*R_1**2
    A_2 = 2*R*Q*(G/c)
    A_3 = (1+k/c)-(G/c)*(R**2-Q**2)

    # Calculations for beta_1, beta_2, alpha_1, alpha_2
    beta_1 = -EI*(R**2-Q**2)+Phi
    beta_2 = 2*(EI)*R*Q
    alpha_1 = A_3*R+A_2*Q
    alpha_2 = A_2*R - A_3 *Q
```

```
# Calculations for A_4, A_5, A_6
A_4 = ((Phi*R_1)/(-EI*(R_1**2)+Phi))*A_1
A_5 = ((alpha_1*beta_1 + alpha_2*beta_2)/(beta_1**2+beta_2**2))*Phi
A_6 = ((alpha_1*beta_2 - alpha_2*beta_1)/(beta_1**2+beta_2**2))*Phi

# Calculations for b_1, b_2, b_3, b_4, d_1, d_2, d_3
b_1 = (R**2-Q**2)*A_5 -(2*R*Q)*A_6
b_2 = (R**2-Q**2)*A_6 + (2*R*Q)*A_5
b_3 = A_6*R+A_5*Q
b_4 = A_5*R - A_6*Q
d_1 = EI*(A_4*R_1**2)
d_2 = EI * b_1
d_3 = EI * b_2

# Calculations for D_2, D_1
D_2 = Q*(A_4*d_2-A_5*d_1)+R*(A_6*d_1-A_4*d_3)+R_1*(A_5*d_3-A_6*d_2)
D_1 = b_3 *(A_3-A_1)+A_2*(b_4-A_4*R_1)

# Calculations for f_1l, f_2l, f_3l , f_4l f_5l and f_6l
f_1l = np.exp(R_1 *L_ele)
f_2l = np.exp(-R_1*L_ele)
f_3l = np.exp(R*L_ele)*np.cos(Q*L_ele)
f_4l = np.exp(R*L_ele)*np.sin(Q*L_ele)
f_5l = np.exp(-R*L_ele)*np.cos(Q*L_ele)
f_6l = np.exp(-R*L_ele)*np.sin(Q*L_ele)

# Initialize Q_M matrix
Q_M = np.zeros((6,6))

# Populate Q_M matrix
Q_M[0,0] = d_1
Q_M[0,1] = -d_1
Q_M[0,2] = d_2
Q_M[0,3] = d_3
Q_M[0,4] = -d_2
Q_M[0,5] = d_3

Q_M[1,0] = -(EI) * (A_4*R_1)
Q_M[1,1] = -(EI) * (A_4*R_1)
Q_M[1,2] = -(EI)* b_4
Q_M[1,3] = -(EI) * b_3
Q_M[1,4] = -(EI) * b_4
Q_M[1,5] = (EI) * b_3

Q_M[2,0] = -G*R_1
Q_M[2,1] = G*R_1
```

```
    Q_M[2,2] = -G*R
    Q_M[2,3] = -G*Q
    Q_M[2,4] = G*R
    Q_M[2,5] = -G*Q

    Q_M[3,0] = -d_1*f_1l
    Q_M[3,1] = d_1*f_2l
    Q_M[3,2] = -d_2*f_3l + d_3*f_4l
    Q_M[3,3] = -d_3*f_3l - d_2*f_4l
    Q_M[3,4] = d_2*f_5l + d_3*f_6l
    Q_M[3,5] = -d_3*f_5l + d_2*f_6l

    Q_M[4,0] = (EI)*A_4*R_1*f_1l
    Q_M[4,1] = (EI)*A_4*R_1*f_2l
    Q_M[4,2] = (EI)*(b_4*f_3l-b_3*f_4l)
    Q_M[4,3] = (EI)*(b_3*f_3l+b_4*f_4l)
    Q_M[4,4] = (EI)*(b_4*f_5l+b_3*f_6l)
    Q_M[4,5] = (EI)*(b_4*f_6l-b_3*f_5l)

    Q_M[5,0] = G*R_1*f_1l
    Q_M[5,1] = -G*R_1*f_2l
    Q_M[5,2] = G*(R*f_3l-Q*f_4l)
    Q_M[5,3] = G*(R*f_4l+Q*f_3l)
    Q_M[5,4] = -G*(R*f_5l+Q*f_6l)
    Q_M[5,5] = G*(-R*f_6l+Q*f_5l)

    # Initialize R_M matrix
    R_M = np.zeros((6,6))

    # Populate R_M matrix
    R_M[0,0] = A_1
    R_M[0,1] = A_1
    R_M[0,2] = A_3
    R_M[0,3] = -A_2
    R_M[0,4] = A_3
    R_M[0,5] = A_2

    R_M[1,0] = A_4
    R_M[1,1] = -A_4
    R_M[1,2] = A_5
    R_M[1,3] = A_6
    R_M[1,4] = -A_5
    R_M[1,5] = A_6

    R_M[2,0] = 1
    R_M[2,1] = 1
    R_M[2,2] = 1
```

```
    R_M[2,3] = 0
    R_M[2,4] = 1
    R_M[2,5] = 0

    R_M[3,0] = A_1*f_1l
    R_M[3,1] = A_1*f_2l
    R_M[3,2] = A_3*f_3l+A_2*f_4l
    R_M[3,3] = A_3*f_4l-A_2*f_3l
    R_M[3,4] = A_3*f_5l-A_2*f_6l
    R_M[3,5] = A_3*f_6l+A_2*f_5l

    R_M[4,0] = A_4*f_1l
    R_M[4,1] = -A_4*f_2l
    R_M[4,2] = A_5*f_3l-A_6*f_4l
    R_M[4,3] = A_6*f_3l+A_5*f_4l
    R_M[4,4] = -A_5*f_5l-A_6*f_6l
    R_M[4,5] = A_6*f_5l-A_5*f_6l

    R_M[5,0] = f_1l
    R_M[5,1] = f_2l
    R_M[5,2] = f_3l
    R_M[5,3] = f_4l
    R_M[5,4] = f_5l
    R_M[5,5] = f_6l

    # Calculate K_M matrix
    K_M = Q_M.dot(np.linalg.inv(R_M))

    # # Print the K_M matrix
    # print("K_M matrix:")
    # df = pd.DataFrame(K_M, columns=[f'Column {i+1}' for i in range(K_M.
↪shape[1])])
    # print(df)

    # print("Q_M matrix:")
    # df_2 = pd.DataFrame(Q_M, columns=[f'Column {i+1}' for i in range(Q_M.
↪shape[1])])
    # print(df_2)

    # print("R_M matrix:")
    # df_3 = pd.DataFrame(R_M, columns=[f'Column {i+1}' for i in range(R_M.
↪shape[1])])
    # print(df_3)

    # Perform checks
    #print("Check 1 (K_M[3,4] + K_M[0,1]):", K_M[3, 4] + K_M[0, 1])
    #print("Check 2 (K_M[4,3] + K_M[1,0]):", K_M[4, 3] + K_M[1, 0])
```

```
    #print("Check 3 (K_M[5,4] + K_M[2,1]):", K_M[5, 4] + K_M[2, 1])
    #print("Check 4 (K_M[4,5] + K_M[1,2]):", K_M[4, 5] + K_M[1, 2])

    # Check if K_M is symmetric
    #is_symmetric = Matrix.check_symmetric(K_M)
    #print("Is K_M symmetric?:", is_symmetric)

    # print(f'A_1 = {A_1}')
    # print(f'A_2 = {A_2}')
    # print(f'A_3 = {A_3}')
    # print(f'A_4 = {A_4}')
    # print(f'A_5 = {A_5}')
    # print(f'A_6 = {A_6}')

    # print(f'J1 = {J_1}')
    # print(f'J2 = {J_2}')
    # print(f'J3 = {J_3}')

    # print(f'alpha = {alpha}')
    # print(f'beta = {beta}')

    # print(f'R = {R}')
    # print(f'Q={Q}')
    # print(f'R_1 = {R_1}')
    # print(f'alpha1 = {alpha_1}')
    # print(f'beta1 = {beta_1}')
    # print(f'alpha2 = {alpha_2}')
    # print(f'beta2 = {beta_2}')

    return K_M, R_M, Q_M
```

[5]:
```
def f_m(EI, k, c, G , Phi, x):
    # Calculations for J_1, J_2, J_3
    J_1 = -((k + c) / G + c / Phi)
    J_2 = (c / Phi) * (Phi / EI + k / G)
    J_3 = -((k * c) / (EI * G))

    # Calculations for alpha, beta, delta
    alpha = (1/3) * (-((J_1**2)/3) + J_2)
    beta = (1/2) * (((2 * J_1**3) / 27) - ((J_1 * J_2) / 3) + J_3)
    delta = alpha ** 3 + beta ** 2

    # Print delta with a descriptive message
    #print(f"Delta value: {delta:.13f}")
    if delta <= 0:
        raise Exception("Sorry, no numbers below zero and check your parameters!
 ↪")
```

```
    # Calculations for n, m, Q, R, R_1
    n = (np.sqrt(3)/2)*(Matrix.cuberoot(-beta+np.sqrt(delta))-(Matrix.
↪cuberoot(-beta-np.sqrt(delta))))
    m = -(1/2)*((Matrix.cuberoot(-beta+np.sqrt(delta)))+(Matrix.
↪cuberoot(-beta-np.sqrt(delta)))+(2*J_1/3))
    Q = np.sqrt((np.sqrt(m**2+n**2)-m)/2)
    R = np.sqrt((np.sqrt(m**2+n**2)+m)/2)
    R_1 = np.sqrt((Matrix.cuberoot(-beta+np.sqrt(delta)))+(Matrix.
↪cuberoot(-beta-np.sqrt(delta)))-(J_1/3))

    # Calculations for A_1, A_2, A_3
    A_1 = (1+k/c)-(G/c)*R_1**2
    A_2 = 2*R*Q*(G/c)
    A_3 = (1+k/c)-(G/c)*(R**2-Q**2)

    # Calculations for beta_1, beta_2, alpha_1, alpha_2
    beta_1 = -EI*(R**2-Q**2)+Phi
    beta_2 = 2*(EI)*R*Q
    alpha_1 = A_3*R+A_2*Q
    alpha_2 = A_2*R - A_3 *Q

    # Calculations for A_4, A_5, A_6
    A_4 = ((Phi*R_1)/(-EI*(R_1**2)+Phi))*A_1
    A_5 = ((alpha_1*beta_1 + alpha_2*beta_2)/(beta_1**2+beta_2**2))*Phi
    A_6 = ((alpha_1*beta_2 - alpha_2*beta_1)/(beta_1**2+beta_2**2))*Phi

    # Calculations for b_1, b_2, b_3, b_4, d_1, d_2, d_3
    b_1 = (R**2-Q**2)*A_5 -(2*R*Q)*A_6
    b_2 = (R**2-Q**2)*A_6 + (2*R*Q)*A_5
    b_3 = A_6*R+A_5*Q
    b_4 = A_5*R - A_6*Q
    d_1 = EI*(A_4*R_1**2)
    d_2 = EI * b_1
    d_3 = EI * b_2

    # Calculations for D_2, D_1
    D_2 = Q*(A_4*d_2-A_5*d_1)+R*(A_6*d_1-A_4*d_3)+R_1*(A_5*d_3-A_6*d_2)
    D_1 = b_3 *(A_3-A_1)+A_2*(b_4-A_4*R_1)

    f_m = np.zeros((6,6))
    n_x = np.sinh(R*x)
    m_x = np.cosh(R*x)
    n_ = np.sin(Q*x)
    m_ = np.cos(Q*x)
    n_1 = np.sinh(R_1*x)
    m_1 = np.cosh(R_1*x)
```

```
  f_m[0,0] = -((A_1 * b_3) / D_1) * m_1 + ((A_3 * b_3 - A_2 * (A_4 * R_1 -␣
↪b_4)) / D_1) * m_x * m_ + ((A_2 * b_3 + A_3 * (A_4 * R_1 - b_4)) / D_1) *␣
↪n_x * n_
  f_m[0,1] = ((A_1 * (d_2 * Q - d_3 * R)) / D_2) * n_1 - ((A_3 * (d_1 * Q -␣
↪d_3 * R_1) + A_2 * (d_1 * R - d_2 * R_1)) / D_2) * n_x * m_ - ((A_3 * (d_2 *␣
↪R_1 - d_1 * R) + A_2 * (d_1 * Q - d_3 * R_1)) / D_2) * m_x * n_
  f_m[0,2] = ((A_1 * (A_3 * b_3 + A_2 * b_4)) / D_1) * m_1 - ((A_1 * (A_3 *␣
↪b_3 + A_2 * b_4)) / D_1) * m_x * m_ - ((A_4 * R_1 * (A_2**2 + A_3**2) + A_1␣
↪* (A_2 * b_3 - A_3 * b_4)) / D_1) * n_x * n_
  f_m[0,3] = -((A_1 * (A_5 * Q - A_6 * R)) / D_2) * n_1 - ((A_3 * (A_6 * R_1␣
↪- A_4 * Q) + A_2 * (A_5 * R_1 - A_4 * R)) / D_2) * n_x * m_ + ((A_2 * (-A_6␣
↪* R_1 + A_4 * Q) + A_3 * (A_5 * R_1 - A_4 * R)) / D_2) * m_x * n_
  f_m[0,4] = ((A_1 * A_2) / (EI*D_1)) * m_1 - ((A_1 * A_2) / (EI*D_1)) * m_x␣
↪* m_ + ((A_3 * (A_1 - A_3) - A_2**2) / (EI*D_1)) * n_x * n_
  f_m[0,5] = -((A_5 * Q - A_6 * R) / D_2) * n_1 + ((A_4 * Q - A_6 * R_1) /␣
↪D_2) * n_x * m_ - ((A_4 * R - A_5 * R_1) / D_2) * m_x * n_

  f_m[1,0] = -((A_4*b_3)/D_1)*n_1+((A_5*b_3+A_6*(A_4*R_1-b_4))/
↪D_1)*n_x*m_-((A_6*b_3+A_5*(b_4-A_4*R_1))/D_1)*m_x*n_
  f_m[1,1] = -((A_2*A_4*R_1)/D_1)*m_1-(((A_1-A_3)*b_3-A_2*b_4)/
↪D_1)*m_x*m_-(((A_1-A_3)*b_4+A_2*b_3)/D_1)*n_x*n_
  f_m[1,2] = ((G*R_1*A_2)/(EI*D_1))*n_1 + ((G*((A_1-A_3)*Q-A_2*R))/
↪(EI*D_1))*n_x*m_ + ((G*((A_1-A_3)*R-A_2*Q))/(EI*D_1))*m_x*n_
  f_m[1,3] = -f_m[0,4]
  f_m[1,4] = ((A_2*A_4)/(EI*D_1))*n_1-((A_2*A_5-(A_1-A_3)*A_6)/
↪(EI*D_1))*n_x*m_+((A_2*A_6+A_5*(A_1-A_3))/(EI*D_1))*m_x*n_
  f_m[1,5] = -((A_2 / (EI*D_1)) * m_1 - ( A_2 / (EI*D_1)) * m_x * m_ + ((A_1␣
↪- A_3)/ (EI*D_1)) * n_x * n_)

  f_m[2,0] = - (b_3/ D_1) * m_1 + (b_3 / D_1) * m_x * m_ + ((A_4 * R_1 - b_4)␣
↪/ D_1) * n_x * n_
  f_m[2,1] = ((d_2 * Q - d_3 * R) / D_2) * n_1 - ((d_1 * Q - d_3 * R_1)/ D_2)␣
↪* n_x * m_ + ((d_1 * R - d_2 * R_1) / D_2) * m_x * n_
  f_m[2,2] = ((A_3 * b_3 + A_2 * b_4) / D_1) * m_1 - ((A_1 * b_3 + A_2 *␣
↪A_4*R_1) / D_1) * m_x * m_ - ((-A_1*b_4+A_3*A_4*R_1) / D_1) * n_x * n_
  f_m[2,3] = f_m[0,5]
  f_m[2,4] = -f_m[1,5]
  f_m[2,5] = -((A_5 * d_3 - A_6 * d_2) / (G*D_2)) * n_1 + ((A_4 * d_3 - A_6 *␣
↪d_1) / (G*D_2)) * n_x * m_ - ((A_4 * d_2 - A_5 * d_1) / (G*D_2)) * m_x * n_

  f_m[3,0] = -((b_3*d_1)/D_1)*n_1 +␣
↪((EI*(A_4*R_1*b_2-Q*(R**2+Q**2)*(A_5**2+A_6**2)))/D_1)*n_x*m_ +␣
↪((EI*(A_4*R_1*b_1-R*(R**2+Q**2)*(A_5**2+A_6**2)))/D_1)*m_x*n_
  f_m[3,1] = -((EI*R_1*A_4*b_3)/D_1)*m_1 + ((EI*R_1*A_4*b_3)/D_1)*m_x*m_ -␣
↪((EI*(b_3**2+b_4*(b_4-A_4*R_1)))/D_1)*n_x*n_
```

```
    f_m[3,2] = ((G*b_3*R_1)/D_1)*n_1 - (((G*(b_3*R-Q*(b_4-A_4*R_1))))/
↪D_1)*n_x*m_ + ((G*(b_3*Q+R*(b_4-A_4*R_1)))/D_1)*m_x*n_
    f_m[3,3] = f_m[0,0]
    f_m[3,4] = -f_m[1,0]
    f_m[3,5] = f_m[2,0]

    f_m[4,0] = -f_m[3,1]
    f_m[4,1] = ((EI*R_1*A_4*(d_3*R-d_2*Q))/D_2)*n_1 -␣
↪((EI*(b_3*(d_1*R-d_2*R_1)-b_4*(d_1*Q-d_3*R_1)))/D_2)*n_x*m_ -␣
↪((EI*(b_3*(d_1*Q-d_3*R_1)+b_4*(d_1*R-d_2*R_1)))/D_2)*m_x*n_
    f_m[4,2] = -((G*R_1*(d_3*R-d_2*Q))/D_2)*m_1 + ((G*R_1*(d_3*R-d_2*Q))/
↪D_2)*m_x*m_ - ((G*(R_1*(d_2*R+d_3*Q)-d_1*(R**2 + Q**2)))/D_2)*n_x*n_
    f_m[4,3] = - f_m[0,1]
    f_m[4,4] = f_m[1,1]
    f_m[4,5] = -f_m[2,1]

    f_m[5,0] = f_m[3,2]
    f_m[5,1] = -f_m[4,2]
    f_m[5,2] = -((G*R_1*(A_3*b_3+A_2*b_4))/D_1)*n_1 +␣
↪((G*(A_1*(R*b_3-Q*b_4)+A_4*R_1*(A_2*R+A_3*Q)))/D_1)*n_x*m_ -␣
↪((G*(A_1*(Q*b_3+R*b_4)+A_4*R_1*(Q*A_2-R*A_3)))/D_1)*m_x*n_
    f_m[5,3] = f_m[0,2]
    f_m[5,4] = -f_m[1,2]
    f_m[5,5] = f_m[2,2]

    # # Print the f_m matrix
    # print("f_m matrix:")
    # df = pd.DataFrame(f_m, columns=[f'Column {i+1}' for i in range(f_m.
↪shape[1])])
    # print(df)

    return f_m
```

[6]:
```
# Initialize K_G_M matrix
def K_G_M(K_M, n_ele):
    l_l = 3*n_ele-3
    m_l = 3*n_ele
    u_l = 3*n_ele+3

    K_G_M = np.zeros((u_l,u_l))

    # Populate K_G_M matrix
    for m in range(0,n_ele+1):
        if m == 0:
            for i in range(0,3):
                for j in range(0,6):
```

```
                    K_G_M[i,j]= K_M[i,j]
        elif m == n_ele:
            for i in range(m_l,u_l):
                for j in range(l_l,u_l):
                    K_G_M[i,j] = K_M[i-3*(m-1),j-3*(m-1)]
        else:
            for i in range(3*m,3*(m+1)):
                for j in range(3*(m-1),3*m):
                    K_G_M[i,j]= K_M[i-3*(m-1),j-3*(m-1)]
                for j in range(3*m,3*(m+1)):
                    K_G_M[i,j] = K_M[i-3*(m-1),j-3*(m-1)] + K_M[i-3*m,j-3*m]
                for j in range(3*(m+1),3*(m+2)):
                    K_G_M[i,j] = K_M[i-3*m,j-3*m]

    a_numbers = 27*n_ele+9-np.count_nonzero(K_G_M)

    if a_numbers==0:
        print("The matrix checks out!")
    else:
        print("Something wrong with the K_G_M - matrix bro!")

    #is_symmetric = Matrix.check_symmetric(K_G_M)
    #print("Is K_G_M symmetric?:", is_symmetric)

    # Set a threshold
    threshold = 1e-6

    # Change values smaller than the threshold to zero
    K_G_M[np.abs(K_G_M) < threshold] = 0

    # Print the K_G_M matrix
    # print("K_G_M matrix:")
    # df = pd.DataFrame(K_G_M, columns=[f'Column {i+1}' for i in range(K_G_M.
↪shape[1])])
    # print(df)


    return K_G_M
```

[7]:
```
def nodal_force_vector(EI, k, c, G, Phi, load_style, a, b, F, L, n_ele):

    if load_style == "Uniform":
        f_m = Matrix.f_m(EI, k, c, G, Phi, L)

        # # Print the f_m matrix
        # print("f_m matrix:")
```

```
        # df = pd.DataFrame(f_m, columns=[f'Column {i+1}' for i in range(f_m.
↪shape[1])])
        # print(df)

        D =␣
↪-f_m[1,4]*(f_m[2,3]**2-f_m[0,3]*f_m[2,5])+f_m[2,4]*(f_m[0,3]*f_m[2,4]-f_m[0,4]*f_m[2,3])-f_

        # Calculations for J_1, J_2, J_3
        J_1 = -((k + c) / G + c / Phi)
        J_2 = (c / Phi) * (Phi / EI + k / G)
        J_3 = -(k * c) / (EI * G)

        # Calculations for alpha, beta, delta
        alpha = (1/3) * (-((J_1**2)/3) + J_2)
        beta = (1/2) * (((2 * J_1**3) / 27) - ((J_1 * J_2) / 3) + J_3)
        delta = alpha ** 3 + beta ** 2

        # Print delta with a descriptive message
        # print(f"Delta value: {delta:.13f}")

        # Calculations for n, m, Q, R, R_1
        n = (np.sqrt(3)/2)*((Matrix.cuberoot(-beta+(delta)**(1/2)))-(Matrix.
↪cuberoot(-beta-(delta)**(1/2))))
        m = -(1/2)*((Matrix.cuberoot(-beta+(delta)**(1/2)))+(Matrix.
↪cuberoot(-beta-(delta)**(1/2)))+(2*J_1/3))
        Q = (((m**2+n**2)**(1/2)-m)/2)**(1/2)
        R = (((m**2+n**2)**(1/2)+m)/2)**(1/2)
        R_1 = ((Matrix.cuberoot(-beta+(delta)**(1/2)))+(Matrix.
↪cuberoot(-beta-(delta)**(1/2)))-(J_1/3))**(1/2)

        # Calculations for A_1, A_2, A_3
        A_1 = (1+k/c)-(G/c)*R_1**2
        A_2 = 2*R*Q*(G/c)
        A_3 = (1+k/c)-(G/c)*(R**2-Q**2)

        # Calculations for beta_1, beta_2, alpha_1, alpha_2
        beta_1 = -EI*(R**2-Q**2)+Phi
        beta_2 = 2*(EI)*R*Q
        alpha_1 = A_3*R+A_2*Q
        alpha_2 = A_2*R - A_3 *Q

        # Calculations for A_4, A_5, A_6
        A_4 = ((Phi*R_1)/(-EI*(R_1**2)+Phi))*A_1
        A_5 = ((alpha_1*beta_1 + alpha_2*beta_2)/(beta_1**2+beta_2**2))*Phi
        A_6 = ((alpha_1*beta_2 - alpha_2*beta_1)/(beta_1**2+beta_2**2))*Phi

        # Calculations for b_1, b_2, b_3, b_4, d_1, d_2, d_3
```

```
        b_1 = (R**2-Q**2)*A_5 -(2*R*Q)*A_6
        b_2 = (R**2-Q**2)*A_6 + (2*R*Q)*A_5
        b_3 = A_6*R+A_5*Q
        b_4 = A_5*R - A_6*Q
        d_1 = EI*(A_4*R_1**2)
        d_2 = EI * b_1
        d_3 = EI * b_2

        # Calculations for D_2, D_1
        D_2 = Q*(A_4*d_2-A_5*d_1)+R*(A_6*d_1-A_4*d_3)+R_1*(A_5*d_3-A_6*d_2)
        D_1 = b_3 *(A_3-A_1)+A_2*(b_4-A_4*R_1)

        qa = F
        qb = F

        m1a = np.cosh(R_1 * L * (a - 1))
        n1a = np.sinh(R_1 * L * (a - 1))
        ma = np.cosh(R * L * (a - 1))
        na = np.sinh(R * L * (a - 1))
        m1b = np.cosh(R_1 * L * (b - 1))
        n1b = np.sinh(R_1 * L * (b - 1))
        mb = np.cosh(R * L * (b - 1))
        nb = np.sinh(R * L * (b - 1))
        na_ = np.sin(Q * L * (a - 1))
        nb_ = np.sin(Q * L * (b - 1))
        ma_ = np.cos(Q * L * (a - 1))
        mb_ = np.cos(Q * L * (b - 1))

        m1a2 = np.cosh(R_1 * L * (-a))
        n1a2 = np.sinh(R_1 * L * (-a))
        ma2 = np.cosh(R * L * (-a))
        na2 = np.sinh(R * L * (-a))
        m1b2 = np.cosh(R_1 * L * (-b ))
        n1b2 = np.sinh(R_1 * L * (-b ))
        mb2 = np.cosh(R * L * (-b))
        nb2 = np.sinh(R * L * (-b))
        na_2 = np.sin(Q * L * (-a))
        nb_2 = np.sin(Q * L * (-b))
        ma_2 = np.cos(Q * L * (-a))
        mb_2 = np.cos(Q * L * (-b))

        qa_ = qa - ((qb - qa) / ((b - a) * L)) * (a * L)

        I1 = qa_ * (m1a - m1b) / R_1 + ((qb - qa) / ((b - a) * L)) * (((n1b - n1a) + R_1 * L * (a * m1a - b * m1b)) / R_1**2)
```

```
      I2 = qa_ * ((R * (ma_ * ma - mb_ * mb) + Q * (na_ * na - nb_ * nb)) /␣
↪(R**2 + Q**2)) + ((qb - qa) / ((b - a) * L)) * (1 / (R**2 + Q**2)**2)␣
↪*((R**2 + Q**2) * (a * L * (R * ma_ * ma + Q * na_ * na) - b * L * (R * mb_␣
↪* mb + Q * nb_ * nb)) +2 * R * Q * (nb_ * mb - na_ * ma) + (R**2 - Q**2) *␣
↪(nb * mb_ - na * ma_))
      I3 = qa_ * ((R*(na_*na-nb_*nb)-Q*(ma_*ma-mb_*mb))/(R**2+Q**2))+ ((qb -␣
↪qa) / ((b - a) * L)) * (1 / (R**2 + Q**2)**2) *␣
↪((R**2+Q**2)*(a*L*(R*na_*na-Q*ma_*ma)+b*L*(Q*mb_*mb-R*nb_*nb))␣
↪+2*R*Q*(na*ma_-nb*mb_)+(R**2-Q**2)*(nb_*mb-na_*ma))
      I4 = qa_ * ((n1b - n1a) / R_1) + ((qb - qa) / ((b - a) * L)) * (((m1a -␣
↪m1b) + R_1 * L * (b * n1b - a * n1a)) / R_1**2)
      I5 = qa_ * ((Q * (mb * nb_ - ma * na_) + R * (mb_ * nb - ma_ * na)) /␣
↪(R**2 + Q**2)) + ((qb - qa) / ((b - a) * L)) * (1 / (R**2 + Q**2)**2)␣
↪*((R**2 + Q**2) * (-a * L * (R * ma_ * na + Q * ma * na_) + b * L * (R * mb_␣
↪* nb + Q * mb * nb_)) +(R**2 - Q**2) * (ma * ma_ - mb * mb_) + 2 * R * Q *␣
↪(na * na_ - nb * nb_))
      I6 = qa_ * ((Q * (ma_ * na - mb_ * nb) + R * (mb * nb_ - ma * na_)) /␣
↪(R**2 + Q**2)) +(qb - qa) / ((b - a) * L) * (1 / (R**2 + Q**2)**2) *((R**2 +␣
↪Q**2) * (a * L * (Q * ma_ * na - R * ma * na_) + b * L * (R * mb * nb_ - Q *␣
↪mb_ * nb)) +(R**2 - Q**2) * (na_ * na - nb_ * nb) + 2 * R * Q * (mb * mb_ -␣
↪ma * ma_))

      I1_ = qa_ * (m1a2 - m1b2) / R_1 + ((qb - qa) / ((b - a) * L)) * (((n1b2␣
↪- n1a2) + R_1 * L * (a * m1a2 - b * m1b2)) / R_1**2)
      I2_ = qa_ * ((R * (ma_2 * ma2 - mb_2 * mb2) + Q * (na_2 * na2 - nb_2 *␣
↪nb2)) / (R**2 + Q**2)) + ((qb - qa) / ((b - a) * L)) * (1 / (R**2 +␣
↪Q**2)**2) *((R**2 + Q**2) * (a * L * (R * ma_2 * ma2 + Q * na_2 * na2) - b *␣
↪L * (R * mb_2 * mb2 + Q * nb_2 * nb2)) +2 * R * Q * (nb_2 * mb2 - na_2 *␣
↪ma2) + (R**2 - Q**2) * (nb2 * mb_2 - na2 * ma_2))
      I3_ = qa_ * ((R*(na_2*na2-nb_2*nb2)-Q*(ma_2*ma2-mb_2*mb2))/(R**2+Q**2))␣
↪+ ((qb - qa) / ((b - a) * L))* (1 / (R**2 + Q**2)**2) *␣
↪((R**2+Q**2)*(a*L*(R*na_2*na2-Q*ma_2*ma2)+b*L*(Q*mb_2*mb2-R*nb_2*nb2))␣
↪+2*R*Q*(na2*ma_2-nb2*mb_2)+(R**2-Q**2)*(nb_2*mb2-na_2*ma2))
      I4_ = qa_ * (n1b2- n1a2) / R_1 + ((qb - qa) / ((b - a) * L)) * ((m1a2 -␣
↪m1b2) + R_1 * L * (b * n1b2 - a * n1a2) / R_1**2)
      I5_ = qa_ * ((Q * (mb2 * nb_2 - ma2 * na_2) + R * (mb_2 * nb2 - ma_2 *␣
↪na2)) / (R**2 + Q**2)) + ((qb - qa) / ((b - a) * L)) * (1 / (R**2 +␣
↪Q**2)**2) *((R**2 + Q**2) * (-a * L * (R * ma_2 * na2 + Q * ma2 * na_2) + b␣
↪* L * (R * mb_2 * nb2 + Q * mb2 * nb_2)) +(R**2 - Q**2) * (ma2 * ma_2 - mb2␣
↪* mb_2) + 2 * R * Q * (na2 * na_2 - nb2 * nb_2))
      I6_ = qa_ * ((Q * (ma_2 * na2 - mb_2 * nb2) + R * (mb2 * nb_2 - ma2 *␣
↪na_2)) / (R**2 + Q**2)) + ((qb - qa) / ((b - a) * L)) * (1 / (R**2 +␣
↪Q**2)**2) *((R**2 + Q**2) * (a * L * (Q * ma_2 * na2 - R * ma2 * na_2) + b *␣
↪L * (R * mb2 * nb_2 - Q * mb_2 * nb2)) +(R**2 - Q**2) * (na_2 * na2 - nb_2*␣
↪nb2) + 2 * R * Q * (mb2 * mb_2 - ma2 * ma_2))
```

```
        IW = (A_1 * (A_5 * Q - A_6 * R) / D_2) * I1 +((A_3 * (A_6 * R_1 - A_4 *␣
↪Q) + A_2 * (A_5 * R_1 - A_4 * R)) / D_2) * I2 -((A_2 * (-A_6 * R_1 + A_4 *␣
↪Q) + A_3 * (A_5 * R_1 - A_4 * R)) / D_2 * I3)
        IWk = ((A_5 * Q - A_6 * R) / D_2) * I1 - ((A_4 * Q - A_6 * R_1) / D_2)␣
↪* I2 + ((A_4 * R - A_5 * R_1) / D_2) * I3
        IG = -((A_1 * A_2) / (EI*D_1)) * (I5 - I4) + ((A_3 * (A_1 - A_3) -␣
↪A_2**2) / (EI*D_1)) * I6

        IW_ = (A_1 * (A_5 * Q - A_6 * R) / D_2) * I1_ +((A_3 * (A_6 * R_1 - A_4␣
↪* Q) + A_2 * (A_5 * R_1 - A_4 * R)) / D_2) * I2_ -((A_2 * (-A_6 * R_1 + A_4␣
↪* Q) + A_3 * (A_5 * R_1 - A_4 * R)) / D_2 * I3_)
        IWk_ = ((A_5 * Q - A_6 * R) / D_2) * I1_ - ((A_4 * Q - A_6 * R_1) /␣
↪D_2) * I2_ + ((A_4 * R - A_5 * R_1) / D_2) * I3_
        IG_ = -((A_1 * A_2) / (EI*D_1)) * (I5_ - I4_) + ((A_3 * (A_1 - A_3) -␣
↪A_2**2) / (EI*D_1)) * I6_

        f_v = -IWk*(f_m[1,4]*f_m[2,3]+f_m[2,4]*f_m[0,4]) +␣
↪IW*(f_m[2,4]**2+f_m[1,4]*f_m[2,5]) - IG*(f_m[0,4]*f_m[2,5]-f_m[2,3]*f_m[2,4])
        f_vg = IWk*(f_m[0,4]**2+f_m[1,4]*f_m[0,3]) -␣
↪IW*(f_m[1,4]*f_m[2,3]+f_m[0,4]*f_m[2,4]) -␣
↪IG*(f_m[0,3]*f_m[2,4]-f_m[0,4]*f_m[2,3])
        f_mo = -IG*(f_m[2,3]**2-f_m[0,3]*f_m[2,5])-␣
↪IW*(f_m[2,3]*f_m[2,4]-f_m[0,4]*f_m[2,5]) -␣
↪IWk*(f_m[0,4]*f_m[2,3]-f_m[0,3]*f_m[2,4])

        f_v_ = -IWk_*(f_m[1,4]*f_m[2,3]+f_m[2,4]*f_m[0,4]) +␣
↪IW_*(f_m[2,4]**2+f_m[1,4]*f_m[2,5]) -␣
↪IG_*(f_m[0,4]*f_m[2,5]-f_m[2,3]*f_m[2,4])
        f_vg_ = IWk_*(f_m[0,4]**2+f_m[1,4]*f_m[0,3]) -␣
↪IW_*(f_m[1,4]*f_m[2,3]+f_m[0,4]*f_m[2,4]) -␣
↪IG_*(f_m[0,3]*f_m[2,4]-f_m[0,4]*f_m[2,3])
        f_mo_ = -IG_*(f_m[2,3]**2-f_m[0,3]*f_m[2,5])-␣
↪IW_*(f_m[2,3]*f_m[2,4]-f_m[0,4]*f_m[2,5]) -␣
↪IWk_*(f_m[0,4]*f_m[2,3]-f_m[0,3]*f_m[2,4])


        V_0 = f_v/D
        M_0 = f_mo/D
        V_G0 = f_vg/D
        V_L = f_v_/D
        M_L = -f_mo_/D
        V_GL = f_vg_/D

        nfv = np.array([[V_0],[M_0],[V_G0],[V_L],[M_L],[V_GL]])

    elif load_style == "Concentrated":
```

```
        f_m = Matrix.f_m(EI, k, c, G, Phi, L)
        f_m_al = Matrix.f_m(EI, c, k, G, Phi, ((1-a)*L))
        f_m_al_ = Matrix.f_m(EI, c, k, G, Phi, (a*L))

        D =␣
↪-f_m[1,4]*(f_m[2,3]**2-f_m[0,3]*f_m[2,5])+f_m[2,4]*(f_m[0,3]*f_m[2,4]-f_m[0,4]*f_m[2,3])-f_

        f_v = f_m_al[2,3]*(f_m[1,4]*f_m[2,3]+f_m[2,4]*f_m[0,4]) -␣
↪f_m_al[0,3]*(f_m[2,4]**2+f_m[1,4]*f_m[2,5]) +␣
↪f_m_al[1,3]*(f_m[0,4]*f_m[2,5]-f_m[2,3]*f_m[2,4])
        f_vg = -f_m_al[2,3]*(f_m[0,4]**2+f_m[1,4]*f_m[0,3]) +␣
↪f_m_al[0,3]*(f_m[1,4]*f_m[2,3]+f_m[0,4]*f_m[2,4]) +␣
↪f_m_al[1,3]*(f_m[0,3]*f_m[2,4]-f_m[0,4]*f_m[2,3])
        f_mo = f_m_al[1,3]*(f_m[2,3]**2-f_m[0,3]*f_m[2,5]) +␣
↪f_m_al[0,3]*(f_m[2,3]*f_m[2,4]-f_m[0,4]*f_m[2,5]) +␣
↪f_m_al[2,3]*(f_m[0,4]*f_m[2,3]-f_m[0,3]*f_m[2,4])

        f_v_ = f_m_al_[2,3]*(f_m[1,4]*f_m[2,3]+f_m[2,4]*f_m[0,4]) -␣
↪f_m_al_[0,3]*(f_m[2,4]**2+f_m[1,4]*f_m[2,5]) +␣
↪f_m_al_[1,3]*(f_m[0,4]*f_m[2,5]-f_m[2,3]*f_m[2,4])
        f_vg_ = -f_m_al_[2,3]*(f_m[0,4]**2+f_m[1,4]*f_m[0,3]) +␣
↪f_m_al_[0,3]*(f_m[1,4]*f_m[2,3]+f_m[0,4]*f_m[2,4]) +␣
↪f_m_al_[1,3]*(f_m[0,3]*f_m[2,4]-f_m[0,4]*f_m[2,3])
        f_mo_ = f_m_al_[1,3]*(f_m[2,3]**2-f_m[0,3]*f_m[2,5]) +␣
↪f_m_al_[0,3]*(f_m[2,3]*f_m[2,4]-f_m[0,4]*f_m[2,5]) +␣
↪f_m_al_[2,3]*(f_m[0,4]*f_m[2,3]-f_m[0,3]*f_m[2,4])

        V_0 = F*(f_v/D)
        M_0 = F*(f_mo/D)
        V_G0 = F*(f_vg/D)
        V_L = F*(f_v_/D)
        M_L = -F*(f_mo_/D)
        V_GL = F*(f_vg_/D)

        nfv = np.array([V_0,M_0,V_G0,V_L,M_L,V_GL])

    else:
        print('no correct load_style was implemented')


    # print(f'I = {I1, I2, I3, I4, I5, I6}')
    # print(f'I_ = {I1_, I2_, I3_, I4_, I5_, I6_}')
    # print(f'D = {D}')

    return nfv
```

[8]:
```
def find_zero_indices(u):
    n = len(u)
    i_loc = np.empty(n, dtype=np.int64)
    count = 0
    for i in range(n):
        if u[i] == 0:
            i_loc[count] = i
            count += 1
    return i_loc[:count]
```

[9]:
```
def unknown_displacements(K_G_M, u , F):
    # Identify indices to remove based on u
    i_loc = find_zero_indices(u)

    # Remove rows and columns from K_L
    Kin = np.delete(K_G_M, i_loc, axis=1)
    Kin = np.delete(Kin, i_loc, axis=0)

    # Remove elements from F
    F_ex = np.delete(F, i_loc)
    load = np.sum(F_ex)

    # Solve for the unknown displacements u_r
    u_r = np.linalg.solve(Kin, F_ex)

    u_f = []
    count = 0

    for i in range (len(u)):
        if u[i] == 0:
            u_f = np.append(u_f,u[i])
            count += 1
        else:
            u_f = np.append(u_f,u_r[i-count])

    return u_f
```

[12]:
```
def defo_profiles(load_style, a, b, L_F, n_ele, L_ele, dx, u_f, F_ex, EI, k, c,␣
↪G, Phi):
    x = np.arange(0,L_ele+dx,dx)
    y_p = np.zeros((6,len(x)))
    y = np.zeros((6,len(x)))
    f_i = np.zeros((6,1))
    y_0 = np.array([u_f[0], 0, u_f[2], F_ex[0], F_ex[1], F_ex[2]]) # Initial␣
↪conditions
    # y_0 = np.array([0, 0, 0, F_ex[0], F_ex[1], F_ex[2]]) # Initial conditions
    # y_0 = np.array([0,0,0,0,0,0])
```

```
  if load_style == 'Concentrated':
      for j in range(len(x)):
          if x[j] < (a*L_ele):
              y_p[:,j] = 0
          elif x[j] >= (a*L_ele):
              f_r = Matrix.f_m(EI, k, c, G, Phi, (x[j] - a * L_ele))
              f_i[0] = f_r[0,3]
              f_i[1] = f_r[1,3]
              f_i[2] = f_r[2,3]
              f_i[3] = f_r[3,3]
              f_i[4] = f_r[4,3]
              f_i[5] = f_r[5,3]
              for i in range(0,6):
                  y_p[i,j] = L_F * f_i[i].item()

          f_m = Matrix.f_m(EI, k, c, G , Phi, x[j])
          y[:,j] = f_m.dot(y_0) + y_p[:,j]

  elif load_style == 'Uniform':
       # Calculations for J_1, J_2, J_3
      J_1 = -((k + c) / G + c / Phi)
      J_2 = (c / Phi) * (Phi / EI + k / G)
      J_3 = -(k * c) / (EI * G)

      # Calculations for alpha, beta, delta
      alpha = (1/3) * (-((J_1**2)/3) + J_2)
      beta = (1/2) * (((2 * J_1**3) / 27) - ((J_1 * J_2) / 3) + J_3)
      delta = alpha ** 3 + beta ** 2

      # Calculations for n, m, Q, R, R_1
      n = (np.sqrt(3)/2)*((Matrix.cuberoot(-beta+(delta)**(1/2)))-(Matrix.
↪cuberoot(-beta-(delta)**(1/2))))
      m = -(1/2)*((Matrix.cuberoot(-beta+(delta)**(1/2)))+(Matrix.
↪cuberoot(-beta-(delta)**(1/2)))+(2*J_1/3))
      Q = (((m**2+n**2)**(1/2)-m)/2)**(1/2)
      R = (((m**2+n**2)**(1/2)+m)/2)**(1/2)
      R_1 = ((Matrix.cuberoot(-beta+(delta)**(1/2)))+(Matrix.
↪cuberoot(-beta-(delta)**(1/2)))-(J_1/3))**(1/2)

      # Calculations for A_1, A_2, A_3
      A_1 = (1+k/c)-(G/c)*R_1**2
      A_2 = 2*R*Q*(G/c)
      A_3 = (1+k/c)-(G/c)*(R**2-Q**2)

      # Calculations for beta_1, beta_2, alpha_1, alpha_2
      beta_1 = -EI*(R**2-Q**2)+Phi
```

```
        beta_2 = 2*(EI)*R*Q
        alpha_1 = A_3*R+A_2*Q
        alpha_2 = A_2*R - A_3 *Q

        # Calculations for A_4, A_5, A_6
        A_4 = ((Phi*R_1)/(-EI*(R_1**2)+Phi))*A_1
        A_5 = ((alpha_1*beta_1 + alpha_2*beta_2)/(beta_1**2+beta_2**2))*Phi
        A_6 = ((alpha_1*beta_2 - alpha_2*beta_1)/(beta_1**2+beta_2**2))*Phi

        # Calculations for b_1, b_2, b_3, b_4, d_1, d_2, d_3
        b_1 = (R**2-Q**2)*A_5 -(2*R*Q)*A_6
        b_2 = (R**2-Q**2)*A_6 + (2*R*Q)*A_5
        b_3 = A_6*R+A_5*Q
        b_4 = A_5*R - A_6*Q
        d_1 = EI*(A_4*R_1**2)
        d_2 = EI * b_1
        d_3 = EI * b_2

        # Calculations for D_2, D_1
        D_2 = Q*(A_4*d_2-A_5*d_1)+R*(A_6*d_1-A_4*d_3)+R_1*(A_5*d_3-A_6*d_2)
        D_1 = b_3 *(A_3-A_1)+A_2*(b_4-A_4*R_1)

        # n_x = np.sinh(0)
        # m_x = np.cosh(0)
        # n_ = np.sin(0)
        # m_ = np.cos(0)
        # n_1 = np.sinh(0)
        # m_1 = np.cosh(0)

        for j in range(len(x)):
            for i in range(0,6):
                # if i == 0:
                #     f_i[i,:] = L_F * (-(A_1 * (A_5 * Q - A_6 * R) / D_2) *
↪n_1 - ((A_3 * (A_6 * R_1 - A_4 * Q) + A_2 * (A_5 * R_1 - A_4 * R)) / D_2) *
↪n_x * m_ + ((A_2 * (-A_6 * R_1 + A_4 * Q) + A_3 * (A_5 * R_1 - A_4 * R)) /
↪D_2) * m_x * n_)
                #     y_p[i,j] = f_i[i,:] * x[j]
                # if i == 1:
                #     f_i[i,:] = L_F*(-(((A_1 * A_2) / (EI*D_1)) * m_1 - ((A_1
↪* A_2) / (EI*D_1)) * m_x * m_ + ((A_3 * (A_1 - A_3) - A_2 ** 2) / (EI*D_1))
↪* n_x * n_))
                #     y_p[i,j] = f_i[i,:] * x[j]
                # if i == 2:
                #     f_i[i,:] = L_F*(-((A_5 * Q - A_6 * R) / D_2) * n_1 +
↪((A_4 * Q - A_6 * R_1) / D_2) * n_x * m_ - ((A_4 * R - A_5 * R_1) / D_2) *
↪m_x * n_)
                #     y_p[i,j] = f_i[i,:] * x[j]
```

```
                # if i == 3:
                #     f_i[i,:] = L_F*(-(A_1 * b_3 / D_1) * m_1 + ((A_3 * b_3 -␣
↪A_2 * (A_4 * R_1 - b_4)) / D_1) * m_x * m_ + ((A_2 * b_3 + A_3 * (A_4 * R_1␣
↪- b_4)) / D_1) * n_x * n_)
                #     y_p[i,j] = f_i[i,:] * x[j]
                # if i == 4:
                #     f_i[i,:] =  L_F*(-((A_1 * (d_2 * Q - d_3 * R) / D_2) *␣
↪n_1 - ((A_3 * (d_1 * Q - d_3 * R_1) + A_2 * (d_1 * R - d_2 * R_1)) / D_2) *␣
↪n_x * m_ - ((A_3 * (d_2 * R_1 - d_1 * R) + A_2 * (d_1 * Q - d_3 * R_1)) /␣
↪D_2) * m_x * n_))
                #     y_p[i,j] = f_i[i,:] * x[j]
                # if i == 5:
                #     f_i[i,:] = L_F*((A_1 * (A_3 * b_3 + A_2 * b_4)) / D_1) *␣
↪m_1 - ((A_1 * (A_3 * b_3 + A_2 * b_4)) / D_1) * m_x * m_ - ((A_4 * R_1 *␣
↪(A_2**2 + A_3**2) + A_1 * (A_2 * b_3 - A_3 * b_4)) / D_1) * n_x * n_
                #     y_p[i,j] = f_i[i,:] * x[j]
                if i == 0:
                    def integrand(z):
                        n_x = np.sinh(R*z)
                        m_x = np.cosh(R*z)
                        n_ = np.sin(Q*z)
                        m_ = np.cos(Q*z)
                        n_1 = np.sinh(R_1*z)
                        m_1 = np.cosh(R_1*z)
                        f_i =L_F * (-(A_1 * (A_5 * Q - A_6 * R) / D_2) * n_1 -␣
↪((A_3 * (A_6 * R_1 - A_4 * Q) + A_2 * (A_5 * R_1 - A_4 * R)) / D_2) * n_x *␣
↪m_ + ((A_2 * (-A_6 * R_1 + A_4 * Q) + A_3 * (A_5 * R_1 - A_4 * R)) / D_2) *␣
↪m_x * n_)
                        return f_i

                    y_p[i,:] = quad(integrand, x[0], x[j])[0]
                elif i == 1:
                    def integrand(z):
                        n_x = np.sinh(R*z)
                        m_x = np.cosh(R*z)
                        n_ = np.sin(Q*z)
                        m_ = np.cos(Q*z)
                        n_1 = np.sinh(R_1*z)
                        m_1 = np.cosh(R_1*z)
                        f_i = L_F*(-(((A_1 * A_2) / (EI*D_1)) * m_1 - ((A_1 *␣
↪A_2) / (EI*D_1)) * m_x * m_ + ((A_3 * (A_1 - A_3) - A_2**2) / (EI*D_1)) *␣
↪n_x * n_))
                        return f_i
                    y_p[i,:] = quad(integrand, x[0], x[j])[0]
                elif i == 2:
                    def integrand(z):
                        n_x = np.sinh(R*z)
```

```
                    m_x = np.cosh(R*z)
                    n_ = np.sin(Q*z)
                    m_ = np.cos(Q*z)
                    n_1 = np.sinh(R_1*z)
                    m_1 = np.cosh(R_1*z)
                    f_i = L_F*(-((A_5 * Q - A_6 * R) / D_2) * n_1 + ((A_4 *␣
↪Q - A_6 * R_1) / D_2) * n_x * m_ - ((A_4 * R - A_5 * R_1) / D_2) * m_x * n_)
                    return f_i
                y_p[i,:] = quad(integrand, x[0], x[j])[0]
            elif i == 3:
                def integrand(z):
                    n_x = np.sinh(R*z)
                    m_x = np.cosh(R*z)
                    n_ = np.sin(Q*z)
                    m_ = np.cos(Q*z)
                    n_1 = np.sinh(R_1*z)
                    m_1 = np.cosh(R_1*z)
                    f_i = L_F*(-(A_1 * b_3 / D_1) * m_1 + ((A_3 * b_3 - A_2␣
↪* (A_4 * R_1 - b_4)) / D_1) * m_x * m_ + ((A_2 * b_3 + A_3 * (A_4 * R_1 -␣
↪b_4)) / D_1) * n_x * n_)
                    return f_i
                y_p[i,:] = quad(integrand, x[0], x[j])[0]
            elif i == 4:
                def integrand(z):
                    n_x = np.sinh(R*z)
                    m_x = np.cosh(R*z)
                    n_ = np.sin(Q*z)
                    m_ = np.cos(Q*z)
                    n_1 = np.sinh(R_1*z)
                    m_1 = np.cosh(R_1*z)
                    f_i =  L_F*(-((A_1 * (d_2 * Q - d_3 * R) / D_2) * n_1 -␣
↪((A_3 * (d_1 * Q - d_3 * R_1) + A_2 * (d_1 * R - d_2 * R_1)) / D_2) * n_x *␣
↪m_ - ((A_3 * (d_2 * R_1 - d_1 * R) + A_2 * (d_1 * Q - d_3 * R_1)) / D_2) *␣
↪m_x * n_))
                    return f_i
                y_p[i,:] = quad(integrand, x[0], x[j])[0]
            elif i == 5:
                def integrand(z):
                    n_x = np.sinh(R*z)
                    m_x = np.cosh(R*z)
                    n_ = np.sin(Q*z)
                    m_ = np.cos(Q*z)
                    n_1 = np.sinh(R_1*z)
                    m_1 = np.cosh(R_1*z)
                    f_i = L_F*(((A_1 * (A_3 * b_3 + A_2 * b_4)) / D_1) *␣
↪m_1 - ((A_1 * (A_3 * b_3 + A_2 * b_4)) / D_1) * m_x * m_ - ((A_4 * R_1 *␣
↪(A_2**2 + A_3**2) + A_1 * (A_2 * b_3 - A_3 * b_4)) / D_1) * n_x * n_)
```

```
                        return f_i
                    y_p[i,:] = quad(integrand, x[0], x[j])[0]

            f_m = Matrix.f_m(EI, k, c, G, Phi, x[j])
            y[:,j] = np.matmul(f_m,y_0) #+ y_p[:,j]

    return y, x
```

[ ]:

[13]:
```
print("Import Matrix was succesfull")
```

```
Import Matrix was succesfull
```

H

Appendix H

The Python script for the calculations for the consolidation is shown here.

Consolidation_models

May 22, 2025

[1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import os
import math
import matplotlib.pyplot as plt
from scipy.stats import truncnorm
from numba import jit
import matplotlib as mpl
import Consolidation_models
import import_ipynb
import time
import pickle
import Matrix
from Regular_element_buoyancy_calculation_Fehmarn import F_res_3_reg,␣
 ↪Bal_con_req_reg
from Special_element_buoyancy_calculation_Fehmarn import F_res_3_spec,␣
 ↪Bal_con_req_spec, A_con_csA
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
Cell In[1], line 10
      8 from numba import jit
      9 import matplotlib as mpl
---> 10 import Consolidation_models
     11 import import_ipynb
     12 import time

ModuleNotFoundError: No module named 'Consolidation_models'
```

[1]:
```
def final_config(Zone_type, load_style, Parameters, el_type, h, t_l, d_sc,␣
 ↪q_rel, F_res_3, Bal_con_req, y_sat_u, t_l_u, gamma_foun, gamma_w,␣
 ↪gamma_prot, t_lay, t_foun, t_prot, t_u_prof, bou_con, M, N, K, t_c,␣
 ↪time_days, start_times, t_end_EOP, sub_t, t_end, t_0, t, alfa, std_dev,␣
 ↪num_simulations, kappa, E_b, v_b, W_ele, L_b, a_u, b_u, dx, times):
```

```
   sim_lay = Consolidation_models.monte_carlo_soil_layers(t_l, d_sc, std_dev,␣
↪num_simulations)
   x = np.arange(0,L_b+dx,dx)

   Prim_set = []
   Tot_prim_set = []
   Sec_set = []
   Tot_sec_set = []
   Ini_set = []
   Tot_lay_set = []
   Tot_zone_set = []
   Set = []

   n = np.zeros((len(times),len(x)))
   closest_locations = [np.abs(t - value).argmin() for value in times]
   common_times = np.intersect1d(time_days, t)
   indices_time_days = np.where(np.in1d(time_days, common_times))[0]
   indices_t = np.where(np.in1d(t, common_times))[0]

   for i in range(num_simulations):
       Zone = Consolidation_models.total(Zone_type, Parameters, el_type, h, np.
↪append(t_l[0],sim_lay[i]), q_rel, F_res_3_reg, Bal_con_req, y_sat_u, t_l_u,␣
↪gamma_foun, gamma_w, gamma_prot, t_lay, t_foun, t_prot, t_u_prof, bou_con,␣
↪M, N, K, t_c, time_days, start_times, t_end_EOP, sub_t, t_end, t_0, t, alfa)
       Tot_ini = timo_kerr_DEM(load_style, Parameters, np.
↪append(t_l[0],sim_lay[i]), kappa, E_b, v_b, W_ele, h, np.sum(sim_lay[i]),␣
↪L_b, a_u, b_u, Zone[7], dx)
       Prim_set.append(Zone[2])
       Tot_prim_set.append(np.sum(Zone[2], axis = 0))
       Sec_set.append(Zone[3])
       Tot_sec_set.append(np.sum(Zone[3], axis = 0))
       Ini_set.append(Tot_ini[0])

       # k = np.append(Zone[3][:,indices_t] + Zone[2][:,indices_time_days],␣
↪Zone[3][:,(indices_t[-1]):-1], axis = 1)
       # m = np.append(np.sum(Zone[3], axis = 0)[indices_t] + np.sum(Zone[2],␣
↪axis = 0)[indices_time_days], np.sum(Zone[3], axis = 0)[(indices_t[-1]):-1])

       k = np.append(Zone[3][:,indices_t] + Zone[2][:,indices_time_days], np.
↪append(Zone[2][:,-2].reshape(-1, 1),Zone[2][:,-1].reshape(-1, 1), axis = 1),␣
↪axis = 1)
       k = np.append(k, Zone[3][:,(indices_t[-1])+2:-1], axis = 1)
       m = np.sum(k, axis = 0)

       Tot_lay_set.append(k)
       Tot_zone_set.append(m)
```

```
        for j in range(len(times)):
            n[j,:] = m[closest_locations[j]]*np.
↪ones(len(Tot_ini[0]))+Tot_ini[0]

        Set.append(n)

    data_to_save = {
        'num_simulations': num_simulations,
        't_l': t_l,
        'time_days' : time_days,
        't' : t,
        'times': times,
        'x' : x,
        'Prim_set': Prim_set,
        'Tot_prim_set': Tot_prim_set,
        'Sec_set': Sec_set,
        'Tot_sec_set': Tot_sec_set,
        'Ini_set': Ini_set,
        'Tot_lay_set': Tot_lay_set,
        'Tot_zone_set': Tot_zone_set,
        'Set': Set,
        'Zone': Zone
    }

    with␣
↪open(f'simulation_data_{Zone_type}_{el_type}_with_{num_simulations}_simulations_and_dev_{st
↪pkl', 'wb') as file:
        pickle.dump(data_to_save, file)

    return Zone
```

[ ]:
```
def final_config_error(datafile):

    with open(datafile, 'rb') as file:
        loaded_data = pickle.load(file)

    num_simulations = loaded_data['num_simulations']
    t_l = loaded_data['t_l']
    time_days = loaded_data['time_days']
    t = loaded_data['t']
    times = loaded_data['times']
    x = loaded_data['x']

    Zone = loaded_data['Zone']
    Prim_set = loaded_data['Prim_set']
    Tot_prim_set = loaded_data['Tot_prim_set']
```

```
    Sec_set = loaded_data['Sec_set']
    Tot_sec_set = loaded_data['Tot_sec_set']
    Ini_set = loaded_data['Ini_set']
    Tot_lay_set = loaded_data['Tot_lay_set']
    Tot_zone_set = loaded_data['Tot_zone_set']
    Set = loaded_data['Set']

    Zone_prim_error = [np.zeros(len(time_days)) for _ in range(4)]
    Zone_sec_error = [np.zeros(len(t)) for _ in range(4)]
    Zone_tot_error = [np.zeros(len(t)) for _ in range(4)]
    Ini_error = [np.zeros(len(x)) for _ in range(4)]
    Set_error =  np.empty((4, len(times), len(x)))

    closest_locations = [np.abs(t - value).argmin() for value in times]

    Prim_set = np.array(Prim_set).
↪reshape(num_simulations*(len(t_l)-1),len(time_days))
    Sec_set = np.array(Sec_set).reshape(num_simulations*(len(t_l)-1),len(t))
    Tot_lay_set = np.array(Tot_lay_set).
↪reshape(num_simulations*(len(t_l)-1),len(t))
    Set = np.array(Set).reshape(num_simulations*(len(times)),len(x))
    # print(Set[0:12,:])

    Prim_error = Consolidation_models.calculate_error(Prim_set, len(t_l)-1)
    Sec_error = Consolidation_models.calculate_error(Sec_set, len(t_l)-1)
    Tot_lay_error = Consolidation_models.calculate_error(Tot_lay_set ,␣
↪len(t_l)-1)
    #Set_error = Consolidation_models.calculate_error(Set, len(times))

    Zone_prim_error[0] = np.mean(Tot_prim_set, axis = 0)
    Zone_prim_error[1] = np.std(Tot_prim_set, axis = 0)
    Zone_prim_error[2] = np.percentile(Tot_prim_set, q=5, axis = 0)
    Zone_prim_error[3] = np.percentile(Tot_prim_set, q=95, axis = 0)

    Zone_sec_error[0] = np.mean(Tot_sec_set, axis =0)
    Zone_sec_error[1] = np.std(Tot_sec_set, axis =0)
    Zone_sec_error[2] = np.percentile(Tot_sec_set, q=5, axis = 0)
    Zone_sec_error[3] = np.percentile(Tot_sec_set, q=95, axis = 0)

    Zone_tot_error[0] = np.mean(Tot_zone_set, axis =0)
    Zone_tot_error[1] = np.std(Tot_zone_set, axis =0)
    Zone_tot_error[2] = np.percentile(Tot_zone_set, q=5, axis = 0)
    Zone_tot_error[3] = np.percentile(Tot_zone_set, q=95, axis = 0)

    Ini_error[0] = np.mean(Ini_set, axis = 0)
    Ini_error[1] = np.std(Ini_set, axis = 0)
    Ini_error[2] = np.percentile(Ini_set, q=5, axis = 0)
```

```
    Ini_error[3] = np.percentile(Ini_set, q=95, axis = 0)


    Final = np.empty((len(times),num_simulations,len(Ini_set[0])))

    for j in range(len(times)):
        Final[j][:,:] = (np.array(Tot_zone_set)[:,closest_locations[j]])[:, np.
↪newaxis] + Ini_set
        Set_error[0][j,:] = Zone_tot_error[0][closest_locations[j]]*np.
↪ones(len(Ini_error[0])) + Ini_error[0]
        Set_error[1][j,:] = np.
↪sqrt((Zone_tot_error[1][closest_locations[j]])**2*np.ones(len(Ini_error[1]))␣
↪+ (Ini_error[1])**2)
        if len(t_l) <= 2:
            Set_error[2][j,:] = np.zeros(len(Ini_error[0]))
            Set_error[3][j,:] = np.zeros(len(Ini_error[0]))
        else:
            Set_error[2][j,:] = Zone_tot_error[0][closest_locations[j]]*np.
↪ones(len(Ini_error[0])) + Ini_error[0] - 1.645*(np.
↪sqrt((Zone_tot_error[2][closest_locations[j]])**2*np.ones(len(Ini_error[2]))␣
↪+ (Ini_error[2])**2))
            Set_error[3][j,:] = Zone_tot_error[0][closest_locations[j]]*np.
↪ones(len(Ini_error[0])) + Ini_error[0] + 1.645*(np.
↪sqrt((Zone_tot_error[2][closest_locations[j]])**2*np.ones(len(Ini_error[2]))␣
↪+ (Ini_error[2])**2))

    return Zone, Prim_error, Sec_error, Tot_lay_error, Zone_prim_error,␣
↪Zone_sec_error, Zone_tot_error, Ini_set, Ini_error, Set, Set_error,␣
↪Prim_set, Sec_set, Tot_lay_set, Tot_prim_set, Tot_sec_set, Tot_zone_set,␣
↪Final
```

[ ]:
```
def timo_kerr_DEM(load_style, Parameters, t_l, kappa, E_b, v_b, B, H, H_s, L_b,␣
↪a, b, F, dx):
    par = np.delete(Parameters, 0, 0)
    i_zero = Consolidation_models.find_zero_index_arr(np.array(t_l))

    if not i_zero == None:
        par = np.delete(par, i_zero, axis = 0)
        t_l = np.delete(t_l, i_zero, axis = 0)

    t_l = np.delete(t_l,0,0)

    E_s = np.sum(t_l[:]*par[:,5])/np.sum(t_l)
    v_s = np.sum(t_l[:]*par[:,12])/np.sum(t_l)

    k_s = (E_s/H_s)*(1/(0.95*(1-v_s)))
```

```
    EI, k, c, G, Phi, A = Matrix.PC(k_s, kappa, E_b, v_b, E_s, v_s, B, H, H_s)
    # print(F, EI, k, c, G, Phi, A)
    K_M = Matrix.K_M(EI, k, c, G, Phi, L_b)[0]     # Calculate K_M matrix
    K_G_M = K_M
    u = np.array([1,0,1,1,0,1])                    #u = {w1, psi1, wk1, w2,␣
↪psi2, wk2, w3, psi3, wk3} (displacement vector u)

    if load_style == 'Concentrated':
        nfv = Matrix.nodal_force_vector(EI, k, c, G, Phi, "Concentrated", a, b,␣
↪F, L_b, 1)    # Nodal force vector for the forces at x=0 and x=L
        F_ex = np.array([nfv[0].item(), nfv[1].item(), nfv[2].item(),
                         nfv[3].item(), nfv[4].item(), nfv[5].item()])          ␣
↪                                          # F = {V1, M1, VG1, V2, M2, VG2, V3,␣
↪M3, VG3} (Force vector F)
        u_r = Matrix.unknown_displacements(K_G_M, u, F_ex)                      ␣
↪                                              # Results of displacements␣
↪calculated by FEM
        y,x = Matrix.defo_profiles("Concentrated", a, b, F, 1, L_b, dx, u_r,␣
↪F_ex, EI, k, c, G, Phi)
        Tot_set_ini = y[0,:] + y[2,:]
    else:
        nfv = Matrix.nodal_force_vector(EI, k, c, G, Phi, "Uniform", a, b, F,␣
↪L_b, 1)           # Nodal force vector for the forces at x=0 and x=L
        F_ex = np.array([nfv[0].item(), nfv[1].item(), nfv[2].item(), -nfv[3].
↪item(), nfv[4].item(), -nfv[5].item()])
        u_r = Matrix.unknown_displacements(K_G_M, u, F_ex)                      ␣
↪                    # Results of displacements calculated by FEM
        y, x = Matrix.defo_profiles("Uniform", a, b, F, 1, L_b, dx, u_r, F_ex,␣
↪EI, k, c, G, Phi)

        Tot_set_ini = y[0,:] + y[2,:]

    return Tot_set_ini, y, u_r, F_ex, x
```

[ ]:
```
def calculate_error(arr, layers):
    means = []
    std_devs = []
    percentiles_5 = []
    percentiles_95 = []

    for i in range(layers):
        layer_data = arr[i::layers]
        mean = np.mean(layer_data, axis=0)
        std_dev = np.std(layer_data, axis=0)
        percentile_5 = np.percentile(layer_data, q=5, axis=0)
        percentile_95 = np.percentile(layer_data, q=95, axis=0)
```

```
        means.append(mean)
        std_devs.append(std_dev)
        percentiles_5.append(percentile_5)
        percentiles_95.append(percentile_95)

    return means, std_devs, percentiles_5, percentiles_95
```

[ ]:
```
def monte_carlo_soil_layers(t_l, d_sc, std_dev, num_simulations):
    total_height = d_sc  # Total height of the soil column
    num_layers = len(t_l) - 1  # Number of soil layers
    avg_thicknesses = t_l[1:]  # Different average thicknesses for each layer

    simulations = np.zeros((num_simulations, num_layers))

    for sim in range(num_simulations):
        remaining_height = total_height

        for i in range(num_layers - 1):
            thickness = 0
            while thickness <= 0.1 or thickness >= remaining_height:
                mu = np.log(avg_thicknesses[i])
                sigma = std_dev
                a, b = (np.log(0.1) - mu) / sigma, (np.log(remaining_height) -␣
 ↪mu) / sigma
                thickness = np.exp(truncnorm(a, b, loc=mu, scale=sigma).rvs())
            simulations[sim, i] = thickness
            remaining_height -= thickness

        simulations[sim, num_layers - 1] = remaining_height
    print(f'Simulation is completed')

    return simulations
```

[ ]:
```
# @jit(nopython= True)
# def monte_carlo_soil_layers(t_l, d_sc, std_dev, num_simulations):
#     total_height = d_sc  # Total height of the soil column
#     num_layers = len(t_l) - 1  # Number of soil layers
#     avg_thicknesses = t_l[1:]  # Different average thicknesses for each layer

#     simulations = np.zeros((num_simulations, num_layers))

#     for sim in range(num_simulations):
#         remaining_height = total_height

#         for i in range(num_layers - 1):
#             thickness = 0
```

```
#             while thickness <= 1.5 or thickness >= remaining_height:
#                 thickness = np.random.normal(avg_thicknesses[i], std_dev)
#             simulations[sim, i] = thickness
#             remaining_height -= thickness

#         simulations[sim, num_layers - 1] = remaining_height

#     return simulations
```

[ ]:
```
def find_zero_index_lst(lst):
    try:
        index = lst.index(0)
        return index
    except ValueError:
        return None
```

[ ]:
```
def find_zero_index_arr(arr):
    result = np.where(arr == 0)
    if result[0].size > 0:
        return result[0][0]
    else:
        return None
```

[ ]:
```
def total(Zone, Parameters, el_type, h, t_l, q_rel, F_res_3, Bal_con_req,␣
 ↪y_sat_u, t_l_u, gamma_foun, gamma_w, gamma_prot, t_lay, t_foun, t_prot,␣
 ↪t_u_prof, bou_con, M, N, K, t_c, time_days, start_times, t_end_EOP, sub_t,␣
 ↪t_end, t_0, t, alfa):

    Color = [param['Color'] for param in Parameters]
    Soil = [param['Name'] for param in Parameters]
    y_sat = [param['y_sat'] for param in Parameters]
    c_v = [param['c_v'] for param in Parameters]
      = [param[' '] for param in Parameters]
    m_v = [param['m_v'] for param in Parameters]
    OCR = [param['OCR'] for param in Parameters]
    e_0 = [param['e_0'] for param in Parameters]
    C_e = [param['C_e'] for param in Parameters]
    C_c = [param['C_c'] for param in Parameters]
    C_ae = [param['C_ae'] for param in Parameters]

    eta_end = np.zeros(len(Soil)-1)

    i_zero = Consolidation_models.find_zero_index_arr(t_l)
    if not i_zero == None:
        Soil = np.delete(Soil, i_zero)
        t_l = np.delete(t_l, i_zero)
        y_sat = np.delete(y_sat, i_zero)
```

```
        c_v = np.delete(c_v, i_zero)
          = np.delete( , i_zero)
        m_v = np.delete(m_v, i_zero)
        bou_con = np.delete(bou_con, i_zero)
        Color = np.delete(Color, i_zero)
        OCR = np.delete(OCR, i_zero)
        e_0 = np.delete(e_0, i_zero)
        C_e = np.delete(C_e, i_zero)
        C_c = np.delete(C_c, i_zero)
        C_ae = np.delete(C_ae, i_zero)
        #print(f'Zero value for thickness layer was discovered in {Zone}')

    d_m_l, sig_m_eff = Consolidation_models.sigma_m(h, Soil, t_lay, t_l, y_sat)
    d_l = Consolidation_models.sigmas(Zone, h, y_sat, t_l, t_foun, t_prot,␣
↪d_m_l, sig_m_eff)
    q = Consolidation_models.loads(Zone, el_type, y_sat_u, t_l_u, gamma_foun,␣
↪gamma_w, gamma_prot, t_foun, t_prot, q_rel, F_res_3, Bal_con_req)
    sigma_values = Consolidation_models.load_profile(M, t_c, q, time_days,␣
↪start_times)
    superimposed_porepressure_value, de = Consolidation_models.
↪excess_pore_pressure(Zone, t_u_prof, M, N, K, sub_t, t_end, t_c, q,␣
↪start_times, Soil, y_sat, t_l, d_l, c_v,  , m_v, bou_con)
    superimposed_settlement = Consolidation_models.set_prof(Soil, t_l, M, N, q,␣
↪t_c, sub_t, time_days, start_times, c_v,  , m_v, bou_con)
    t_EOP = Consolidation_models.calculate_t_EOP(Zone, Soil, m_v, t_l, q, M, N,␣
↪c_v,  , t_c, start_times, t_end_EOP, bou_con)


    for i in range(len(Soil)-1):
        index_start_time_1 = np.where(time_days == time_days[-1])[0][0]
        eta_end[i] = superimposed_settlement[i, index_start_time_1]/t_l[i+1]
    sig_zp, eta_zp, sig_z, eta_z, t_e, eta_fj, note = Consolidation_models.
↪stress_path(Soil, sig_m_eff, OCR, q, e_0, C_e, C_c, C_ae, t_0, start_times,␣
↪t_c, t_EOP, eta_end)
    Cre = Consolidation_models.Creep(Soil, t, t_0, np.delete(t_EOP,0,1), np.
↪delete(start_times,0), alfa, C_ae, e_0, t_e, t_l, note, eta_end, time_days)

    q_tot = np.sum(q)
    #print(f'q_tot_{Zone}_{el_type}={q_tot}')
    #print(f'q_{Zone}_{el_type}={q}')
    #print(f't_EOP_{Zone}_{el_type} = {t_EOP}')
    #print(f'sig_z_{Zone}_{el_type} = {sig_z}')
    #print(f'eta_z_{Zone}_{el_type} = {eta_z}')
    #print(f'sig_zp_{Zone}_{el_type} = {sig_zp}')
    #print(f'eta_zp_{Zone}_{el_type} = {eta_zp}')
    #print(f't_e_{Zone}_{el_type}={t_e}')
```

```
    #print(f'eta_fj_{Zone}_{el_type} = {eta_fj}')
    #print(f'eta_end_{Zone}_{el_type} = {eta_end}')

    return superimposed_porepressure_value, de, superimposed_settlement, Cre,␣
↪Color, Soil, t_EOP, q_tot
```

[ ]:
```
def reg_el(t_foun, t_prot):
    h = 8.900                   # [m]
    l = 217                     # [m]
    w = 42.930                  # [m] including the foot of the structure
    w_fo = 1.725                # [m] width of the foot
    h_fo = 1.57                 # [m] height of the foot
    A_reg = l * w               # [m] area of the base plate tunnel

    vol_foun = w*l*t_foun                                                    #␣
↪[m3] volume of the foundation layer
    vol_prot = (w-w_fo)*l*t_prot                                             #␣
↪[m3] volume of the protection layer
    d_tun_tren = t_foun + t_prot + h                                         #␣
↪[m] depth of the tunnel trench
    w_tun_tren_top = d_tun_tren * 3 * 2 + w                                  #␣
↪[m] width of the tunnel trench at the top
    vol_fil = (d_tun_tren**2) * 3 * l + l * w_fo * (h-h_fo)                  #␣
↪[m3] volume of the filling material
    vol_rem =  vol_fil + w * l * d_tun_tren                                  #␣
↪[m3] volume removed to place the tunnel part

    return vol_foun, vol_prot, vol_fil, vol_rem, A_reg
```

[ ]:
```
def spec_el(t_foun, t_prot):
    h = 13.450                  # [m]
    l = 39                      # [m]
    w = 50.055-(2*1.725)        # [m] excluding the protuding parts
    w_pro = 1.725               # [m] width of the foot
    h_pro = 2.1                 # [m] height of the foot
    A_spec = l * w               # [m] area of the base plate tunnel

    vol_foun = w*l*t_foun                                                    ␣
↪# [m3] volume of the foundation layer
    vol_prot = w*l*t_prot                                                    ␣
↪# [m3] volume of the protection layer
    d_tun_tren = t_foun + t_prot + h                                         ␣
↪# [m] depth of the tunnel trench
    w_tun_tren_top = d_tun_tren * 3 * 2 + w                                  ␣
↪# [m] width of the tunnel trench at the top
```

```
    vol_fil = (d_tun_tren**2) * 3 * l - l * w_pro * h_pro                  ␣
↪# [m3] volume of the filling material
    vol_rem =  vol_fil + w * l * d_tun_tren + l * w_pro * h_pro            ␣
↪# [m3] volume removed to place the tunnel part

    return vol_foun, vol_prot, vol_fil, vol_rem, A_spec
```

[ ]:
```
def loads(Zone, el_type, y_sat_u, t_l_u, gamma_foun, gamma_w, gamma_prot,␣
↪t_foun, t_prot, q_rel, F_res_3, Bal_con_req):
    q_unl = Consolidation_models.unloading(y_sat_u, t_l_u)
    q_foun = t_foun * (gamma_foun - gamma_w)
    q_prot = t_prot * (gamma_prot- gamma_w)

    if el_type == 'regular':
        q_ele = (F_res_3 + Bal_con_req) / reg_el(t_foun, t_prot)[4]

    else:
        q_ele = (F_res_3 + Bal_con_req) / spec_el(t_foun, t_prot)[4]

    if (Zone == 'A') or (Zone == 'D') or (Zone == 'E'):
        q = np.array([-q_unl, q_foun, float(q_ele), q_rel], dtype=np.float64)
    elif Zone == 'B':
        q = np.array([-q_unl, q_foun, float(q_ele), q_prot], dtype=np.float64)
    else:
        q = np.array([-q_unl, q_foun, float(q_ele), q_prot, q_rel], dtype=np.
↪float64)

    return q
```

[ ]:
```
def sigmas(Zone, h_ele, y_sat, t_l, t_foun, t_prot, d_m_l, sig_m_eff):

    for i in range(len(t_l)):
        if i == 0:
            d_l = [t_l[i]]
            if (Zone == 'B') or (Zone == 'C'):
                d_l = np.append(d_l,(d_l[-1] + t_prot))
                d_l = np.append(d_l,(d_l[-1] + h_ele))
                d_l = np.append(d_l,(d_l[-1] + t_foun))
            elif (Zone == 'A') or (Zone == 'D') or (Zone == 'E'):
                d_l = np.append(d_l,(d_l[-1] + h_ele))
                d_l = np.append(d_l,(d_l[-1] + t_foun))
        if i > 0:
            d_l = np.append(d_l,(d_l[-1] + t_l[i]))

    y_sat_u = np.insert(y_sat,1,9.81)
    y_sat_u = np.insert(y_sat_u,1,9.81)
    d = np.arange(0,100,1)
```

```
    am = 0
    amount = 0
    sig_tot = []

    if (Zone == 'B') or (Zone =='C'):
        y_sat_u = np.insert(y_sat_u,1,9.81)

    for j in range(len(d_l)):
        for i in range(am,len(d)):
            if d[i] < d_l[j]:
                if i == 0:
                    sig_tot = np.append(sig_tot, 0 + (d[i]-0) * y_sat_u[j])
                    amount += 1
                else:
                    sig_tot = np.append(sig_tot, sig_tot[-1] + (d[i]-d[i-1]) *␣
↪y_sat_u[j])
                    amount += 1
            else:
                am = amount

    return d_l
```

[ ]:
```
@jit(nopython = True)
def unloading(y_sat, t_l):
    x = 0
    for i in range(len(y_sat)):
        x += y_sat[i] * t_l[i]
    return x
```

[ ]:
```
@jit(nopython = True)
def sigma_m(h_ele, Soil, t_lay, t_l, y_sat):

    sig_m_eff = np.zeros(len(Soil)-1)
    sig_m = np.zeros(len(Soil)-1)
    sig_por = np.zeros(len(Soil)-1)
    d_m_l = np.zeros(len(Soil)-1)

    for i in range(len(Soil)-1):
        if i == 0:
            d_m_l[i] = 0.5 * t_l[i+1] + t_l[i] + t_lay + h_ele
            sig_m[i] = 0.5 * t_l[i+1] * y_sat[i+1] + y_sat[0] * (t_l[i] + t_lay␣
↪+ h_ele)
        else:
            d_m_l[i] = d_m_l[i-1] + 0.5 * t_l[i] + 0.5 * t_l[i+1]
            sig_m[i] = sig_m[i-1] +  0.5 * t_l[i] * y_sat[i] + 0.5 * t_l[i+1] *␣
↪y_sat[i+1]
```

```
    sig_por = d_m_l * 9.81
    sig_m_eff = sig_m - sig_por

    return d_m_l, sig_m_eff
```

[ ]:
```
# Function to calculate omega_k
@jit(nopython = True)
def omega_k(k, T):
    return (2 * k * np.pi) / T

# Functions to calculate A_k and B_k
@jit(nopython = True)
def A_k(k, q, T, t_c, t_b):
    omega_k_val = omega_k(k, T)
    return ((q * T) / (2 * t_c * k**2 * np.pi**2)) * (np.cos(omega_k_val * t_c)␣
 ↪+ omega_k_val * t_c * np.sin(omega_k_val * t_b) - 1)

@jit(nopython = True)
def B_k(k, q, T, t_c, t_b):
    omega_k_val = omega_k(k, T)
    return ((q * T) / (2 * t_c * k**2 * np.pi**2)) * (np.sin(omega_k_val * t_c)␣
 ↪- omega_k_val * t_c * np.cos(omega_k_val * t_b))

# Function to calculate sigma(t)
@jit(nopython = True)
def sigma(t, M, q, T, t_c, t_b):
    sigma_t = (q / T) * (t_b - t_c / 2)
    for k in range(1, M + 1):
        A_k_val = A_k(k, q, T, t_c, t_b)
        B_k_val = B_k(k, q, T, t_c, t_b)
        omega_k_val = omega_k(k, T)
        sigma_t += A_k_val * np.cos(omega_k_val * t) + B_k_val * np.
 ↪sin(omega_k_val * t)
    return sigma_t
```

[ ]:
```
# Function to calculate u_omega(z,t)
@jit(nopython = True)
def u_omega(z, t, M, N, q, T, t_c, t_b, nu, c_v, m_v, H):
    u_omega_t = 0
    for k in range(1, M + 1):
        omega_k_val = omega_k(k, T)
        A_k_val = A_k(k, q, T, t_c, t_b)
        B_k_val = B_k(k, q, T, t_c, t_b)
        theta = (nu * c_v) / (omega_k_val * H**2)
        for j in range(1, N + 1):
            T_v = (c_v * t) / H**2
            xi_j = (2 * j - 1) * np.pi / 2
```

```
            Y_j = ((A_k_val + B_k_val * theta * xi_j**2) *
                   (np.cos(omega_k_val * t) - np.exp(-nu * xi_j**2 * T_v)) -
                   (A_k_val * theta * xi_j**2 - B_k_val) *
                   np.sin(omega_k_val * t))
            u_omega_t += (((-1)**j) / (xi_j + theta**2 * xi_j**5)) * Y_j * np.
 ↪cos((xi_j * z) / H)
    return -2 * nu * u_omega_t
```

[ ]:
```
# Function to calculate s_omega(t)
@jit(nopython = True)
def s_omega(t, M, N, q, T, T_v, t_c, t_b, nu, c_v, m_v, H):
    s_omega_t = 0
    for k in range(1, M + 1):
        omega_k_val = omega_k(k, T)
        A_k_val = A_k(k, q, T, t_c, t_b)
        B_k_val = B_k(k, q, T, t_c, t_b)
        theta = (nu * c_v) / (omega_k_val * H**2)
        sum_Y_j = 0
        for j in range(1, N + 1):
            xi_j = (2*j - 1) * np.pi / 2
            Y_j = ((A_k_val + B_k_val*theta*xi_j**2) *
                   (np.cos(omega_k_val*t) - np.exp(-nu*xi_j**2*T_v)) -
                   (A_k_val*theta*xi_j**2 - B_k_val) *
                   np.sin(omega_k_val*t))
            sum_Y_j += Y_j / (xi_j**2 + theta**2*xi_j**6)
        s_omega_t += m_v * H * (A_k_val * np.cos(omega_k_val * t) + B_k_val *␣
 ↪np.sin(omega_k_val * t) - 2 * nu * sum_Y_j)
    return s_omega_t
```

[ ]:
```
@jit(nopython = True)
def load_profile(M, t_c, q, time_days, start_times):

    if isinstance(time_days, float):
        superimposed_sigma_values = 0
        t_b = 1.2 * time_days
        T = 1.1 * t_b
        for j in range(len(t_c)):
            if time_days >= start_times[j]:
                superimposed_sigma_values += sigma(time_days - start_times[j],␣
 ↪M, q[j], T, t_c[j], t_b)

    else:
        t_b = 1.2 * time_days[-1]
        T = 1.1 * t_b

        # Calculate superimposed sigma(t) values
        superimposed_sigma_values = np.zeros_like(time_days)
```

```
        for i in range(len(time_days)):
            for j in range(len(t_c)):
                if time_days[i] >= start_times[j]:
                    superimposed_sigma_values[i] += sigma(time_days[i] -␣
↪start_times[j], M, q[j], T, t_c[j], t_b)

    return superimposed_sigma_values
```

[ ]:
```
@jit(nopython = True)
def u_prof(H, M, N, q, t_c, t, start_times, c_v, nu, m_v, bou_con, t_end,␣
↪norm_load, K, sub_t):
    z = np.linspace(0, H, K)
    superimposed_u_values_depth = np.zeros_like(z)

    if bou_con == "open":
        z = np.linspace(0, H / 2, int(K / 2))
        superimposed_u_values_depth = np.zeros_like(z)

    depths = z
    t_b = 1.2 * t_end
    T = 1.1 * t_b
    start = np.zeros_like(z)

    for j in range(len(t_c)):
        if t >= start_times[j]:
            superimposed_u_values_depth = (start + np.array([u_omega(z, t, M,␣
↪N, q[j], T, t_c[j], t_b, nu, c_v, m_v, H) for z in depths])) / norm_load
            break
        else:
            start += np.array([u_omega(z, start_times[j], M, N, q[j], T,␣
↪t_c[j], t_b, nu, c_v, m_v, H) for z in depths])

    if bou_con == "upper":
        superimposed_u_values_depth = superimposed_u_values_depth[::-1]

    elif bou_con == "lower":
        superimposed_u_values_depth = superimposed_u_values_depth

    elif bou_con == "open":
        superimposed_u_values_depth = np.append(superimposed_u_values_depth[::
↪-1], superimposed_u_values_depth)

    else:
        superimposed_u_values_depth = np.zeros_like(z)

    return superimposed_u_values_depth
```

```
[ ]: @jit(nopython = True)
def excess_pore_pressure(Zone, t, M, N, K, sub_t, t_end, t_c, q_reg,␣
 ↪start_times, Soil, y_sat, t_l, d_l, c_v,  , m_v, bou_con):
    num_times = len(t)
    num_layers = len(Soil) - 1
    total_superimposed_porepressure_value = np.zeros((num_times, num_layers *␣
 ↪K))
    total_de = np.zeros((num_times, num_layers * K))

    if Zone == 'A' or Zone == 'D' or Zone == 'E':
        for idx, times in enumerate(t):
            norm_load = load_profile(M, t_c, q_reg, times, start_times) -␣
 ↪y_sat[0] * t_l[0]
            superimposed_porepressure_value = u_prof(t_l[1], M, N, q_reg, t_c,␣
 ↪times, start_times, c_v[1],  [1], m_v[1], bou_con[0], t_end, norm_load, K,␣
 ↪sub_t)
            de = np.linspace(d_l[2], d_l[3], K)

            for i in range(1, num_layers):
                de = np.concatenate((de, np.linspace(d_l[i + 2], d_l[i + 3],␣
 ↪K)))
                superimposed_porepressure_value = np.
 ↪concatenate((superimposed_porepressure_value,␣
 ↪superimposed_porepressure_value[-1] + u_prof(t_l[i + 1], M, N, q_reg, t_c,␣
 ↪times, start_times, c_v[i + 1],  [i + 1], m_v[i + 1], bou_con[i], t_end,␣
 ↪norm_load, K, sub_t)))

            total_superimposed_porepressure_value[idx, :] =␣
 ↪superimposed_porepressure_value
            total_de[idx, :] = de

    else:
        for idx, times in enumerate(t):
            norm_load = load_profile(M, t_c, q_reg, times, start_times) -␣
 ↪y_sat[0] * t_l[0]
            superimposed_porepressure_value = u_prof(t_l[1], M, N, q_reg, t_c,␣
 ↪times, start_times, c_v[1],  [1], m_v[1], bou_con[0], t_end, norm_load, K,␣
 ↪sub_t)
            de = np.linspace(d_l[3], d_l[4], K)

            for i in range(1, num_layers):
                de = np.concatenate((de, np.linspace(d_l[i + 3], d_l[i + 4],␣
 ↪K)))
```

```
                superimposed_porepressure_value = np.
↪concatenate((superimposed_porepressure_value,␣
↪superimposed_porepressure_value[-1] + u_prof(t_l[i + 1], M, N, q_reg, t_c,␣
↪times, start_times, c_v[i + 1],  [i + 1], m_v[i + 1], bou_con[i], t_end,␣
↪norm_load, K, sub_t)))

            total_superimposed_porepressure_value[idx, :] =␣
↪superimposed_porepressure_value
            total_de[idx, :] = de

    return total_superimposed_porepressure_value, total_de
```

[ ]:
```
@jit(nopython = True)
def set_prof(Soil, H, M, N, q, t_c, sub_t, time_days, start_times, c_v, nu,␣
↪m_v, bou_con):
    t_b = 1.2 * time_days[-1]
    T = 1.1 * t_b

    # eta_end = np.zeros(len(Soil)-1)
    times = np.zeros((len(t_c), len(time_days)))

    for j in range(len(t_c)):
        for i in range(len(time_days)):
            if time_days[i] == start_times[j]:
                times[j,:] = np.hstack((np.zeros(i),time_days[i:
↪len(time_days)]-time_days[i]))

    superimposed_settlement_value = np.zeros((len(Soil)-1, len(time_days)))
    superimposed_settlement_values = np.zeros((len(t_c), len(time_days)))

    for k in range(len(Soil) - 1):
        if bou_con[k] == "upper" or bou_con[k] == "lower":
            L = H[k + 1]
        else:
            L = H[k + 1] / 2

        for j in range(len(t_c)):
            for m in range(len(time_days)):
                if time_days[m] == start_times[j]:
                    i_loc =  m
            for i in range(i_loc,len(time_days)):
                superimposed_settlement_values[j, i] = (m_v[k + 1] * L * (q[j] /
↪ T) * (t_b - t_c[j] / 2)) + s_omega(times[j, i], M, N, q[j], T, ((times[j,␣
↪i]) * c_v[k + 1]) / L**2, t_c[j], t_b, nu[k + 1], c_v[k + 1], m_v[k + 1], L)
```

```
        superimposed_settlement_value[k, :] = np.
↪sum(superimposed_settlement_values, axis = 0)

        if Soil[k + 1] == 'Meltwater sand' or Soil[k + 1] == 'Postglacial Sand'␣
↪or Soil[k+1] ==  'Chalk':
            superimposed_settlement_value[k, :] = np.zeros((1, len(time_days)))

        tolerance = 10

        # for j in range(len(t_c)):
        #     for i in range(len(times)):
        #         if abs(time_days[i] - (start_times[j] + t_c[j])) < tolerance:
        #             eta_end[k,j] = superimposed_settlement_value[k,i] / H[k+1]
        #             break



    return superimposed_settlement_value
```

[ ]:
```
#Does the time needs to start at zero? How does a previous timestep laod␣
↪influence the current timestep load?
@jit(nopython = True)
def calculate_t_EOP(Zone, Soil, m_v, H, q, M, N, c_v, nu, t_c, start_times,␣
↪t_end, bou_con):
    t_b = 1.2 * t_end
    T = 1.1 * t_b

    #t_EOP = np.zeros(len(t_c))
    #s_EOP = np.zeros(len(t_c))

    l = 0

    t_EOP = np.zeros(((len(Soil)-1),len(t_c)))
    s_EOP = np.zeros(((len(Soil)-1),len(t_c)))

    for k in range(len(Soil)-1):
        if bou_con[k] == "open":
            L = (H[k+1]/2)
        else:
            L = H[k+1]

        if Soil[k+1] == 'Meltwater sand' or Soil[k+1] == 'Postglacial Sand' or␣
↪Soil[k+1] == 'Chalk':
            t_EOP[k,:] = t_EOP[k,:]
            s_EOP[k,:] = s_EOP[k,:]

        else:
```

```
            start = 0
            for j in range(len(t_c)):
                l += q[j]
                days = 0
                settlement_values = np.zeros(t_end)
                found = False

                for i in range(1, t_end):
                    settlement_values[i] = start + m_v[k+1] * L  * (l / T) *␣
 ↪(t_b - t_c[j] / 2) + s_omega(days, M, N, l, T, ((days * c_v[k+1]) / L**2),␣
 ↪t_c[j], t_b, nu[k+1], c_v[k+1], m_v[k+1], L)
                    if np.isclose(settlement_values[i], settlement_values[i -␣
 ↪1], atol=1e-6):
                        t_EOP[k,j] = days + start_times[j] + t_c[j]
                        s_EOP[k,j] = settlement_values[i]
                        found = True
                        break

                    days += 1
                    if found:
                        start = settlement_values[i]
                        break  # Break the for loop if found
                if not found:
                    raise ValueError("No close values found within the␣
 ↪specified range. Increase t_end or decrease substeps in settlement_profile")


    return t_EOP
```

[ ]:
```
@jit(nopython = True)
def Creep(Soil, t, t_0, t_EOP, start_times, alfa, C_ae, e_0, t_e, t_l, note,␣
 ↪eta_end, time_days):
    Cre = np.zeros((len(C_ae) - 1, len(t)))
    w = np.zeros(len(t))

    for i in range(len(C_ae) - 1):
        start_times_sub = np.append(start_times, t_EOP[i,-1])
        Cre_sub = np.zeros(len(t))
        if Soil[i+1] == 'Meltwater sand' or Soil[i+1] == 'Postglacial Sand' or␣
 ↪Soil[i+1] == 'Chalk':
            Cre[i,:] = np.zeros(len(t))
            continue
        else:
            for k in range(len(t)):
                for j in range(len(start_times)):
                    start_times_sub = np.append(start_times, time_days[-1])
```

```
                    if t[k] >= start_times_sub[j] and t[k] <= start_times_sub[j␣
↪+ 1]:
                        time = start_times_sub[j+1] - start_times_sub[j]
                        if note[i,j] == 1:
                            Cre_sub[k] = (alfa * (C_ae[i + 1] / (1 + e_0[i +␣
↪1])) * math.log10(time/t_0)) * t_l[i + 1]
                        else:
                            Cre_sub[k] = (alfa * (C_ae[i + 1] / (1 + e_0[i +␣
↪1])) * math.log10((time + t_e[i,j])/(t_0 + t_e[i,j]))) * t_l[i + 1]

                    elif t[k] >= time_days[-1] and t[k] <= t_EOP[i,-1]:
                        time = start_times_sub[j+1] - start_times_sub[j]
                        if note[i,j] == 1:
                            Cre_sub[k] = eta_end[i]* t_l[i + 1] + (alfa *␣
↪(C_ae[i + 1] / (1 + e_0[i + 1])) * math.log10(time/t_0)) * t_l[i + 1]
                        else:
                            Cre_sub[k] = eta_end[i]* t_l[i + 1] + (alfa *␣
↪(C_ae[i + 1] / (1 + e_0[i + 1])) * math.log10((time + t_e[i,j])/(t_0 +␣
↪t_e[i,j]))) * t_l[i + 1]

                    elif t[k] >= t_EOP[i,-1]:
                        time = t_EOP[i,-1]
                        if note[i,j] == 1:
                            Cre_sub[k] = eta_end[i]* t_l[i + 1] + ((1 - alfa) *␣
↪(C_ae[i + 1] / (1 + e_0[i + 1])) * math.log10((t[k]) / (t_EOP[i,-1]))) *␣
↪t_l[i + 1] + (alfa * (C_ae[i + 1] / (1 + e_0[i + 1])) * math.log10(time /␣
↪t_0 )) * t_l[i + 1]
                        else:
                            Cre_sub[k] = eta_end[i]* t_l[i + 1] + ((1 - alfa) *␣
↪(C_ae[i + 1] / (1 + e_0[i + 1])) * math.log10((t[k]) / (t_EOP[i,-1]))) *␣
↪t_l[i + 1]  + (alfa * (C_ae[i + 1] / (1 + e_0[i + 1])) * math.log10((time +␣
↪t_e[i, j]) / (t_0 + t_e[i,j]))) * t_l[i + 1]
                    else:
                        continue
            Cre[i,:] = Cre_sub
        return Cre
```

[ ]:
```
def stress_path(Soil, sig_m_eff, OCR, q, e_0, C_e, C_c, C_ae, t_0, start_times,␣
↪t_c, t_EOP, eta_end):

    Soil = np.delete(Soil,0)
    OCR = np.delete(OCR, 0)
    e_0 = np.delete(e_0, 0)
    C_e = np.delete(C_e, 0)
    C_c = np.delete(C_c, 0)
    C_ae = np.delete(C_ae, 0)
```

```
    t_EOP = t_EOP[:, 1:]
    q = np.delete(q,0)
    start_times = np.delete(start_times,0)

    sig_zp_0 = np.zeros(len(sig_m_eff))
    eta_zp_0 = np.zeros(len(sig_m_eff))
    sig_zp = np.zeros((len(sig_m_eff),len(q)))
    eta_zp = np.zeros((len(sig_m_eff),len(q)))
    sig_z = np.zeros((len(sig_m_eff),len(q)))
    eta_z = np.zeros((len(sig_m_eff),len(q)))
    t_e = np.zeros((len(sig_m_eff),len(q)))
    eta_fj = np.zeros((len(sig_m_eff),len(q)))

    sig_z_0 = sig_m_eff
    eta_z_0 = np.zeros(len(sig_m_eff))
    note = np.zeros((len(sig_m_eff),len(q)))

    for i in range(len(sig_m_eff)):

        if Soil[i] == 'Meltwater sand' or Soil[i] == 'Postglacial Sand' or␣
↪Soil[i] == 'Chalk':
            continue

        else:
            sig_zp_0[i] = sig_m_eff[i] * OCR[i]
            eta_zp_0[i] = (C_e[i]/(1 + e_0[i]))*math.log10(OCR[i])

            for j in range(0,len(q)):
                if j == 0:
                    sig_z[i,j]  = sig_m_eff[i] + q[j]

                    if sig_z[i,j] >= sig_zp_0[i]:
                        eta_z[i,j] = eta_end[i] + (C_e[i]/(1 + e_0[i]))*math.
↪log10(sig_zp_0[i]/sig_m_eff[i]) + (C_c[i]/(1 + e_0[i]))*math.
↪log10(sig_z[i,j]/sig_zp_0[i])
                        sig_zp[i,j] = sig_zp_0[i] +␣
↪(10**((eta_z[i,j]-eta_zp_0[i])*((1+e_0[i])/
↪(C_c[i]-C_e[i]))))*((sig_z[i,j])**(-(C_e[i])/
↪(C_c[i]-C_e[i])))*(sig_zp_0[i])**((C_c[i])/(C_c[i]-C_e[i]))
                        eta_zp[i,j] = eta_z_0[i] + (C_e[i]/(1 + e_0[i]))*math.
↪log10(sig_zp[i,j]/sig_z[i,j])
                        t_e[i,j]    = (t_0*(10**((eta_z[i,j] -␣
↪eta_zp_0[i])*((1+e_0[i])/C_ae[i])))*(sig_z[i,j]/sig_zp_0[i])**(-C_c[i]/
↪C_ae[i])) - t_0
                        t           = start_times[j+1] - start_times[j]
                        eta_fj[i,j] = (C_ae[i]/(1+e_0[i]))*math.log10(t/t_0)
                        note [i,j] = 1
```

```
                    else:
                        eta_z[i,j] = eta_end[i] + (C_e[i]/(1 + e_0[i]))*math.
↪log10(sig_z[i,j]/sig_m_eff[i])
                        sig_zp[i,j] = sig_zp_0[i] +␣
↪(10**((eta_z[i,j]-eta_zp_0[i])*((1+e_0[i])/
↪(C_c[i]-C_e[i]))))*((sig_z[i,j])**(-(C_e[i])/
↪(C_c[i]-C_e[i])))*(sig_zp_0[i])**((C_c[i])/(C_c[i]-C_e[i]))
                        eta_zp[i,j] = eta_z_0[i] + (C_e[i]/(1 + e_0[i]))*math.
↪log10(sig_zp[i,j]/sig_z[i,j])
                        t_e[i,j]    = (t_0*(10**((eta_z[i,j] -␣
↪eta_zp_0[i])*((1+e_0[i])/C_ae[i])))*(sig_z[i,j]/sig_zp_0[i])**(-C_c[i]/
↪C_ae[i])) - t_0
                        t           = start_times[j+1] - start_times[j]
                        eta_fj[i,j] = (C_ae[i]/(1+e_0[i]))*math.log10((t +␣
↪t_e[i,j])/(t_0+ t_e[i,j]))

                elif j == (len(q)-1):
                    sig_z[i,j]  = sig_z[i,j-1] + q[j]

                    if sig_z[i,j]   >= sig_zp[i,j-1]:
                        eta_z[i,j]  = eta_z[i,j-1] +(C_e[i]/(1 + e_0[i]))*math.
↪log10(sig_zp[i,j-1]/sig_z[i,j-1]) + (C_c[i]/(1 + e_0[i]))*math.
↪log10(sig_z[i,j]/sig_zp[i,j-1]) #+ eta_fj[i,j-1]
                        sig_zp[i,j] = sig_zp[i,j-1] +␣
↪(10**((eta_z[i,j]-eta_zp[i,j-1])*((1+e_0[i])/
↪(C_c[i]-C_e[i]))))*((sig_z[i,j])**(-(C_e[i])/
↪(C_c[i]-C_e[i])))*(sig_zp[i,j-1])**((C_c[i])/(C_c[i]-C_e[i]))
                        eta_zp[i,j] = eta_z[i,j-1] + (C_e[i]/(1 + e_0[i]))*math.
↪log10(sig_zp[i,j]/sig_z[i,j])
                        t_e[i,j]    = (t_0*(10**((eta_z[i,j] -␣
↪eta_zp_0[i])*((1+e_0[i])/C_ae[i])))*(sig_z[i,j]/sig_zp_0[i])**(-C_c[i]/
↪C_ae[i])) - t_0
                        t           = t_EOP[i,j] - start_times[j]
                        eta_fj[i,j] = (C_ae[i]/(1+e_0[i]))*math.log10(t/t_0)
                        note [i,j] = 1
                    else:
                        eta_z[i,j]  = eta_z[i,j-1] + (C_e[i]/(1 + e_0[i]))*math.
↪log10(sig_z[i,j]/sig_z[i,j-1]) #+ eta_fj[i,j-1]
                        sig_zp[i,j] = sig_zp[i,j-1] +␣
↪(10**((eta_z[i,j]-eta_zp[i,j-1])*((1+e_0[i])/
↪(C_c[i]-C_e[i]))))*((sig_z[i,j])**(-(C_e[i])/
↪(C_c[i]-C_e[i])))*(sig_zp[i,j-1])**((C_c[i])/(C_c[i]-C_e[i]))
                        eta_zp[i,j] = eta_z[i,j-1] + (C_e[i]/(1 + e_0[i]))*math.
↪log10(sig_zp[i,j]/sig_z[i,j])
```

```
                        t_e[i,j]    = (t_0*(10**((eta_z[i,j] -␣
↪eta_zp_0[i])*((1+e_0[i])/C_ae[i])))*(sig_z[i,j]/sig_zp_0[i])**(-C_c[i]/
↪C_ae[i])) - t_0
                        t           = t_EOP[i,j] - start_times[j]
                        eta_fj[i,j] = (C_ae[i]/(1+e_0[i]))*math.log10((t +␣
↪t_e[i,j])/(t_0+ t_e[i,j]))


                else:
                    sig_z[i,j]  = sig_z[i,j-1] + q[j]

                    if sig_z[i,j]   >= sig_zp[i,j-1]:
                        eta_z[i,j]  = eta_z[i,j-1] +(C_e[i]/(1 + e_0[i]))*math.
↪log10(sig_zp[i,j-1]/sig_z[i,j-1]) + (C_c[i]/(1 + e_0[i]))*math.
↪log10(sig_z[i,j]/sig_zp[i,j-1]) #+ eta_fj[i,j-1]
                        sig_zp[i,j] = sig_zp[i,j-1] +␣
↪(10**((eta_z[i,j]-eta_zp[i,j-1])*((1+e_0[i])/
↪(C_c[i]-C_e[i]))))*((sig_z[i,j])**(-(C_e[i])/
↪(C_c[i]-C_e[i])))*(sig_zp[i,j-1])**((C_c[i])/(C_c[i]-C_e[i]))
                        eta_zp[i,j] = eta_z[i,j-1] + (C_e[i]/(1 + e_0[i]))*math.
↪log10(sig_zp[i,j]/sig_z[i,j])
                        t_e[i,j]    = (t_0*(10**((eta_z[i,j] -␣
↪eta_zp_0[i])*((1+e_0[i])/C_ae[i])))*(sig_z[i,j]/sig_zp_0[i])**(-C_c[i]/
↪C_ae[i])) - t_0
                        t           = start_times[j+1] - start_times[j]
                        eta_fj[i,j] = (C_ae[i]/(1+e_0[i]))*math.log10(t/t_0)
                        note [i,j] = 1
                    else:
                        eta_z[i,j]  = eta_z[i,j-1] + (C_e[i]/(1 + e_0[i]))*math.
↪log10(sig_z[i,j]/sig_z[i,j-1]) #+ eta_fj[i,j-1]
                        sig_zp[i,j] = sig_zp[i,j-1] +␣
↪(10**((eta_z[i,j]-eta_zp[i,j-1])*((1+e_0[i])/
↪(C_c[i]-C_e[i]))))*((sig_z[i,j])**(-(C_e[i])/
↪(C_c[i]-C_e[i])))*(sig_zp[i,j-1])**((C_c[i])/(C_c[i]-C_e[i]))
                        eta_zp[i,j] = eta_z[i,j-1] + (C_e[i]/(1 + e_0[i]))*math.
↪log10(sig_zp[i,j]/sig_z[i,j])
                        t_e[i,j]    = (t_0*(10**((eta_z[i,j] -␣
↪eta_zp_0[i])*((1+e_0[i])/C_ae[i])))*(sig_z[i,j]/sig_zp_0[i])**(-C_c[i]/
↪C_ae[i])) - t_0
                        t           = start_times[j+1] - start_times[j]
                        eta_fj[i,j] = (C_ae[i]/(1+e_0[i]))*math.log10((t +␣
↪t_e[i,j])/(t_0+ t_e[i,j]))

    return sig_zp, eta_zp, sig_z, eta_z, t_e, eta_fj, note
```

[ ]:
```
print("Import Settlement_profile done")
```

Appendix I

The Python script for the plotting of results is shown here.

Soil_profiles

May 22, 2025

[1]:
```
# pip install import_ipynb
```

[2]:
```
# pip install numpy==2.1
```

[3]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import os
import math
import matplotlib.pyplot as plt
from numba import jit
import matplotlib as mpl
from scipy.stats import truncnorm
import import_ipynb
import time
import Plotting
import pickle
import Consolidation_models
import matplotlib.lines as mlines
import matplotlib.gridspec as gridspec
from scipy.stats import norm
#import Timoshenko_beam_on_Kerr_foundation
from Regular_element_buoyancy_calculation_Fehmarn import F_res_3_reg,␣
 ↪Bal_con_req_reg
from Regular_element_buoyancy_calculation_Fehmarn import l_buo_csA as L_ele_reg
from Regular_element_buoyancy_calculation_Fehmarn import w_pl as W_ele_reg
from Special_element_buoyancy_calculation_Fehmarn import l_buo_csA as L_ele_spe
from Special_element_buoyancy_calculation_Fehmarn import w_pl as W_ele_spe
from Special_element_buoyancy_calculation_Fehmarn import F_res_3_spec,␣
 ↪Bal_con_req_spec, A_con_csA
```

```
Import Matrix was succesfull
50456.51774415001
The required force of the ballast concrete needs to be: 172487 [kN]
48456.87277274989
14782.302785250009
Import Settlement_profile done
Import Plotting was succesfull
```

[4]:
```
df = pd.read_excel('Soil_parameters.xlsx', engine='openpyxl')
name =     df.iloc[:,0]   #[-]
col =      df.iloc[:,1]   #[-]
OCR =      df.iloc[:,2]   #[-]
y_sat =    df.iloc[:,3]   #[kN/m3]
  =       df.iloc[:,4]   #[-] fluid compressibility
  ↪                          #On the contrary, if pore-fluid compressibility
  ↪is ignored,   = 1.0.
E_oed =    df.iloc[:,5]   #[kPa]
m_v =      df.iloc[:,6]   #Volumetric compressibility [kPa^-1]
e_0 =      df.iloc[:,7]   #[-]
C_e =      df.iloc[:,8]   #[-]
C_c =      df.iloc[:,9]   #[-]
C_ae =     df.iloc[:,10]  #[-]
c_v =      df.iloc[:,11]  #[m2/day] Coefficient of consolidation

e_0 = [float(x) for x in e_0]
C_ae = [float(x) for x in C_ae]
C_c = [float(x) for x in C_c]

#Soil = [name, color, OCR, y_sat,  , m_v, e_0, C_e, C_c, C_ae, c_v]
Soil_0 = df.iloc[0]
Soil_1 = df.iloc[1]
Soil_2 = df.iloc[2]
Soil_3 = df.iloc[3]
Soil_4 = df.iloc[4]
Soil_5 = df.iloc[5]
Soil_6 = df.iloc[6]
Soil_7 = df.iloc[7]

#Height elements
h_reg = 8.900
h_spe = 13.455

# Unit weight
gamma_foun = 21
  ↪            #[kN/m3] Foundation (from Fu and Song 2021) #Thickness changes
  ↪will only add a load and have no significant effect on total creep/
  ↪consolidation
gamma_w = 9.81
  ↪            #[kN/m3] Water
gamma_bf = 18
  ↪            #[kN/m3] Backfill (from Fu and Song 2021)
gamma_ele = 9.81
  ↪            #[kN/m3] Element Same weight as water in the beginning
```

```
gamma_prot = 26             #[kN/m3] Protection layer (protection layer needs a researched value)

#Thicknesses
t_foun = 1                  #[m] thickness of foundation layer (Monte Carlo needed in the end)
t_prot = 1.5                #[m] thickness of protection layer (Can be changed afterwards to check for influence)
t_fill = 1                  #[m] thickness of filling material on top of protection layer/ tunnel element

# Parameters for signal calculation
M = 600                     #Amount of time fourier series is added to total value, make this value higher if the graph produced is not smooth enough
N = 50
K = 30

#Definition of times used in calculation
t_0 = 1                     #[days] reference days for creep calculation
t_end = 2190                #[days]
sub_t = 10
t_u_prof = [0.5*t_end, 0.75*t_end, 0.99*t_end]  #[days] visualised u_prof for these times further down
time_days = np.arange(0, t_end+sub_t, sub_t)    #[days] time for the total project to take place and up untill what age we want to now the secondary settlement

t_c_un =  3*365             #[days] 1.Unloading/trenching of soil
t_c_foun = 50               #[days] 2.Deposition of foundation layer
t_c_tun = 1                 #[days] 3.Installation of tunnel elements
t_c_prot_d = 50             #[days] 4.Placement of protection layer
t_c_rel_d = 50              #[days] 5.Reloading soil with soil that was removed
```

```
#6.Back siltation from (Wang et al., 2023b) and tidal load from (Shoa and Li,␣
 ↪2003)
#Cyclic degradation is disregarded in this research

t_c = [t_c_un, t_c_foun, t_c_tun]                                              ␣
 ↪            #[days] time of the construction period of that loading stage
t_c_rel = [t_c_un, t_c_foun, t_c_tun, t_c_rel_d]                               ␣
 ↪            #[days] time of the construction period of that loading stage␣
 ↪including the reloading
t_c_prot = [t_c_un, t_c_foun, t_c_tun, t_c_prot_d]                             ␣
 ↪            #[days] time of the construction period of that loading stage␣
 ↪including the protection layer
t_c_rel_prot = [t_c_un, t_c_foun, t_c_tun, t_c_prot_d, t_c_rel_d]              ␣
 ↪            #[days] time of the construction period of that loading stage␣
 ↪including the protection layer and reloading

start_times = [0, 1150, 1300]                                                  ␣
 ↪            #[days]
start_times_rel = [0, 1150, 1300, 1600]                                        ␣
 ↪            #[days]
start_times_prot = [0, 1150, 1300, 1450]                                       ␣
 ↪            #[days]
start_times_rel_prot = [0, 1150, 1300, 1450, 1600]                             ␣
 ↪            #[days]

# Creep parameters
t_end_EOP = 50000
alfa = 0.5
t = np.arange(0,100*365,10)
times = np.array([5, 10, 20, 25, 50, 100])*365

closest_locations = [np.abs(t - value).argmin() for value in times]

#Monte Carlo parameters
std_dev = 2 # Standard deviation
num_simulations = 500  # Number of Monte Carlo simulations

#TK-model space dimensions
n_ele = 2                                                                      ␣
 ↪            # [-]
load_style = "Uniform"                                                         ␣
 ↪            # [-]  (choose between Concentrated and Uniform)
a_u = 0
b_u = 1
dx = 0.1
```

```
#TK-model parameters of concrete beam
kappa = 0.833                                                                  ␣
 ↪           # [-]
E_b = 2.9*10**7                                                                ␣
 ↪           # [kPa]
v_b = 0.2                                                                      ␣
 ↪           # [-]
```

1 Zones

[5]:
```
#Zone A
Parameters_A = [Soil_0, Soil_5]
t_l_A =       [9.625, 80.475]
t_l_A_spe =   [9.625, 75.92]
t_l_A_u =     [1.563, 11.463]
t_l_A_u_spe = [1.563, 16.018]
y_sat_A_u =   [y_sat[7]-gamma_w, y_sat[4]-gamma_w]
q_rel_A =     1.563 * (y_sat[7]-gamma_w) + 1.563 * (y_sat[4]-gamma_w)
bou_con_A =   ["upper"]
d_sc_A =      np.sum(np.delete(t_l_A, [0]))
d_sc_A_spe =  np.sum(np.delete(t_l_A_spe, [0]))

# Zone B
Parameters_B = [Soil_0, Soil_1, Soil_2, Soil_3, Soil_4]
t_l_B =       [34.625, 3.6625, 14.0625, 6.5, 32.25]
t_l_B_spe =   [34.625, 0.5, 13.17, 6.5, 32.25]
t_l_B_u =     [11.4]
t_l_B_u_spe = [15.955]
y_sat_B_u =   [y_sat[1]-gamma_w]
q_rel_B =     0
bou_con_B =   ["upper", "open", "closed", "upper"]
d_sc_B =      np.sum(np.delete(t_l_B, [0]))
d_sc_B_spe =  np.sum(np.delete(t_l_B_spe, [0]))

# Zone C
Parameters_C = [Soil_0, Soil_2, Soil_4, Soil_6]
t_l_C =      [34.375, 0.625, 12.5, 43.6]
t_l_C_u =    [14.525]
y_sat_C_u =  [y_sat[1]-gamma_w]
q_rel_C =    3.125 * (y_sat[1]-gamma_w)
bou_con_C =  ["upper", "upper", "closed"]
d_sc_C =     np.sum(np.delete(t_l_C, [0]))

# Zone D
Parameters_D = [Soil_0, Soil_2, Soil_4, Soil_5]
t_l_D =       [21.875, 27.125, 28.0625, 14.0375]
```

```
t_l_D_spe =   [21.875, 22.57, 28.0625, 14.0375]
t_l_D_u =     [6.25, 9.9]
t_l_D_u_spe = [6.25, 14.455]
y_sat_D_u =   [y_sat[7]-gamma_w, y_sat[2]-gamma_w]
q_rel_D =      6.25 * (y_sat[7]-gamma_w)
bou_con_D =   ["upper", "upper", "upper"]
d_sc_D =      np.sum(np.delete(t_l_D, [0]))
d_sc_D_spe =  np.sum(np.delete(t_l_D_spe, [0]))

# Zone E
Parameters_E = [Soil_0, Soil_3, Soil_2, Soil_4, Soil_5]
t_l_E =       [12.5, 2.125, 18.75, 21.875, 35.85]
t_l_E_spe =   [12.5, 0.5, 16.32, 21.875, 35.85]
t_l_E_u =     [15.4, 1.0]
t_l_E_u_spe = [15.15, 5.555]
y_sat_E_u =   [y_sat[2]-gamma_w, y_sat[3]-gamma_w]
q_rel_E =     6.25 * (y_sat[2]-gamma_w)
bou_con_E =   ["closed","upper","upper","upper"]
d_sc_E =      np.sum(np.delete(t_l_E, [0]))
d_sc_E_spe =  np.sum(np.delete(t_l_E_spe, [0]))

Zone_lay = ['A', 'B', 'C', 'D', 'E']
Zone_lay_spe = ['A', 'B', 'D','E']
```

[6]:
```
# start = time.perf_counter()
# Zone_A    = Consolidation_models.final_config('A', "Uniform",  Parameters_A,␣
 ↪'regular', h_reg, t_l_A, d_sc_A, q_rel_A, F_res_3_reg, Bal_con_req_reg,␣
 ↪y_sat_A_u, t_l_A_u, gamma_foun, gamma_w, gamma_prot, t_foun, t_foun, t_prot,␣
 ↪t_u_prof, bou_con_A, M, N, K, t_c_rel, time_days, start_times_rel,␣
 ↪t_end_EOP, sub_t, t_end, t_0, t, alfa, std_dev, num_simulations, kappa, E_b,␣
 ↪v_b, W_ele_reg, L_ele_reg, a_u, b_u, dx, times)
# Zone_A_spe  = Consolidation_models.final_config('A', "Uniform", ␣
 ↪Parameters_A, 'special', h_spe, t_l_A_spe, d_sc_A_spe, q_rel_A,␣
 ↪F_res_3_spec, Bal_con_req_spec, y_sat_A_u, t_l_A_u_spe, gamma_foun, gamma_w,␣
 ↪gamma_prot, t_foun, t_foun, t_prot, t_u_prof, bou_con_A, M, N, K, t_c_rel,␣
 ↪time_days, start_times_rel, t_end_EOP, sub_t, t_end, t_0, t, alfa, std_dev,␣
 ↪num_simulations, kappa, E_b, v_b, W_ele_spe, L_ele_spe, a_u, b_u, dx, times)
# end = time.perf_counter()

# print(f'Time it takes for Zone A- calculation: {(end-start):.2f} sec or␣
 ↪{(end-start)/60:.2f} minutes')
```

[7]:
```
# start = time.perf_counter()
```

```
# Zone_B = Consolidation_models.final_config('B', "Uniform",  Parameters_B,␣
 ↪'regular', h_reg, t_l_B, d_sc_B, q_rel_B, F_res_3_reg, Bal_con_req_reg,␣
 ↪y_sat_B_u, t_l_B_u, gamma_foun, gamma_w, gamma_prot, t_foun+t_prot, t_foun,␣
 ↪t_prot, t_u_prof, bou_con_B, M, N, K, t_c_prot, time_days, start_times_prot,␣
 ↪t_end_EOP, sub_t, t_end, t_0, t, alfa, std_dev, num_simulations, kappa, E_b,␣
 ↪v_b, W_ele_reg, L_ele_reg, a_u, b_u, dx, times)
# Zone_B_spe = Consolidation_models.final_config('B', "Uniform",  Parameters_B,␣
 ↪'special', h_spe, t_l_B_spe, d_sc_B_spe, q_rel_B, F_res_3_spec,␣
 ↪Bal_con_req_spec, y_sat_B_u, t_l_B_u_spe, gamma_foun, gamma_w, gamma_prot,␣
 ↪t_foun+t_prot, t_foun, t_prot, t_u_prof, bou_con_B, M, N, K, t_c_prot,␣
 ↪time_days, start_times_prot, t_end_EOP, sub_t, t_end, t_0, t, alfa, std_dev,␣
 ↪num_simulations, kappa, E_b, v_b, W_ele_spe, L_ele_spe, a_u, b_u, dx, times)
# end = time.perf_counter()

# print(f'Time it takes for Zone B- calculation: {(end-start):.2f} sec or␣
 ↪{(end-start)/60:.2f} minutes')
```

[8]:
```
# start = time.perf_counter()
# Zone_C = Consolidation_models.final_config('C', "Uniform",  Parameters_C,␣
 ↪'regular', h_reg, t_l_C, d_sc_C, q_rel_C, F_res_3_reg, Bal_con_req_reg,␣
 ↪y_sat_C_u, t_l_C_u, gamma_foun, gamma_w, gamma_prot, t_foun+t_prot, t_foun,␣
 ↪t_prot, t_u_prof, bou_con_C, M, N, K, t_c_rel_prot, time_days,␣
 ↪start_times_rel_prot, t_end_EOP, sub_t, t_end, t_0, t, alfa, std_dev,␣
 ↪num_simulations, kappa, E_b, v_b, W_ele_reg, L_ele_reg, a_u, b_u, dx, times)
# end = time.perf_counter()

# print(f'Time it takes for Zone C- calculation: {(end-start):.2f} sec or␣
 ↪{(end-start)/60:.2f} minutes')
```

[9]:
```
# start = time.perf_counter()
# Zone_D = Consolidation_models.final_config('D', "Uniform",  Parameters_D,␣
 ↪'regular', h_reg, t_l_D, d_sc_D, q_rel_D, F_res_3_reg, Bal_con_req_reg,␣
 ↪y_sat_D_u, t_l_D_u, gamma_foun, gamma_w, gamma_prot, t_foun, t_foun, t_prot,␣
 ↪t_u_prof, bou_con_D, M, N, K, t_c_rel, time_days, start_times_rel,␣
 ↪t_end_EOP, sub_t, t_end, t_0, t, alfa, std_dev, num_simulations, kappa, E_b,␣
 ↪v_b, W_ele_reg, L_ele_reg, a_u, b_u, dx, times)
# Zone_D_spe  = Consolidation_models.final_config('D', "Uniform", ␣
 ↪Parameters_D, 'special', h_spe, t_l_D_spe, d_sc_D_spe, q_rel_D,␣
 ↪F_res_3_spec, Bal_con_req_spec, y_sat_D_u, t_l_D_u_spe, gamma_foun, gamma_w,␣
 ↪gamma_prot, t_foun, t_foun, t_prot, t_u_prof, bou_con_D, M, N, K, t_c_rel,␣
 ↪time_days, start_times_rel, t_end_EOP, sub_t, t_end, t_0, t, alfa, std_dev,␣
 ↪num_simulations, kappa, E_b, v_b, W_ele_spe, L_ele_spe, a_u, b_u, dx, times)
# end = time.perf_counter()

# print(f'Time it takes for Zone D- calculation: {(end-start):.2f} sec or␣
 ↪{(end-start)/60:.2f} minutes')
```

[10]:
```
# start = time.perf_counter()
# Zone_E = Consolidation_models.final_config('E', "Uniform",  Parameters_E,␣
 ↪'regular', h_reg, t_l_E, d_sc_E, q_rel_E, F_res_3_reg, Bal_con_req_reg,␣
 ↪y_sat_E_u, t_l_E_u, gamma_foun, gamma_w, gamma_prot, t_foun, t_foun, t_prot,␣
 ↪t_u_prof, bou_con_E, M, N, K, t_c_rel, time_days, start_times_rel,␣
 ↪t_end_EOP, sub_t, t_end, t_0, t, alfa, std_dev, num_simulations, kappa, E_b,␣
 ↪v_b, W_ele_reg, L_ele_reg, a_u, b_u, dx, times)
# Zone_E_spe = Consolidation_models.final_config('E', "Uniform",  Parameters_E,␣
 ↪'special', h_spe, t_l_E_spe, d_sc_E_spe, q_rel_E, F_res_3_spec,␣
 ↪Bal_con_req_spec, y_sat_E_u, t_l_E_u_spe, gamma_foun, gamma_w, gamma_prot,␣
 ↪t_foun, t_foun, t_prot, t_u_prof, bou_con_E, M, N, K, t_c_rel, time_days,␣
 ↪start_times_rel, t_end_EOP, sub_t, t_end, t_0, t, alfa, std_dev,␣
 ↪num_simulations, kappa, E_b, v_b, W_ele_spe, L_ele_spe, a_u, b_u, dx, times)
# end = time.perf_counter()

# print(f'Time it takes for Zone E- calculation: {(end-start):.2f} sec or␣
 ↪{(end-start)/60:.2f} minutes')
```

[11]:
```
start = time.perf_counter()
Zone_A, Prim_error_A, Sec_error_A, Tot_lay_error_A, Zone_prim_error_A,␣
 ↪Zone_sec_error_A, Zone_tot_error_A, Ini_set_A, Ini_error_A, Set_A,␣
 ↪Set_error_A, Prim_set_A, Sec_set_A, Tot_lay_set_A, Tot_prim_set_A,␣
 ↪Tot_sec_set_A, Tot_zone_set_A, Final_A = Consolidation_models.
 ↪final_config_error(f'simulation_data_A_regular_with_{num_simulations}_simulations_and_dev_{
 ↪pkl')
Zone_A_spe, Prim_error_A_spe, Sec_error_A_spe, Tot_lay_error_A_spe,␣
 ↪Zone_prim_error_A_spe, Zone_sec_error_A_spe, Zone_tot_error_A_spe,␣
 ↪Ini_set_A_spe, Ini_error_A_spe, Set_A_spe, Set_error_A_spe, Prim_set_A_spe,␣
 ↪Sec_set_A_spe, Tot_lay_set_A_spe, Tot_prim_set_A_spe, Tot_sec_set_A_spe,␣
 ↪Tot_zone_set_A_spe, Final_A_spe = Consolidation_models.
 ↪final_config_error(f'simulation_data_A_special_with_{num_simulations}_simulations_and_dev_{
 ↪pkl')
Zone_B, Prim_error_B, Sec_error_B, Tot_lay_error_B, Zone_prim_error_B,␣
 ↪Zone_sec_error_B, Zone_tot_error_B, Ini_set_B, Ini_error_B, Set_B,␣
 ↪Set_error_B, Prim_set_B, Sec_set_B, Tot_lay_set_B, Tot_prim_set_B,␣
 ↪Tot_sec_set_B, Tot_zone_set_B, Final_B = Consolidation_models.
 ↪final_config_error(f'simulation_data_B_regular_with_{num_simulations}_simulations_and_dev_{
 ↪pkl')
Zone_B_spe, Prim_error_B_spe, Sec_error_B_spe, Tot_lay_error_B_spe,␣
 ↪Zone_prim_error_B_spe, Zone_sec_error_B_spe, Zone_tot_error_B_spe,␣
 ↪Ini_set_B_spe, Ini_error_B_spe, Set_B_spe, Set_error_B_spe, Prim_set_B_spe,␣
 ↪Sec_set_B_spe, Tot_lay_set_B_spe, Tot_prim_set_B_spe, Tot_sec_set_B_spe,␣
 ↪Tot_zone_set_B_spe, Final_B_spe = Consolidation_models.
 ↪final_config_error(f'simulation_data_B_special_with_{num_simulations}_simulations_and_dev_{
 ↪pkl')
```

```
Zone_C, Prim_error_C, Sec_error_C, Tot_lay_error_C, Zone_prim_error_C,␣
 ↪Zone_sec_error_C, Zone_tot_error_C, Ini_set_C, Ini_error_C, Set_C,␣
 ↪Set_error_C, Prim_set_C, Sec_set_C, Tot_lay_set_C, Tot_prim_set_C,␣
 ↪Tot_sec_set_C, Tot_zone_set_C, Final_C = Consolidation_models.
 ↪final_config_error(f'simulation_data_C_regular_with_{num_simulations}_simulations_and_dev_{
 ↪pkl')
Zone_D, Prim_error_D, Sec_error_D, Tot_lay_error_D, Zone_prim_error_D,␣
 ↪Zone_sec_error_D, Zone_tot_error_D, Ini_set_D, Ini_error_D, Set_D,␣
 ↪Set_error_D, Prim_set_D, Sec_set_D, Tot_lay_set_D, Tot_prim_set_D,␣
 ↪Tot_sec_set_D, Tot_zone_set_D, Final_D  = Consolidation_models.
 ↪final_config_error(f'simulation_data_D_regular_with_{num_simulations}_simulations_and_dev_{
 ↪pkl')
Zone_D_spe, Prim_error_D_spe, Sec_error_D_spe, Tot_lay_error_D_spe,␣
 ↪Zone_prim_error_D_spe, Zone_sec_error_D_spe, Zone_tot_error_D_spe,␣
 ↪Ini_set_D_spe, Ini_error_D_spe, Set_D_spe, Set_error_D_spe, Prim_set_D_spe,␣
 ↪Sec_set_D_spe, Tot_lay_set_D_spe, Tot_prim_set_D_spe, Tot_sec_set_D_spe,␣
 ↪Tot_zone_set_D_spe, Final_D_spe = Consolidation_models.
 ↪final_config_error(f'simulation_data_D_special_with_{num_simulations}_simulations_and_dev_{
 ↪pkl')
Zone_E, Prim_error_E, Sec_error_E, Tot_lay_error_E, Zone_prim_error_E,␣
 ↪Zone_sec_error_E, Zone_tot_error_E, Ini_set_E, Ini_error_E, Set_E,␣
 ↪Set_error_E, Prim_set_E, Sec_set_E, Tot_lay_set_E, Tot_prim_set_E,␣
 ↪Tot_sec_set_E, Tot_zone_set_E, Final_E  = Consolidation_models.
 ↪final_config_error(f'simulation_data_E_regular_with_{num_simulations}_simulations_and_dev_{
 ↪pkl')
Zone_E_spe, Prim_error_E_spe, Sec_error_E_spe, Tot_lay_error_E_spe,␣
 ↪Zone_prim_error_E_spe, Zone_sec_error_E_spe, Zone_tot_error_E_spe,␣
 ↪Ini_set_E_spe, Ini_error_E_spe, Set_E_spe, Set_error_E_spe, Prim_set_E_spe,␣
 ↪Sec_set_E_spe, Tot_lay_set_E_spe, Tot_prim_set_E_spe, Tot_sec_set_E_spe,␣
 ↪Tot_zone_set_E_spe, Final_E_spe =  Consolidation_models.
 ↪final_config_error(f'simulation_data_E_special_with_{num_simulations}_simulations_and_dev_{
 ↪pkl')
end = time.perf_counter()

print(f'Time it takes for error- calculation: {(end-start):.2f} sec or␣
 ↪{(end-start)/60:.2f} minutes')
```

```
Time it takes for error- calculation: 7.13 sec or 0.12 minutes
```

[12]:
```
superimposed_settlements = [np.sum(Zone_A[2], axis = 0), np.sum(Zone_B[2], axis␣
 ↪= 0), np.sum(Zone_C[2], axis = 0), np.sum(Zone_D[2], axis = 0), np.
 ↪sum(Zone_E[2], axis = 0)]
superimposed_settlements_spe = [np.sum(Zone_A_spe[2], axis = 0), np.
 ↪sum(Zone_B_spe[2], axis = 0), np.sum(Zone_D_spe[2], axis = 0), np.
 ↪sum(Zone_E_spe[2], axis = 0)]
```

```
Cre_values = [np.sum(Zone_A[3], axis = 0), np.sum(Zone_B[3], axis = 0), np.
 ↪sum(Zone_C[3], axis = 0), np.sum(Zone_D[3], axis = 0), np.sum(Zone_E[3],␣
 ↪axis = 0)]
Cre_values_spe = [np.sum(Zone_A_spe[3], axis = 0), np.sum(Zone_B_spe[3], axis =␣
 ↪0), np.sum(Zone_D_spe[3], axis = 0), np.sum(Zone_E_spe[3], axis = 0)]

Tot_prim_set = [Tot_prim_set_A, Tot_prim_set_B, Tot_prim_set_C, Tot_prim_set_D,␣
 ↪Tot_prim_set_E]
prim_mean_superimposed_settlements = [Zone_prim_error_A[0],␣
 ↪Zone_prim_error_B[0], Zone_prim_error_C[0], Zone_prim_error_D[0],␣
 ↪Zone_prim_error_E[0]]
prim_std_superimposed_settlements = [Zone_prim_error_A[1],␣
 ↪Zone_prim_error_B[1], Zone_prim_error_C[1], Zone_prim_error_D[1],␣
 ↪Zone_prim_error_E[1]]
prim_fi_perc_superimposed_settlements = [Zone_prim_error_A[2],␣
 ↪Zone_prim_error_B[2], Zone_prim_error_C[2], Zone_prim_error_D[2],␣
 ↪Zone_prim_error_E[2]]
prim_la_perc_superimposed_settlements = [Zone_prim_error_A[3],␣
 ↪Zone_prim_error_B[3], Zone_prim_error_C[3], Zone_prim_error_D[3],␣
 ↪Zone_prim_error_E[3]]

Tot_prim_set_spe = [Tot_prim_set_A_spe, Tot_prim_set_B_spe, Tot_prim_set_D_spe,␣
 ↪Tot_prim_set_E_spe]
prim_mean_superimposed_settlements_spe = [Zone_prim_error_A_spe[0],␣
 ↪Zone_prim_error_B_spe[0], Zone_prim_error_D_spe[0], Zone_prim_error_E_spe[0]]
prim_std_superimposed_settlements_spe = [Zone_prim_error_A_spe[1],␣
 ↪Zone_prim_error_B_spe[1], Zone_prim_error_D_spe[1], Zone_prim_error_E_spe[1]]
prim_fi_perc_superimposed_settlements_spe = [Zone_prim_error_A_spe[2],␣
 ↪Zone_prim_error_B_spe[2], Zone_prim_error_D_spe[2], Zone_prim_error_E_spe[2]]
prim_la_perc_superimposed_settlements_spe = [Zone_prim_error_A_spe[3],␣
 ↪Zone_prim_error_B_spe[3], Zone_prim_error_D_spe[3], Zone_prim_error_E_spe[3]]

Tot_sec_set = [Tot_sec_set_A, Tot_sec_set_B, Tot_sec_set_C, Tot_sec_set_D,␣
 ↪Tot_sec_set_E]
sec_mean_superimposed_settlements = [Zone_sec_error_A[0], Zone_sec_error_B[0],␣
 ↪Zone_sec_error_C[0], Zone_sec_error_D[0], Zone_sec_error_E[0]]
sec_std_superimposed_settlements = [Zone_sec_error_A[1], Zone_sec_error_B[1],␣
 ↪Zone_sec_error_C[1], Zone_sec_error_D[1], Zone_sec_error_E[1]]
sec_fi_perc_superimposed_settlements = [Zone_sec_error_A[2],␣
 ↪Zone_sec_error_B[2], Zone_sec_error_C[2], Zone_sec_error_D[2],␣
 ↪Zone_sec_error_E[2]]
sec_la_perc_superimposed_settlements = [Zone_sec_error_A[3],␣
 ↪Zone_sec_error_B[3], Zone_sec_error_C[3], Zone_sec_error_D[3],␣
 ↪Zone_sec_error_E[3]]
```

```
Tot_sec_set_spe = [Tot_sec_set_A_spe, Tot_sec_set_B_spe, Tot_sec_set_D_spe,␣
 ↪Tot_sec_set_E_spe]
sec_mean_superimposed_settlements_spe = [Zone_sec_error_A_spe[0],␣
 ↪Zone_sec_error_B_spe[0], Zone_sec_error_D_spe[0], Zone_sec_error_E_spe[0]]
sec_std_superimposed_settlements_spe = [Zone_sec_error_A_spe[1],␣
 ↪Zone_sec_error_B_spe[1], Zone_sec_error_D_spe[1], Zone_sec_error_E_spe[1]]
sec_fi_perc_superimposed_settlements_spe = [Zone_sec_error_A_spe[2],␣
 ↪Zone_sec_error_B_spe[2], Zone_sec_error_D_spe[2], Zone_sec_error_E_spe[2]]
sec_la_perc_superimposed_settlements_spe = [Zone_sec_error_A_spe[3],␣
 ↪Zone_sec_error_B_spe[3], Zone_sec_error_D_spe[3], Zone_sec_error_E_spe[3]]

Tot_set_zone = [Tot_zone_set_A, Tot_zone_set_B, Tot_zone_set_C, Tot_zone_set_D,␣
 ↪Tot_zone_set_E]
Tot_mean_superimposed_settlements =[Zone_tot_error_A[0], Zone_tot_error_B[0],␣
 ↪Zone_tot_error_C[0], Zone_tot_error_D[0], Zone_tot_error_E[0]]
Tot_std_superimposed_settlements = [Zone_tot_error_A[1], Zone_tot_error_B[1],␣
 ↪Zone_tot_error_C[1], Zone_tot_error_D[1], Zone_tot_error_E[1]]
Tot_fi_perc_superimposed_settlements =[Zone_tot_error_A[2],␣
 ↪Zone_tot_error_B[2], Zone_tot_error_C[2], Zone_tot_error_D[2],␣
 ↪Zone_tot_error_E[2]]
Tot_la_perc_superimposed_settlements = [Zone_tot_error_A[3],␣
 ↪Zone_tot_error_B[3], Zone_tot_error_C[3], Zone_tot_error_D[3],␣
 ↪Zone_tot_error_E[3]]

Tot_set_zone_spe = [Tot_zone_set_A_spe, Tot_zone_set_B_spe, Tot_zone_set_D_spe,␣
 ↪Tot_zone_set_E_spe]
Tot_mean_superimposed_settlements_spe =[Zone_tot_error_A_spe[0],␣
 ↪Zone_tot_error_B_spe[0], Zone_tot_error_D_spe[0], Zone_tot_error_E_spe[0]]
Tot_std_superimposed_settlements_spe = [Zone_tot_error_A_spe[1],␣
 ↪Zone_tot_error_B_spe[1], Zone_tot_error_D_spe[1], Zone_tot_error_E_spe[1]]
Tot_fi_perc_superimposed_settlements_spe =[Zone_tot_error_A_spe[2],␣
 ↪Zone_tot_error_B_spe[2], Zone_tot_error_D_spe[2], Zone_tot_error_E_spe[2]]
Tot_la_perc_superimposed_settlements_spe = [Zone_tot_error_A_spe[3],␣
 ↪Zone_tot_error_B_spe[3], Zone_tot_error_D_spe[3], Zone_tot_error_E_spe[3]]

common_times = np.intersect1d(time_days, t)
indices_time_days = np.where(np.isin(time_days, common_times))[0]
indices_t = np.where(np.isin(t, common_times))[0]

Cre_values = np.array(Cre_values)
Cre_values_spe = np.array(Cre_values_spe)
superimposed_settlements = np.array(superimposed_settlements)
superimposed_settlements_spe = np.array(superimposed_settlements_spe)
```

```
Tot_set = np.append(Cre_values[:,indices_t] + superimposed_settlements[:
 ↪,indices_time_days], np.append(superimposed_settlements[:,-2].reshape(-1,␣
 ↪1),superimposed_settlements[:,-1].reshape(-1, 1), axis = 1), axis = 1)
Tot_set = np.append(Tot_set, Cre_values[:,(indices_t[-1])+2:-1], axis = 1)
Tot_set_spe = np.append(Cre_values_spe[:,indices_t] +␣
 ↪superimposed_settlements_spe[:,indices_time_days], np.
 ↪append(superimposed_settlements_spe[:,-2].reshape(-1,␣
 ↪1),superimposed_settlements_spe[:,-1].reshape(-1, 1), axis = 1), axis = 1)
Tot_set_spe = np.append(Tot_set_spe, Cre_values_spe[:,(indices_t[-1])+2:-1],␣
 ↪axis = 1)
```

[13]:
```
Tot_set_ini_A, y_q_A, u_r_q_A, F_ex_q_A, x = Consolidation_models.
 ↪timo_kerr_DEM("Uniform", Parameters_A, t_l_A, kappa, E_b, v_b, W_ele_reg,␣
 ↪h_reg, np.sum(d_sc_A), L_ele_reg, a_u, b_u, Zone_A[7], dx)
Tot_set_ini_A_spe, y_q_A_spe, u_r_q_A_spe, F_ex_q_A_spe, x_spe =␣
 ↪Consolidation_models.timo_kerr_DEM("Uniform", Parameters_A, t_l_A_spe,␣
 ↪kappa, E_b, v_b, W_ele_spe, h_spe, np.sum(d_sc_A_spe), L_ele_spe, a_u, b_u,␣
 ↪Zone_A_spe[7], dx)
Tot_set_ini_B, y_q_B, u_r_q_B, F_ex_q_B, x = Consolidation_models.
 ↪timo_kerr_DEM("Uniform", Parameters_B, t_l_B, kappa, E_b, v_b, W_ele_reg,␣
 ↪h_reg, np.sum(d_sc_B), L_ele_reg, a_u, b_u, Zone_B[7], dx)
Tot_set_ini_B_spe, y_q_B_spe, u_r_q_B_spe, F_ex_q_B_spe, x_spe  =␣
 ↪Consolidation_models.timo_kerr_DEM("Uniform", Parameters_B, t_l_B_spe,␣
 ↪kappa, E_b, v_b, W_ele_spe, h_spe, np.sum(d_sc_B_spe), L_ele_spe, a_u, b_u,␣
 ↪Zone_B_spe[7], dx)
Tot_set_ini_C, y_q_C, u_r_q_C, F_ex_q_C, x = Consolidation_models.
 ↪timo_kerr_DEM("Uniform", Parameters_C, t_l_C,  kappa, E_b, v_b, W_ele_reg,␣
 ↪h_reg, np.sum(d_sc_C), L_ele_reg, a_u, b_u, Zone_C[7], dx)
Tot_set_ini_D, y_q_D, u_r_q_D, F_ex_q_D, x = Consolidation_models.
 ↪timo_kerr_DEM("Uniform", Parameters_D, t_l_D,  kappa, E_b, v_b, W_ele_reg,␣
 ↪h_reg, np.sum(d_sc_D), L_ele_reg, a_u, b_u, Zone_D[7], dx)
Tot_set_ini_D_spe, y_q_D_spe, u_r_q_D_spe, F_ex_q_D_spe, x_spe =␣
 ↪Consolidation_models.timo_kerr_DEM("Uniform", Parameters_D, t_l_D_spe, ␣
 ↪kappa, E_b, v_b, W_ele_spe, h_spe, np.sum(d_sc_D_spe), L_ele_spe, a_u, b_u,␣
 ↪Zone_D_spe[7], dx)
Tot_set_ini_E, y_q_E, u_r_q_E, F_ex_q_E, x = Consolidation_models.
 ↪timo_kerr_DEM("Uniform", Parameters_E, t_l_E, kappa, E_b, v_b, W_ele_reg,␣
 ↪h_reg, np.sum(d_sc_E), L_ele_reg, a_u, b_u, Zone_E[7], dx)
Tot_set_ini_E_spe, y_q_E_spe, u_r_q_E_spe, F_ex_q_E_spe, x_spe =␣
 ↪Consolidation_models.timo_kerr_DEM("Uniform", Parameters_E, t_l_E_spe, ␣
 ↪kappa, E_b, v_b, W_ele_spe, h_spe, np.sum(d_sc_E_spe), L_ele_spe, a_u, b_u,␣
 ↪Zone_E_spe[7], dx)
```

[14]:
```
arrays_prim_con = [
    Zone_A[2],
    Zone_A_spe[2],
```

```
    Zone_B[2],
    Zone_B_spe[2],
    Zone_C[2],
    Zone_D[2],
    Zone_D_spe[2],
    Zone_E[2],
    Zone_E_spe[2],
]

# Concatenate all into one long array
combined_prim_con = np.concatenate(arrays_prim_con)

# Now get min and max
y_min_prim_con = np.min(combined_prim_con)
y_max_prim_con = np.max(combined_prim_con)
```

[15]:
```
fig, axs = plt.subplots(2, 1, figsize=(6,5), constrained_layout = True)
Plotting.primary_consolidation('A', 'regular', Zone_A[4], Zone_A[5], time_days,␣
 ↪Zone_A[2], 0, axs, fig, y_min_prim_con, y_max_prim_con)
Plotting.primary_consolidation('A', 'special', Zone_A_spe[4], Zone_A_spe[5],␣
 ↪time_days, Zone_A_spe[2], 1, axs, fig, y_min_prim_con, y_max_prim_con)
plt.savefig(f'Pictures/Primary_Consolidation_of_separated_layers/
 ↪Primary_Consolidation_of_separated_layers_Zone_A')
plt.close()

fig, axs = plt.subplots(2, 1, figsize=(6,5), constrained_layout = True)
Plotting.primary_consolidation('B', 'regular', Zone_B[4], Zone_B[5], time_days,␣
 ↪Zone_B[2], 0, axs, fig, y_min_prim_con, y_max_prim_con)
Plotting.primary_consolidation('B', 'special', Zone_B_spe[4], Zone_B_spe[5],␣
 ↪time_days, Zone_B_spe[2], 1, axs, fig, y_min_prim_con, y_max_prim_con)
plt.savefig(f'Pictures/Primary_Consolidation_of_separated_layers/
 ↪Primary_Consolidation_of_separated_layers_Zone_B')
plt.close()

fig, axs = plt.subplots(2, 1, figsize=(6,5), constrained_layout = True)
Plotting.primary_consolidation('C', 'regular', Zone_C[4], Zone_C[5], time_days,␣
 ↪Zone_C[2], 0, axs, fig, y_min_prim_con, y_max_prim_con)
axs[1].axis('off')
plt.savefig(f'Pictures/Primary_Consolidation_of_separated_layers/
 ↪Primary_Consolidation_of_separated_layers_Zone_C')
plt.close()

fig, axs = plt.subplots(2, 1, figsize=(6,5), constrained_layout = True)
Plotting.primary_consolidation('D', 'regular', Zone_D[4], Zone_D[5], time_days,␣
 ↪Zone_D[2], 0, axs, fig, y_min_prim_con, y_max_prim_con)
```

```
Plotting.primary_consolidation('D', 'special', Zone_D_spe[4], Zone_D_spe[5],␣
 ↪time_days, Zone_D_spe[2], 1, axs, fig, y_min_prim_con, y_max_prim_con)
plt.savefig(f'Pictures/Primary_Consolidation_of_separated_layers/
 ↪Primary_Consolidation_of_separated_layers_Zone_D')
plt.close()

fig, axs = plt.subplots(2, 1, figsize=(6,5), constrained_layout = True)
Plotting.primary_consolidation('E', 'regular', Zone_E[4], Zone_E[5], time_days,␣
 ↪Zone_E[2], 0, axs, fig, y_min_prim_con, y_max_prim_con)
Plotting.primary_consolidation('E', 'special', Zone_E_spe[4], Zone_E_spe[5],␣
 ↪time_days, Zone_E_spe[2], 1,axs, fig, y_min_prim_con, y_max_prim_con)
plt.savefig(f'Pictures/Primary_Consolidation_of_separated_layers/
 ↪Primary_Consolidation_of_separated_layers_Zone_E')
plt.close()
```

[16]:
```
arrays_sec_con = [
    Zone_A[3],
    Zone_A_spe[3],
    Zone_B[3],
    Zone_B_spe[3],
    Zone_C[3],
    Zone_D[3],
    Zone_D_spe[3],
    Zone_E[3],
    Zone_E_spe[3],
]

# Concatenate all into one long array
combined_sec_con = np.concatenate(arrays_sec_con)

# Now get min and max
y_min_sec_con = np.min(combined_sec_con)
y_max_sec_con = np.max(combined_sec_con)
```

[17]:
```
# fig, axs = plt.subplots(2, 1, figsize=(7,7), constrained_layout = True)

# Plotting.secondary_consolidation('A', 'regular', Zone_A[4], Zone_A[5], t,␣
 ↪Zone_A[3], 0, axs, fig, Zone_A[6], y_min_sec_con, y_max_sec_con)
# Plotting.secondary_consolidation('A', 'special', Zone_A_spe[4],␣
 ↪Zone_A_spe[5], t, Zone_A_spe[3], 1, axs, fig, Zone_A_spe[6], y_min_sec_con,␣
 ↪y_max_sec_con)

# plt.savefig(f'Pictures/Secondary_Consolidation_of_separated_layers/
 ↪Secondary_Consolidation_of_separated_layers_Zone_A')
# plt.close()
```

```
# fig, axs = plt.subplots(2, 1, figsize=(7,7), constrained_layout = True)

# Plotting.secondary_consolidation('B', 'regular', Zone_B[4], Zone_B[5], t,␣
 ↪Zone_B[3], 0, axs, fig,  Zone_B[6], y_min_sec_con, y_max_sec_con)
# Plotting.secondary_consolidation('B', 'special', Zone_B_spe[4],␣
 ↪Zone_B_spe[5], t, Zone_B_spe[3], 1, axs, fig,  Zone_B_spe[6], y_min_sec_con,␣
 ↪y_max_sec_con)

# plt.savefig(f'Pictures/Secondary_Consolidation_of_separated_layers/
 ↪Secondary_Consolidation_of_separated_layers_Zone_B')
# plt.close()

# fig, axs = plt.subplots(2, 1, figsize=(7,7), constrained_layout = True)

# Plotting.secondary_consolidation('C', 'regular', Zone_C[4], Zone_C[5], t,␣
 ↪Zone_C[3], 0, axs, fig, Zone_C[6], y_min_sec_con, y_max_sec_con)
# axs[1].axis('off')

# plt.savefig(f'Pictures/Secondary_Consolidation_of_separated_layers/
 ↪Secondary_Consolidation_of_separated_layers_Zone_C')
# plt.close()

# fig, axs = plt.subplots(2, 1, figsize=(7,7), constrained_layout = True)

# Plotting.secondary_consolidation('D', 'regular', Zone_D[4], Zone_D[5], t,␣
 ↪Zone_D[3], 0, axs, fig, Zone_D[6], y_min_sec_con, y_max_sec_con)
# Plotting.secondary_consolidation('D', 'special', Zone_D_spe[4],␣
 ↪Zone_D_spe[5], t, Zone_D_spe[3], 1, axs, fig, Zone_D_spe[6], y_min_sec_con,␣
 ↪y_max_sec_con)

# plt.savefig(f'Pictures/Secondary_Consolidation_of_separated_layers/
 ↪Secondary_Consolidation_of_separated_layers_Zone_D')
# plt.close()

# fig, axs = plt.subplots(2, 1, figsize=(7,7), constrained_layout = True)

# Plotting.secondary_consolidation('E', 'regular', Zone_E[4], Zone_E[5], t,␣
 ↪Zone_E[3], 0, axs, fig, Zone_E[6], y_min_sec_con, y_max_sec_con)
# Plotting.secondary_consolidation('E', 'special', Zone_E_spe[4],␣
 ↪Zone_E_spe[5], t, Zone_E_spe[3], 1, axs, fig, Zone_E_spe[6], y_min_sec_con,␣
 ↪y_max_sec_con)

# plt.savefig(f'Pictures/Secondary_Consolidation_of_separated_layers/
 ↪Secondary_Consolidation_of_separated_layers_Zone_E')
# plt.close()
```

```
# plt.savefig('Pictures/Secondary Settelement of seperated layers')
```

[18]:
```
# fig, axs = plt.subplots(2, 1, figsize = (8,8), constrained_layout = True)

# Plotting.set_time('Primary', Zone_lay, 'regular', time_days,␣
 ↪superimposed_settlements, start_times_rel_prot, fig, axs, 0)
# Plotting.set_time('Primary', Zone_lay_spe, 'special', time_days,␣
 ↪superimposed_settlements_spe, start_times_rel_prot, fig, axs, 1)

# plt.savefig('Pictures/
 ↪Primary_Secondary_and_Total_Consolidation_per_defined_zone/
 ↪Primary_Consolidation_per_defined_zone')
# plt.close()

# fig, axs = plt.subplots(2, 1, figsize = (8,8), constrained_layout = True)

# Plotting.set_time('Secondary', Zone_lay, 'regular', t/365, Cre_values,␣
 ↪start_times_rel_prot,fig, axs, 0)
# Plotting.set_time('Secondary', Zone_lay_spe, 'special', t/365,␣
 ↪Cre_values_spe, start_times_rel_prot,fig, axs, 1)

# plt.savefig('Pictures/
 ↪Primary_Secondary_and_Total_Consolidation_per_defined_zone/
 ↪Secondary_Consolidation_per_defined_zone')
# plt.close()

# fig, axs = plt.subplots(2, 1, figsize = (8,8), constrained_layout = True)

# Plotting.set_time('Total', Zone_lay, 'regular', t/365, Tot_set,␣
 ↪start_times_rel_prot,fig, axs, 0)
# Plotting.set_time('Total', Zone_lay_spe, 'special', t/365, Tot_set_spe,␣
 ↪start_times_rel_prot, fig, axs, 1)

# plt.savefig('Pictures/
 ↪Primary_Secondary_and_Total_Consolidation_per_defined_zone/
 ↪Total_Consolidation_per_defined_zone')
```

[19]:
```
arrays_prim_con_lay = [
    Prim_error_A[2],
    Prim_error_A[3],
    Prim_error_A_spe[2],
    Prim_error_A_spe[3],
    Prim_error_B[2],
    Prim_error_B[3],
    Prim_error_B_spe[2],
    Prim_error_B_spe[3],
```

```
    Prim_error_C[2],
    Prim_error_C[3],
    Prim_error_D[2],
    Prim_error_D[3],
    Prim_error_D_spe[2],
    Prim_error_D_spe[3],
    Prim_error_E[2],
    Prim_error_E[3],
    Prim_error_E_spe[2],
    Prim_error_E_spe[3],
]

# Concatenate all into one long array
combined_prim_con_lay = np.concatenate(arrays_prim_con_lay)

# Now get min and max
y_min_prim_con_lay = np.min(combined_prim_con_lay)
y_max_prim_con_lay = np.max(combined_prim_con_lay)
```

[20]:
```
arrays_sec_con_lay = [
    Sec_error_A[2],
    Sec_error_A[3],
    Sec_error_A_spe[2],
    Sec_error_A_spe[3],
    Sec_error_B[2],
    Sec_error_B[3],
    Sec_error_B_spe[2],
    Sec_error_B_spe[3],
    Sec_error_C[2],
    Sec_error_C[3],
    Sec_error_D[2],
    Sec_error_D[3],
    Sec_error_D_spe[2],
    Sec_error_D_spe[3],
    Sec_error_E[2],
    Sec_error_E[3],
    Sec_error_E_spe[2],
    Sec_error_E_spe[3],
]

# Concatenate all into one long array
combined_sec_con_lay = np.concatenate(arrays_sec_con_lay)

# Now get min and max
y_min_sec_con_lay = np.min(combined_sec_con_lay)
y_max_sec_con_lay = np.max(combined_sec_con_lay)
```

[21]:
```
arrays_tot_con_lay = [
    Tot_lay_error_A[2],
    Tot_lay_error_A[3],
    Tot_lay_error_A_spe[2],
    Tot_lay_error_A_spe[3],
    Tot_lay_error_B[2],
    Tot_lay_error_B[3],
    Tot_lay_error_B_spe[2],
    Tot_lay_error_B_spe[3],
    Tot_lay_error_C[2],
    Tot_lay_error_C[3],
    Tot_lay_error_D[2],
    Tot_lay_error_D[3],
    Tot_lay_error_D_spe[2],
    Tot_lay_error_D_spe[3],
    Tot_lay_error_E[2],
    Tot_lay_error_E[3],
    Tot_lay_error_E_spe[2],
    Tot_lay_error_E_spe[3],
]

# Concatenate all into one long array
combined_tot_con_lay = np.concatenate(arrays_tot_con_lay)

# Now get min and max
y_min_tot_con_lay = np.min(combined_tot_con_lay)
y_max_tot_con_lay = np.max(combined_tot_con_lay)
```

[22]:
```
# fig, axs = plt.subplots(2,1,figsize=(12,8))

# Plotting.errorbar_lay('A', 'Primary', 'regular', Zone_A, time_days,␣
 ↪Prim_error_A[0], Prim_error_A[1], Prim_error_A[2], Prim_error_A[3], t_l_A,␣
 ↪fig, axs, 0)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,8))
# Plotting.errorbar_lay('A', 'Secondary', 'regular', Zone_A, t/365,␣
 ↪Sec_error_A[0], Sec_error_A[1], Sec_error_A[2], Sec_error_A[3], t_l_A, fig,␣
 ↪axs, 0)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,8))
# Plotting.errorbar_lay('A', 'Total', 'regular', Zone_A, t/365,␣
 ↪Tot_lay_error_A[0], Tot_lay_error_A[1], Tot_lay_error_A[2],␣
 ↪Tot_lay_error_A[3], t_l_A, fig, axs, 0)
```

```
# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,8))
# Plotting.errorbar_lay('A', 'Primary', 'special', Zone_A_spe, time_days,␣
 ↪Prim_error_A_spe[0], Prim_error_A_spe[1], Prim_error_A_spe[2],␣
 ↪Prim_error_A_spe[3], t_l_A_spe, fig, axs, 0)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,8))
# Plotting.errorbar_lay('A', 'Secondary', 'special', Zone_A_spe, t/365,␣
 ↪Sec_error_A_spe[0], Sec_error_A_spe[1], Sec_error_A_spe[2],␣
 ↪Sec_error_A_spe[3], t_l_A_spe, fig, axs, 0)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,8))
# Plotting.errorbar_lay('A', 'Total', 'special', Zone_A_spe, t/365,␣
 ↪Tot_lay_error_A_spe[0], Tot_lay_error_A_spe[1], Tot_lay_error_A_spe[2],␣
 ↪Tot_lay_error_A_spe[3], t_l_A_spe, fig, axs, 0)

# plt.close()
```

[23]:
```
# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('B', 'Primary', 'regular', Zone_B, time_days,␣
 ↪Prim_error_B[0], Prim_error_B[1], Prim_error_B[2], Prim_error_B[3],␣
 ↪Prim_set_B, t_l_B, fig, axs, 0, y_min_prim_con_lay, y_max_prim_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('B', 'Secondary', 'regular', Zone_B, t/365,␣
 ↪Sec_error_B[0], Sec_error_B[1], Sec_error_B[2], Sec_error_B[3], Sec_set_B,␣
 ↪t_l_B, fig, axs, 0, y_min_sec_con_lay, y_max_sec_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('B', 'Total', 'regular', Zone_B, t/365,␣
 ↪Tot_lay_error_B[0], Tot_lay_error_B[1], Tot_lay_error_B[2],␣
 ↪Tot_lay_error_B[3], Tot_lay_set_B, t_l_B, fig, axs, 0, y_min_tot_con_lay,␣
 ↪y_max_tot_con_lay)
```

```
# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('B', 'Primary', 'special', Zone_B_spe, time_days,␣
 ↪Prim_error_B_spe[0], Prim_error_B_spe[1], Prim_error_B_spe[2],␣
 ↪Prim_error_B_spe[3], Prim_set_B_spe, t_l_B_spe, fig, axs, 0,␣
 ↪y_min_prim_con_lay, y_max_prim_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('B', 'Secondary', 'special', Zone_B_spe, t/365,␣
 ↪Sec_error_B_spe[0], Sec_error_B_spe[1], Sec_error_B_spe[2],␣
 ↪Sec_error_B_spe[3], Sec_set_B_spe, t_l_B_spe, fig, axs, 0,␣
 ↪y_min_sec_con_lay, y_max_sec_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('B', 'Total', 'special', Zone_B_spe, t/365,␣
 ↪Tot_lay_error_B_spe[0], Tot_lay_error_B_spe[1], Tot_lay_error_B_spe[2],␣
 ↪Tot_lay_error_B_spe[3], Tot_lay_set_B_spe, t_l_B_spe, fig, axs, 0,␣
 ↪y_min_tot_con_lay, y_max_tot_con_lay)

# plt.close()
```

[24]:
```
# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('C', 'Primary', 'regular', Zone_C, time_days,␣
 ↪Prim_error_C[0], Prim_error_C[1], Prim_error_C[2], Prim_error_C[3],␣
 ↪Prim_set_C, t_l_C, fig, axs, 0, y_min_prim_con_lay, y_max_prim_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('C', 'Secondary', 'regular', Zone_C, t/365,␣
 ↪Sec_error_C[0], Sec_error_C[1], Sec_error_C[2], Sec_error_C[3], Sec_set_C,␣
 ↪t_l_C, fig, axs, 0, y_min_sec_con_lay, y_max_sec_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))
```

```
# Plotting.errorbar_lay('C', 'Total', 'regular', Zone_C, t/365,␣
 ↪Tot_lay_error_C[0], Tot_lay_error_C[1], Tot_lay_error_C[2],␣
 ↪Tot_lay_error_C[3], Tot_lay_set_C, t_l_C, fig, axs, 0, y_min_tot_con_lay,␣
 ↪y_max_tot_con_lay)

# plt.close()
```

[25]:
```
# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('D', 'Primary', 'regular', Zone_D, time_days,␣
 ↪Prim_error_D[0], Prim_error_D[1], Prim_error_D[2], Prim_error_D[3],␣
 ↪Prim_set_D, t_l_D, fig, axs, 0, y_min_prim_con_lay, y_max_prim_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('D', 'Secondary', 'regular', Zone_D, t/365,␣
 ↪Sec_error_D[0], Sec_error_D[1], Sec_error_D[2], Sec_error_D[3], Sec_set_D,␣
 ↪t_l_D, fig, axs, 0, y_min_sec_con_lay, y_max_sec_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('D', 'Total', 'regular', Zone_D, t/365,␣
 ↪Tot_lay_error_D[0], Tot_lay_error_D[1], Tot_lay_error_D[2],␣
 ↪Tot_lay_error_D[3], Tot_lay_set_D, t_l_D, fig, axs, 0, y_min_tot_con_lay,␣
 ↪y_max_tot_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('D', 'Primary', 'special', Zone_D_spe, time_days,␣
 ↪Prim_error_D_spe[0], Prim_error_D_spe[1], Prim_error_D_spe[2],␣
 ↪Prim_error_D_spe[3], Prim_set_D_spe, t_l_D_spe, fig, axs, 0,␣
 ↪y_min_prim_con_lay, y_max_prim_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('D', 'Secondary', 'special', Zone_D_spe, t/365,␣
 ↪Sec_error_D_spe[0], Sec_error_D_spe[1], Sec_error_D_spe[2],␣
 ↪Sec_error_D_spe[3], Sec_set_D_spe, t_l_D_spe, fig, axs, 0,␣
 ↪y_min_sec_con_lay, y_max_sec_con_lay)
```

```
# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('D', 'Total', 'special', Zone_D_spe, t/365,␣
 ↪Tot_lay_error_D_spe[0], Tot_lay_error_D_spe[1], Tot_lay_error_D_spe[2],␣
 ↪Tot_lay_error_D_spe[3], Tot_lay_set_D_spe, t_l_D_spe, fig, axs, 0,␣
 ↪y_min_tot_con_lay, y_max_tot_con_lay)

# plt.close()
```

[26]:
```
# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('E', 'Primary', 'regular', Zone_E, time_days,␣
 ↪Prim_error_E[0], Prim_error_E[1], Prim_error_E[2], Prim_error_E[3],␣
 ↪Prim_set_E, t_l_E, fig, axs, 0, y_min_prim_con_lay, y_max_prim_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('E', 'Secondary', 'regular', Zone_E, t/365,␣
 ↪Sec_error_E[0], Sec_error_E[1], Sec_error_E[2], Sec_error_E[3], Sec_set_E,␣
 ↪t_l_E, fig, axs, 0, y_min_sec_con_lay, y_max_sec_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('E', 'Total', 'regular', Zone_E, t/365,␣
 ↪Tot_lay_error_E[0], Tot_lay_error_E[1], Tot_lay_error_E[2],␣
 ↪Tot_lay_error_E[3], Tot_lay_set_E, t_l_E, fig, axs, 0, y_min_tot_con_lay,␣
 ↪y_max_tot_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('E', 'Primary', 'special', Zone_E_spe, time_days,␣
 ↪Prim_error_E_spe[0], Prim_error_E_spe[1], Prim_error_E_spe[2],␣
 ↪Prim_error_E_spe[3], Prim_set_E_spe, t_l_E_spe, fig, axs, 0,␣
 ↪y_min_prim_con_lay, y_max_prim_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))
```

```
# Plotting.errorbar_lay('E', 'Secondary', 'special', Zone_E_spe, t/365,␣
 ↪Sec_error_E_spe[0], Sec_error_E_spe[1], Sec_error_E_spe[2],␣
 ↪Sec_error_E_spe[3], Sec_set_E_spe, t_l_E_spe, fig, axs, 0,␣
 ↪y_min_sec_con_lay, y_max_sec_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('E', 'Total', 'special', Zone_E_spe, t/365,␣
 ↪Tot_lay_error_E_spe[0], Tot_lay_error_E_spe[1], Tot_lay_error_E_spe[2],␣
 ↪Tot_lay_error_E_spe[3], Tot_lay_set_E_spe, t_l_E_spe, fig, axs, 0,␣
 ↪y_min_tot_con_lay, y_max_tot_con_lay)
# plt.close()
```

[27]:
```
# print(f'primary regular {np.array(Prim_error_E)[2][3][-1]*1000:.1f}')
# print(f'primary regular {np.array(Prim_error_E)[3][3][-1]*1000:.1f}')
# print(f'primary special {np.array(Prim_error_E_spe)[2][3][-1]*1000:.1f}')
# print(f'primary special {np.array(Prim_error_E_spe)[3][3][-1]*1000:.1f}')
# print(f'secondary regular {np.array(Sec_error_E)[2][3][-1]*1000:.1f}')
# print(f'secondary regular {np.array(Sec_error_E)[3][3][-1]*1000:.1f}')
# print(f'secondary special {np.array(Sec_error_E_spe)[2][3][-1]*1000:.1f}')
# print(f'secondary special {np.array(Sec_error_E_spe)[3][3][-1]*1000:.1f}')
```

[28]:
```
# print(f'primary regular {prim_fi_perc_superimposed_settlements[4][-1]*1000:.
 ↪1f}')
# print(f'primary regular {prim_la_perc_superimposed_settlements[4][-1]*1000:.
 ↪1f}')
# print(f'primary special␣
 ↪{prim_fi_perc_superimposed_settlements_spe[3][-1]*1000:.1f}')
# print(f'primary special␣
 ↪{prim_la_perc_superimposed_settlements_spe[3][-1]*1000:.1f}')
print(f'secondary regular B {sec_fi_perc_superimposed_settlements[1][-1]*1000:.
 ↪1f}')
print(f'secondary regular B {sec_la_perc_superimposed_settlements[1][-1]*1000:.
 ↪1f}')
print(f'secondary special B␣
 ↪{sec_fi_perc_superimposed_settlements_spe[1][-1]*1000:.1f}')
print(f'secondary special B␣
 ↪{sec_la_perc_superimposed_settlements_spe[1][-1]*1000:.1f}')
print(f'secondary regular C {sec_fi_perc_superimposed_settlements[2][-1]*1000:.
 ↪1f}')
print(f'secondary regular C {sec_la_perc_superimposed_settlements[2][-1]*1000:.
 ↪1f}')
```

```
print(f'secondary regular D {sec_fi_perc_superimposed_settlements[3][-1]*1000:.
↪1f}')
print(f'secondary regular D {sec_la_perc_superimposed_settlements[3][-1]*1000:.
↪1f}')
print(f'secondary special D␣
↪{sec_fi_perc_superimposed_settlements_spe[2][-1]*1000:.1f}')
print(f'secondary special D␣
↪{sec_la_perc_superimposed_settlements_spe[2][-1]*1000:.1f}')
print(f'secondary regular E {sec_fi_perc_superimposed_settlements[4][-1]*1000:.
↪1f}')
print(f'secondary regular E {sec_la_perc_superimposed_settlements[4][-1]*1000:.
↪1f}')
print(f'secondary special E␣
↪{sec_fi_perc_superimposed_settlements_spe[3][-1]*1000:.1f}')
print(f'secondary special E␣
↪{sec_la_perc_superimposed_settlements_spe[3][-1]*1000:.1f}')
# print(f'initial regular {- Ini_error_E[2][0]*1000:.1f}')
# print(f'initial regular {- Ini_error_E[3][0]*1000:.1f}')
# print(f'initial special {- Ini_error_E_spe[2][0]*1000:.1f}')
# print(f'initial special {- Ini_error_E_spe[3][0]*1000:.1f}')
```

```
secondary regular B 26.5
secondary regular B 129.1
secondary special B 18.5
secondary special B 118.9
secondary regular C 5.2
secondary regular C 34.3
secondary regular D -37.9
secondary regular D 47.1
secondary special D -55.7
secondary special D 34.4
secondary regular E -44.8
secondary regular E 44.6
secondary special E -46.1
secondary special E 44.2
```

[29]:
```
fig, axs = plt.subplots(2, 2, figsize = (16,9), constrained_layout = True)

# Plotting.errorbar_zone('Primary', Zone_lay, 'regular', time_days,␣
↪prim_mean_superimposed_settlements, prim_std_superimposed_settlements,␣
↪start_times_rel_prot, fig, axs, 0, 0)
# Plotting.errorbar_zone('Primary', Zone_lay_spe, 'special', time_days,␣
↪prim_mean_superimposed_settlements_spe,␣
↪prim_std_superimposed_settlements_spe, start_times_rel_prot, fig, axs, 1, 0)
```

```
Plotting.perc_zone('Primary', Zone_lay, 'regular', time_days,␣
 ↪prim_mean_superimposed_settlements, prim_std_superimposed_settlements,␣
 ↪prim_fi_perc_superimposed_settlements,␣
 ↪prim_la_perc_superimposed_settlements, Tot_prim_set, start_times_rel_prot,␣
 ↪fig, axs, 0)
Plotting.perc_zone('Primary', Zone_lay_spe, 'special', time_days,␣
 ↪prim_mean_superimposed_settlements_spe,␣
 ↪prim_std_superimposed_settlements_spe,␣
 ↪prim_fi_perc_superimposed_settlements_spe,␣
 ↪prim_la_perc_superimposed_settlements_spe, Tot_prim_set_spe,␣
 ↪start_times_rel_prot, fig, axs, 1)
plt.savefig('Pictures/
 ↪Variability_of_Primary_Secondary_and_Total_consolidation_per_defined_zone/
 ↪Variability_of_primary_consolidation_per_defined_zone')
plt.close()

fig, axs = plt.subplots(2, 2, figsize = (16,9), constrained_layout = True)

# Plotting.errorbar_zone('Secondary', Zone_lay, 'regular', t/365, ␣
 ↪sec_mean_superimposed_settlements, sec_std_superimposed_settlements,␣
 ↪start_times_rel_prot, fig, axs, 0, 0)
# Plotting.errorbar_zone('Secondary', Zone_lay_spe, 'special', t/365,␣
 ↪sec_mean_superimposed_settlements_spe, sec_std_superimposed_settlements_spe,␣
 ↪start_times_rel_prot, fig, axs, 1, 0)

Plotting.perc_zone('Secondary', Zone_lay, 'regular', t/365, ␣
 ↪sec_mean_superimposed_settlements, sec_std_superimposed_settlements,␣
 ↪sec_fi_perc_superimposed_settlements, sec_la_perc_superimposed_settlements,␣
 ↪Tot_sec_set, start_times_rel_prot, fig, axs, 0)
Plotting.perc_zone('Secondary', Zone_lay_spe, 'special', t/365,␣
 ↪sec_mean_superimposed_settlements_spe, sec_std_superimposed_settlements_spe,␣
 ↪sec_fi_perc_superimposed_settlements_spe,␣
 ↪sec_la_perc_superimposed_settlements_spe, Tot_sec_set_spe,␣
 ↪start_times_rel_prot, fig, axs, 1)

plt.savefig('Pictures/
 ↪Variability_of_Primary_Secondary_and_Total_consolidation_per_defined_zone/
 ↪Variability_of_secondary_consolidation_per_defined_zone')
plt.close()

fig, axs = plt.subplots(2, 2, figsize = (16,9), constrained_layout = True)

# Plotting.errorbar_zone('Total', Zone_lay, 'regular', t/365,␣
 ↪Tot_mean_superimposed_settlements, Tot_std_superimposed_settlements,␣
 ↪start_times_rel_prot, fig, axs, 0,0)
```

```
# Plotting.errorbar_zone('Total', Zone_lay_spe, 'special', t/365,␣
 ↪Tot_mean_superimposed_settlements_spe, Tot_std_superimposed_settlements_spe,␣
 ↪start_times_rel_prot, fig, axs, 1,0)

Plotting.perc_zone('Total', Zone_lay, 'regular', t/365,␣
 ↪Tot_mean_superimposed_settlements, Tot_std_superimposed_settlements,␣
 ↪Tot_fi_perc_superimposed_settlements, Tot_la_perc_superimposed_settlements,␣
 ↪Tot_set_zone, start_times_rel_prot, fig, axs, 0)
Plotting.perc_zone('Total', Zone_lay_spe, 'special', t/365,␣
 ↪Tot_mean_superimposed_settlements_spe, Tot_std_superimposed_settlements_spe,␣
 ↪ Tot_fi_perc_superimposed_settlements_spe,␣
 ↪Tot_la_perc_superimposed_settlements_spe, Tot_set_zone_spe,␣
 ↪start_times_rel_prot, fig, axs, 1)

plt.savefig('Pictures/
 ↪Variability_of_Primary_Secondary_and_Total_consolidation_per_defined_zone/
 ↪Variability_of_total_consolidation_per_defined_zone')
plt.close()
print('done')
```

```
done
```

[30]:
```
# fig, axs = plt.subplots(6, 1, figsize = (13,5*10), constrained_layout = True)

# Plotting.perc_zone('Primary', Zone_lay, 'regular', time_days,␣
 ↪prim_mean_superimposed_settlements, prim_fi_perc_superimposed_settlements,␣
 ↪prim_la_perc_superimposed_settlements, start_times_rel_prot, fig, axs, 0)
# Plotting.perc_zone('Primary', Zone_lay_spe, 'special', time_days,␣
 ↪prim_mean_superimposed_settlements_spe,␣
 ↪prim_fi_perc_superimposed_settlements_spe,␣
 ↪prim_la_perc_superimposed_settlements_spe, start_times_rel_prot, fig, axs, 1)
# Plotting.perc_zone('Secondary', Zone_lay, 'regular', t/365, ␣
 ↪sec_mean_superimposed_settlements, sec_fi_perc_superimposed_settlements,␣
 ↪sec_la_perc_superimposed_settlements, start_times_rel_prot, fig, axs, 2)
# Plotting.perc_zone('Secondary', Zone_lay_spe, 'special', t/365,␣
 ↪sec_mean_superimposed_settlements_spe,␣
 ↪sec_fi_perc_superimposed_settlements_spe,␣
 ↪sec_la_perc_superimposed_settlements_spe, start_times_rel_prot, fig, axs, 3)
# Plotting.perc_zone('Total', Zone_lay, 'regular', t/365,␣
 ↪Tot_mean_superimposed_settlements, Tot_fi_perc_superimposed_settlements,␣
 ↪Tot_la_perc_superimposed_settlements, start_times_rel_prot, fig, axs, 4)
# Plotting.perc_zone('Total', Zone_lay_spe, 'special', t/365,␣
 ↪Tot_mean_superimposed_settlements_spe,␣
 ↪Tot_fi_perc_superimposed_settlements_spe,␣
 ↪Tot_la_perc_superimposed_settlements_spe, start_times_rel_prot, fig, axs, 5)

# plt.savefig('Pictures/5th and 95th Percentile Settelement per Defined Zone')
```

[31]:
```
arrays_q = [
    Tot_set_ini_A,
    Tot_set_ini_A_spe,
    Tot_set_ini_B,
    Tot_set_ini_B_spe,
    Tot_set_ini_C,
    Tot_set_ini_D,
    Tot_set_ini_D_spe,
    Tot_set_ini_E,
    Tot_set_ini_E_spe
]

# Concatenate all into one long array
combined_q = np.concatenate(arrays_q)

# Now get min and max
y_min_q = np.min(combined_q)
y_max_q = np.max(combined_q)
```

[32]:
```
# fig, axs = plt.subplots(5, 2, figsize=(10,12))

# Plotting.plotting_q('A', 'regular', Zone_A[7], y_q_A, dx, L_ele_reg, u_r_q_A,␣
 ↪F_ex_q_A, 0, fig, axs, y_min_q, y_max_q)
# Plotting.plotting_q('A', 'special', Zone_A_spe[7], y_q_A_spe, dx, L_ele_spe,␣
 ↪u_r_q_A_spe, F_ex_q_A_spe, 0, fig, axs, y_min_q, y_max_q)
# Plotting.plotting_q('B', 'regular', Zone_B[7], y_q_B, dx, L_ele_reg, u_r_q_B,␣
 ↪F_ex_q_B, 1 , fig, axs, y_min_q, y_max_q)
# Plotting.plotting_q('B', 'special', Zone_B_spe[7], y_q_B_spe, dx, L_ele_spe,␣
 ↪u_r_q_B_spe, F_ex_q_B_spe, 1 , fig, axs, y_min_q, y_max_q)
# Plotting.plotting_q('C', 'regular', Zone_C[7], y_q_C, dx, L_ele_reg, u_r_q_C,␣
 ↪F_ex_q_C, 2 , fig, axs, y_min_q, y_max_q)

# axs[2,1].axis('off')

# Plotting.plotting_q('D', 'regular', Zone_D[7], y_q_D, dx, L_ele_reg, u_r_q_D,␣
 ↪F_ex_q_D, 3 , fig, axs, y_min_q, y_max_q)
# Plotting.plotting_q('D', 'special', Zone_D_spe[7], y_q_D_spe, dx, L_ele_spe,␣
 ↪u_r_q_D_spe, F_ex_q_D_spe, 3 , fig, axs, y_min_q, y_max_q)
# Plotting.plotting_q('E', 'regular', Zone_E[7], y_q_E, dx, L_ele_reg, u_r_q_E,␣
 ↪F_ex_q_E, 4 , fig, axs, y_min_q, y_max_q)
# Plotting.plotting_q('E', 'special', Zone_E_spe[7], y_q_E_spe, dx, L_ele_spe,␣
 ↪u_r_q_E_spe, F_ex_q_E_spe, 4 , fig, axs, y_min_q, y_max_q)


# plt.savefig('Pictures/Soil structure interaction over length complete beam')
# plt.close()
```

[33]:
```
# arrays_ini = [
#     -Ini_error_B[2],
#     -Ini_error_B[3],
#     -Ini_error_B_spe[2],
#     -Ini_error_B_spe[3],
#     -Ini_error_C[2],
#     -Ini_error_C[3],
#     -Ini_error_D[2],
#     -Ini_error_D[3],
#     -Ini_error_D_spe[2],
#     -Ini_error_D_spe[3],
#     -Ini_error_E[2],
#     -Ini_error_E[3],
#     -Ini_error_E_spe[2],
#     -Ini_error_E_spe[3],
# ]

# # Concatenate all into one long array
# combined_ini = np.concatenate(arrays_ini)

# # Now get min and max
# y_min_ini = np.min(combined_ini)
# y_max_ini = np.max(combined_ini)

# Plotting.errorbar_ini('B', 'regular', x, Ini_error_B, Ini_set_B, y_min_ini,␣
 ↪y_max_ini)
# Plotting.errorbar_ini('B', 'special', x_spe, Ini_error_B_spe, Ini_set_B_spe,␣
 ↪y_min_ini, y_max_ini)
# Plotting.errorbar_ini('C', 'regular', x, Ini_error_C, Ini_set_C, y_min_ini,␣
 ↪y_max_ini)
# Plotting.errorbar_ini('D', 'regular', x, Ini_error_D, Ini_set_D, y_min_ini,␣
 ↪y_max_ini)
# Plotting.errorbar_ini('D', 'special', x_spe, Ini_error_D_spe, Ini_set_D_spe,␣
 ↪y_min_ini, y_max_ini)
# Plotting.errorbar_ini('E', 'regular', x, Ini_error_E, Ini_set_E, y_min_ini,␣
 ↪y_max_ini)
# Plotting.errorbar_ini('E', 'special', x_spe, Ini_error_E_spe, Ini_set_E_spe,␣
 ↪y_min_ini, y_max_ini)
```

[34]:
```
# fig, axs = plt.subplots(nrows=2, ncols=2,  figsize=(8,8))
# Plotting.Set_zone('A', 'regular', times,  Set_error_A, x, 0, fig, axs)
# Plotting.Set_zone('A', 'special', times,  Set_error_A_spe, x_spe, 1, fig, axs)
# plt.savefig('Pictures/Variability_initial_deformation_per_time_and_zone/
 ↪Variability_initial_deformation_per_time_and_zone_A')
# plt.close()

# fig, axs = plt.subplots(nrows=2, ncols=2,  figsize=(8,8))
```

```
# Plotting.Set_zone('B', 'regular', times,  Set_error_B, x, 0, fig, axs)
# Plotting.Set_zone('B', 'special', times,  Set_error_B_spe, x_spe, 1, fig, axs)
# plt.savefig('Pictures/Variability_initial_deformation_per_time_and_zone/
 ↪Variability_initial_deformation_per_time_and_zone_B')
# plt.close()

# fig, axs = plt.subplots(nrows=2, ncols=2,  figsize=(8,8))
# Plotting.Set_zone('C', 'regular', times,  Set_error_C, x, 0, fig, axs)
# axs[1,0].axis('off')
# axs[1,1].axis('off')
# plt.savefig('Pictures/Variability_initial_deformation_per_time_and_zone/
 ↪Variability_initial_deformation_per_time_and_zone_C')
# plt.close()

# fig, axs = plt.subplots(nrows=2, ncols=2,  figsize=(8,8))
# Plotting.Set_zone('D', 'regular', times,  Set_error_D, x, 0, fig, axs)
# Plotting.Set_zone('D', 'special', times,  Set_error_D_spe, x_spe, 1, fig, axs)
# plt.savefig('Pictures/Variability_initial_deformation_per_time_and_zone/
 ↪Variability_initial_deformation_per_time_and_zone_D')
# plt.close()

# fig, axs = plt.subplots(nrows=2, ncols=2,  figsize=(8,8))
# Plotting.Set_zone('E', 'regular', times,  Set_error_E, x, 0, fig, axs)
# Plotting.Set_zone('E', 'special', times,  Set_error_E_spe, x_spe, 1, fig, axs)
# plt.savefig('Pictures/Variability_initial_deformation_per_time_and_zone/
 ↪Variability_initial_deformation_per_time_and_zone_E')
# plt.close()
```

[35]:
```
# fig, axs = plt.subplots(8, 2*len(times), figsize = (13*len(times),6*10))

# # Plotting.errorbar_set('A', 'regular', times, Set_error_A[0],␣
 ↪Set_error_A[1], Set_error_A[2], Set_error_A[3], x, fig, axs, 0)
# # Plotting.errorbar_set('A', 'special', times, Set_error_A_spe[0],␣
 ↪Set_error_A_spe[1], Set_error_A_spe[2], Set_error_A_spe[3], x_spe, fig, axs,␣
 ↪0)
# Plotting.errorbar_set('B', 'regular', times, Set_error_B[0], Set_error_B[1],␣
 ↪Set_error_B[2], Set_error_B[3], x, fig, axs, 0)
# Plotting.errorbar_set('B', 'special', times, Set_error_B_spe[0],␣
 ↪Set_error_B_spe[1], Set_error_B_spe[2], Set_error_B_spe[3], x_spe, fig, axs,␣
 ↪0)
# Plotting.errorbar_set('C', 'regular', times, Set_error_C[0], Set_error_C[1],␣
 ↪Set_error_C[2], Set_error_C[3], x, fig, axs, 2)
# for col in range(6, 12):
#     axs[2, col].axis('off')
#     axs[3, col].axis('off')
```

```
# Plotting.errorbar_set('D', 'regular', times, Set_error_D[0], Set_error_D[1],␣
 ↪Set_error_D[2], Set_error_D[3], x, fig, axs, 4)
# Plotting.errorbar_set('D', 'special', times, Set_error_D_spe[0],␣
 ↪Set_error_D_spe[1], Set_error_D_spe[2], Set_error_D_spe[3], x_spe, fig, axs,␣
 ↪4)
# Plotting.errorbar_set('E', 'regular', times, Set_error_E[0], Set_error_E[1],␣
 ↪Set_error_E[2], Set_error_E[3], x, fig, axs, 6)
# Plotting.errorbar_set('E', 'special', times, Set_error_E_spe[0],␣
 ↪Set_error_E_spe[1], Set_error_E_spe[2], Set_error_E_spe[3], x_spe, fig, axs,␣
 ↪6)

# # # Create legends for each column
# # for col in range(2*len(times)):
# #     handles, labels = axs[0, col].get_legend_handles_labels()
# #     fig.legend(handles, labels, loc='lower center', ncol=3,␣
 ↪bbox_to_anchor=(0.5, -0.05 - col * 0.05))

# plt.subplots_adjust(hspace=0.4, wspace=0.4, bottom=0.2)

# plt.savefig('Pictures/Error_total_settlement_seperate_tunnelparts')
# # plt.close()
```

[36]:
```
x = np.arange(0,L_ele_reg+dx,dx)
x_spe = np.arange(0,L_ele_spe+dx,dx)
#[5, 10, 20, 25, 50, 100]

x_tot_A = np.concatenate([x, x + x[-1] + dx, x_spe + 2 * (x[-1] + dx), x + 2 *␣
 ↪(x[-1] + dx) + x_spe[-1] + dx, x + 3 * (x[-1] + dx) + x_spe[-1] + dx])
x_tot_B = x_tot_A
x_tot_C = np.concatenate([x, x + x[-1] + dx, x + 2 * (x[-1] + dx), x + 2 *␣
 ↪(x[-1] + dx) + x[-1] + dx, x + 3 * (x[-1] + dx) + x[-1] + dx])
x_tot_D = x_tot_A
x_tot_E = x_tot_A

Set_A_comp = np.concatenate([Set_A, Set_A, Set_A_spe, Set_A, Set_A], axis=1)
Set_B_comp = np.concatenate([Set_B, Set_B, Set_B_spe, Set_B, Set_B], axis=1)
Set_C_comp = np.concatenate([Set_C, Set_C, Set_C, Set_C, Set_C], axis=1)
Set_D_comp = np.concatenate([Set_D, Set_D, Set_D_spe, Set_D, Set_D], axis=1)
Set_E_comp = np.concatenate([Set_E, Set_E, Set_E_spe, Set_E, Set_E], axis=1)

fig, axs = plt.subplots(nrows=2, ncols=1, sharex=True, figsize=(10, 10))
Plotting.plot_settlement(axs[0], x_tot_A, Set_A_comp, times, 'A')
Plotting.plot_settlement(axs[1], x_tot_B, Set_B_comp, times, 'B')
plt.savefig('Pictures/Total_deformation_over_length_of_tunnel/
 ↪Total_deformation_over_length_of_tunnel_zone_A_B')
plt.close()
```

```
fig, axs = plt.subplots(nrows=2, ncols=1, sharex=True, figsize=(10, 10))
Plotting.plot_settlement(axs[0], x_tot_C, Set_C_comp, times, 'C')
axs[1].axis('off')
plt.savefig('Pictures/Total_deformation_over_length_of_tunnel/
 ↪Total_deformation_over_length_of_tunnel_zone_C')
plt.close()

fig, axs = plt.subplots(nrows=2, ncols=1, sharex=True, figsize=(10, 10))
Plotting.plot_settlement(axs[0], x_tot_D, Set_D_comp, times, 'D')
Plotting.plot_settlement(axs[1], x_tot_E, Set_E_comp, times, 'E')
plt.savefig('Pictures/Total_deformation_over_length_of_tunnel/
 ↪Total_deformation_over_length_of_tunnel_zone_D_E')
plt.close()
```

[37]:
```
arrays_tot = [
    Set_error_B[2],
    Set_error_B[3],
    Set_error_C[2],
    Set_error_C[3],
    Set_error_D[2],
    Set_error_D[3],
    Set_error_E[2],
    Set_error_E[3],
]


arrays_tot_spe = [
    Set_error_B_spe[2],
    Set_error_B_spe[3],
    Set_error_D_spe[2],
    Set_error_D_spe[3],
    Set_error_E_spe[2],
    Set_error_E_spe[3]
]

# Concatenate all into one long array
combined_tot = np.concatenate(arrays_tot)
combined_tot_spe = np.concatenate(arrays_tot_spe)

# Now get min and max
y_min_tot = np.min(combined_tot)
y_max_tot = np.max(combined_tot)
y_min_tot_spe = np.min(combined_tot_spe)
y_max_tot_spe = np.max(combined_tot_spe)

y_min_tot = np.min([y_min_tot, y_min_tot_spe])
```

```
y_max_tot = np.max([y_max_tot, y_max_tot_spe])
```

[38]:
```
# Plotting.Tot_error('A', x_tot_A, times, Set_error_A, Set_error_A_spe, 0, fig,␣
 ↪axs)

fig, axs = plt.subplots(nrows=4, ncols=2, figsize=(10, 15))
Plotting.Tot_error('B', x_tot_B, times, Final_B, Final_B_spe, Set_error_B,␣
 ↪Set_error_B_spe, 0, fig, axs, y_min_tot, y_max_tot)
plt.subplots_adjust(hspace=0.5, wspace=0.4, bottom=0.2)
plt.savefig('Pictures/Total_settlement_over_length_of_tunnel_including_errors/
 ↪Total_settlement_over_length_of_tunnel_including_errors_for_zone_B')
plt.close()

fig, axs = plt.subplots(nrows=4, ncols=2, figsize=(10, 15))
Plotting.Tot_error('C', x_tot_C, times, Final_C, 0, Set_error_C, 0, 0, fig,␣
 ↪axs, y_min_tot, y_max_tot)
plt.subplots_adjust(hspace=0.5, wspace=0.4, bottom=0.2)
axs[3,1].axis('off')
plt.savefig('Pictures/Total_settlement_over_length_of_tunnel_including_errors/
 ↪Total_settlement_over_length_of_tunnel_including_errors_for_zone_C')
plt.close()

fig, axs = plt.subplots(nrows=4, ncols=2, figsize=(10, 15))
Plotting.Tot_error('D', x_tot_D, times, Final_D, Final_D_spe, Set_error_D,␣
 ↪Set_error_D_spe, 0, fig, axs, y_min_tot, y_max_tot)
plt.subplots_adjust(hspace=0.5, wspace=0.4, bottom=0.2)
plt.savefig('Pictures/Total_settlement_over_length_of_tunnel_including_errors/
 ↪Total_settlement_over_length_of_tunnel_including_errors_for_zone_D')
plt.close()

fig, axs = plt.subplots(nrows=4, ncols=2, figsize=(10, 15))
Plotting.Tot_error('E', x_tot_E, times, Final_E, Final_E_spe, Set_error_E,␣
 ↪Set_error_E_spe, 0, fig, axs, y_min_tot, y_max_tot)
plt.subplots_adjust(hspace=0.5, wspace=0.4, bottom=0.2)
plt.savefig('Pictures/Total_settlement_over_length_of_tunnel_including_errors/
 ↪Total_settlement_over_length_of_tunnel_including_errors_for_zone_E')
plt.close()

print('done')
```

```
done
```

[39]:
```
print(f'regular {np.array(Set_error_E)[2][5][-1]*1000 :.1f}')
print(f'regular {np.array(Set_error_E)[3][5][-1]*1000 :.1f}')
print(f'special {np.array(Set_error_E_spe)[2][5][-1]*1000 :.1f}')
print(f'special {np.array(Set_error_E_spe)[3][5][-1]*1000 :.1f}')
```

```
regular -87.0
```

```
regular 60.3
special -91.1
special 60.4
```

J

Appendix J

The main Python script used to adjust parameters, load soil parameters and execute the other Python scripts is shown here.

Soil_profiles

May 22, 2025

[1]:
```
# pip install import_ipynb
```

[2]:
```
# pip install numpy==2.1
```

[3]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import os
import math
import matplotlib.pyplot as plt
from numba import jit
import matplotlib as mpl
from scipy.stats import truncnorm
import import_ipynb
import time
import Plotting
import pickle
import Consolidation_models
import matplotlib.lines as mlines
import matplotlib.gridspec as gridspec
from scipy.stats import norm
#import Timoshenko_beam_on_Kerr_foundation
from Regular_element_buoyancy_calculation_Fehmarn import F_res_3_reg,␣
 ↪Bal_con_req_reg
from Regular_element_buoyancy_calculation_Fehmarn import l_buo_csA as L_ele_reg
from Regular_element_buoyancy_calculation_Fehmarn import w_pl as W_ele_reg
from Special_element_buoyancy_calculation_Fehmarn import l_buo_csA as L_ele_spe
from Special_element_buoyancy_calculation_Fehmarn import w_pl as W_ele_spe
from Special_element_buoyancy_calculation_Fehmarn import F_res_3_spec,␣
 ↪Bal_con_req_spec, A_con_csA
```

```
Import Matrix was succesfull
50456.51774415001
The required force of the ballast concrete needs to be: 172487 [kN]
48456.87277274989
14782.302785250009
Import Settlement_profile done
Import Plotting was succesfull
```

[4]:
```
df = pd.read_excel('Soil_parameters.xlsx', engine='openpyxl')
name =     df.iloc[:,0]   #[-]
col =      df.iloc[:,1]   #[-]
OCR =      df.iloc[:,2]   #[-]
y_sat =    df.iloc[:,3]   #[kN/m3]
  =       df.iloc[:,4]   #[-] fluid compressibility
  ↪                          #On the contrary, if pore-fluid compressibility
  ↪is ignored,   = 1.0.
E_oed =    df.iloc[:,5]   #[kPa]
m_v =      df.iloc[:,6]   #Volumetric compressibility [kPa^-1]
e_0 =      df.iloc[:,7]   #[-]
C_e =      df.iloc[:,8]   #[-]
C_c =      df.iloc[:,9]   #[-]
C_ae =     df.iloc[:,10]  #[-]
c_v =      df.iloc[:,11]  #[m2/day] Coefficient of consolidation

e_0 = [float(x) for x in e_0]
C_ae = [float(x) for x in C_ae]
C_c = [float(x) for x in C_c]

#Soil = [name, color, OCR, y_sat,  , m_v, e_0, C_e, C_c, C_ae, c_v]
Soil_0 = df.iloc[0]
Soil_1 = df.iloc[1]
Soil_2 = df.iloc[2]
Soil_3 = df.iloc[3]
Soil_4 = df.iloc[4]
Soil_5 = df.iloc[5]
Soil_6 = df.iloc[6]
Soil_7 = df.iloc[7]

#Height elements
h_reg = 8.900
h_spe = 13.455

# Unit weight
gamma_foun = 21
  ↪            #[kN/m3] Foundation (from Fu and Song 2021) #Thickness changes
  ↪will only add a load and have no significant effect on total creep/
  ↪consolidation
gamma_w = 9.81
  ↪            #[kN/m3] Water
gamma_bf = 18
  ↪            #[kN/m3] Backfill (from Fu and Song 2021)
gamma_ele = 9.81
  ↪            #[kN/m3] Element Same weight as water in the beginning
```

```
gamma_prot = 26             #[kN/m3] Protection layer (protection layer needs a researched value)

#Thicknesses
t_foun = 1                  #[m] thickness of foundation layer (Monte Carlo needed in the end)
t_prot = 1.5                #[m] thickness of protection layer (Can be changed afterwards to check for influence)
t_fill = 1                  #[m] thickness of filling material on top of protection layer/ tunnel element

# Parameters for signal calculation
M = 600                     #Amount of time fourier series is added to total value, make this value higher if the graph produced is not smooth enough
N = 50
K = 30

#Definition of times used in calculation
t_0 = 1                     #[days] reference days for creep calculation
t_end = 2190                #[days]
sub_t = 10
t_u_prof = [0.5*t_end, 0.75*t_end, 0.99*t_end]  #[days] visualised u_prof for these times further down
time_days = np.arange(0, t_end+sub_t, sub_t)    #[days] time for the total project to take place and up untill what age we want to now the secondary settlement

t_c_un =  3*365             #[days] 1.Unloading/trenching of soil
t_c_foun = 50               #[days] 2.Deposition of foundation layer
t_c_tun = 1                 #[days] 3.Installation of tunnel elements
t_c_prot_d = 50             #[days] 4.Placement of protection layer
t_c_rel_d = 50              #[days] 5.Reloading soil with soil that was removed
```

```
#6.Back siltation from (Wang et al., 2023b) and tidal load from (Shoa and Li,␣
 ↪2003)
#Cyclic degradation is disregarded in this research

t_c = [t_c_un, t_c_foun, t_c_tun]                                              ␣
 ↪            #[days] time of the construction period of that loading stage
t_c_rel = [t_c_un, t_c_foun, t_c_tun, t_c_rel_d]                               ␣
 ↪            #[days] time of the construction period of that loading stage␣
 ↪including the reloading
t_c_prot = [t_c_un, t_c_foun, t_c_tun, t_c_prot_d]                             ␣
 ↪            #[days] time of the construction period of that loading stage␣
 ↪including the protection layer
t_c_rel_prot = [t_c_un, t_c_foun, t_c_tun, t_c_prot_d, t_c_rel_d]              ␣
 ↪            #[days] time of the construction period of that loading stage␣
 ↪including the protection layer and reloading

start_times = [0, 1150, 1300]                                                  ␣
 ↪            #[days]
start_times_rel = [0, 1150, 1300, 1600]                                        ␣
 ↪            #[days]
start_times_prot = [0, 1150, 1300, 1450]                                       ␣
 ↪            #[days]
start_times_rel_prot = [0, 1150, 1300, 1450, 1600]                             ␣
 ↪            #[days]

# Creep parameters
t_end_EOP = 50000
alfa = 0.5
t = np.arange(0,100*365,10)
times = np.array([5, 10, 20, 25, 50, 100])*365

closest_locations = [np.abs(t - value).argmin() for value in times]

#Monte Carlo parameters
std_dev = 2 # Standard deviation
num_simulations = 500  # Number of Monte Carlo simulations

#TK-model space dimensions
n_ele = 2                                                                      ␣
 ↪            # [-]
load_style = "Uniform"                                                         ␣
 ↪            # [-]  (choose between Concentrated and Uniform)
a_u = 0
b_u = 1
dx = 0.1
```

```
#TK-model parameters of concrete beam
kappa = 0.833                                                                  ␣
 ↪           # [-]
E_b = 2.9*10**7                                                                ␣
 ↪           # [kPa]
v_b = 0.2                                                                      ␣
 ↪           # [-]
```

1 Zones

[5]:
```
#Zone A
Parameters_A = [Soil_0, Soil_5]
t_l_A =       [9.625, 80.475]
t_l_A_spe =   [9.625, 75.92]
t_l_A_u =     [1.563, 11.463]
t_l_A_u_spe = [1.563, 16.018]
y_sat_A_u =   [y_sat[7]-gamma_w, y_sat[4]-gamma_w]
q_rel_A =     1.563 * (y_sat[7]-gamma_w) + 1.563 * (y_sat[4]-gamma_w)
bou_con_A =   ["upper"]
d_sc_A =      np.sum(np.delete(t_l_A, [0]))
d_sc_A_spe =  np.sum(np.delete(t_l_A_spe, [0]))

# Zone B
Parameters_B = [Soil_0, Soil_1, Soil_2, Soil_3, Soil_4]
t_l_B =       [34.625, 3.6625, 14.0625, 6.5, 32.25]
t_l_B_spe =   [34.625, 0.5, 13.17, 6.5, 32.25]
t_l_B_u =     [11.4]
t_l_B_u_spe = [15.955]
y_sat_B_u =   [y_sat[1]-gamma_w]
q_rel_B =     0
bou_con_B =   ["upper", "open", "closed", "upper"]
d_sc_B =      np.sum(np.delete(t_l_B, [0]))
d_sc_B_spe =  np.sum(np.delete(t_l_B_spe, [0]))

# Zone C
Parameters_C = [Soil_0, Soil_2, Soil_4, Soil_6]
t_l_C =      [34.375, 0.625, 12.5, 43.6]
t_l_C_u =    [14.525]
y_sat_C_u =  [y_sat[1]-gamma_w]
q_rel_C =    3.125 * (y_sat[1]-gamma_w)
bou_con_C =  ["upper", "upper", "closed"]
d_sc_C =     np.sum(np.delete(t_l_C, [0]))

# Zone D
Parameters_D = [Soil_0, Soil_2, Soil_4, Soil_5]
t_l_D =       [21.875, 27.125, 28.0625, 14.0375]
```

```
t_l_D_spe =   [21.875, 22.57, 28.0625, 14.0375]
t_l_D_u =     [6.25, 9.9]
t_l_D_u_spe = [6.25, 14.455]
y_sat_D_u =   [y_sat[7]-gamma_w, y_sat[2]-gamma_w]
q_rel_D =      6.25 * (y_sat[7]-gamma_w)
bou_con_D =   ["upper", "upper", "upper"]
d_sc_D =      np.sum(np.delete(t_l_D, [0]))
d_sc_D_spe =  np.sum(np.delete(t_l_D_spe, [0]))

# Zone E
Parameters_E = [Soil_0, Soil_3, Soil_2, Soil_4, Soil_5]
t_l_E =       [12.5, 2.125, 18.75, 21.875, 35.85]
t_l_E_spe =   [12.5, 0.5, 16.32, 21.875, 35.85]
t_l_E_u =     [15.4, 1.0]
t_l_E_u_spe = [15.15, 5.555]
y_sat_E_u =   [y_sat[2]-gamma_w, y_sat[3]-gamma_w]
q_rel_E =     6.25 * (y_sat[2]-gamma_w)
bou_con_E =   ["closed","upper","upper","upper"]
d_sc_E =      np.sum(np.delete(t_l_E, [0]))
d_sc_E_spe =  np.sum(np.delete(t_l_E_spe, [0]))

Zone_lay = ['A', 'B', 'C', 'D', 'E']
Zone_lay_spe = ['A', 'B', 'D','E']
```

[6]:
```
# start = time.perf_counter()
# Zone_A    = Consolidation_models.final_config('A', "Uniform",  Parameters_A,␣
 ↪'regular', h_reg, t_l_A, d_sc_A, q_rel_A, F_res_3_reg, Bal_con_req_reg,␣
 ↪y_sat_A_u, t_l_A_u, gamma_foun, gamma_w, gamma_prot, t_foun, t_foun, t_prot,␣
 ↪t_u_prof, bou_con_A, M, N, K, t_c_rel, time_days, start_times_rel,␣
 ↪t_end_EOP, sub_t, t_end, t_0, t, alfa, std_dev, num_simulations, kappa, E_b,␣
 ↪v_b, W_ele_reg, L_ele_reg, a_u, b_u, dx, times)
# Zone_A_spe  = Consolidation_models.final_config('A', "Uniform", ␣
 ↪Parameters_A, 'special', h_spe, t_l_A_spe, d_sc_A_spe, q_rel_A,␣
 ↪F_res_3_spec, Bal_con_req_spec, y_sat_A_u, t_l_A_u_spe, gamma_foun, gamma_w,␣
 ↪gamma_prot, t_foun, t_foun, t_prot, t_u_prof, bou_con_A, M, N, K, t_c_rel,␣
 ↪time_days, start_times_rel, t_end_EOP, sub_t, t_end, t_0, t, alfa, std_dev,␣
 ↪num_simulations, kappa, E_b, v_b, W_ele_spe, L_ele_spe, a_u, b_u, dx, times)
# end = time.perf_counter()

# print(f'Time it takes for Zone A- calculation: {(end-start):.2f} sec or␣
 ↪{(end-start)/60:.2f} minutes')
```

[7]:
```
# start = time.perf_counter()
```

```
# Zone_B = Consolidation_models.final_config('B', "Uniform",  Parameters_B,␣
 ↪'regular', h_reg, t_l_B, d_sc_B, q_rel_B, F_res_3_reg, Bal_con_req_reg,␣
 ↪y_sat_B_u, t_l_B_u, gamma_foun, gamma_w, gamma_prot, t_foun+t_prot, t_foun,␣
 ↪t_prot, t_u_prof, bou_con_B, M, N, K, t_c_prot, time_days, start_times_prot,␣
 ↪t_end_EOP, sub_t, t_end, t_0, t, alfa, std_dev, num_simulations, kappa, E_b,␣
 ↪v_b, W_ele_reg, L_ele_reg, a_u, b_u, dx, times)
# Zone_B_spe = Consolidation_models.final_config('B', "Uniform",  Parameters_B,␣
 ↪'special', h_spe, t_l_B_spe, d_sc_B_spe, q_rel_B, F_res_3_spec,␣
 ↪Bal_con_req_spec, y_sat_B_u, t_l_B_u_spe, gamma_foun, gamma_w, gamma_prot,␣
 ↪t_foun+t_prot, t_foun, t_prot, t_u_prof, bou_con_B, M, N, K, t_c_prot,␣
 ↪time_days, start_times_prot, t_end_EOP, sub_t, t_end, t_0, t, alfa, std_dev,␣
 ↪num_simulations, kappa, E_b, v_b, W_ele_spe, L_ele_spe, a_u, b_u, dx, times)
# end = time.perf_counter()

# print(f'Time it takes for Zone B- calculation: {(end-start):.2f} sec or␣
 ↪{(end-start)/60:.2f} minutes')
```

[8]:
```
# start = time.perf_counter()
# Zone_C = Consolidation_models.final_config('C', "Uniform",  Parameters_C,␣
 ↪'regular', h_reg, t_l_C, d_sc_C, q_rel_C, F_res_3_reg, Bal_con_req_reg,␣
 ↪y_sat_C_u, t_l_C_u, gamma_foun, gamma_w, gamma_prot, t_foun+t_prot, t_foun,␣
 ↪t_prot, t_u_prof, bou_con_C, M, N, K, t_c_rel_prot, time_days,␣
 ↪start_times_rel_prot, t_end_EOP, sub_t, t_end, t_0, t, alfa, std_dev,␣
 ↪num_simulations, kappa, E_b, v_b, W_ele_reg, L_ele_reg, a_u, b_u, dx, times)
# end = time.perf_counter()

# print(f'Time it takes for Zone C- calculation: {(end-start):.2f} sec or␣
 ↪{(end-start)/60:.2f} minutes')
```

[9]:
```
# start = time.perf_counter()
# Zone_D = Consolidation_models.final_config('D', "Uniform",  Parameters_D,␣
 ↪'regular', h_reg, t_l_D, d_sc_D, q_rel_D, F_res_3_reg, Bal_con_req_reg,␣
 ↪y_sat_D_u, t_l_D_u, gamma_foun, gamma_w, gamma_prot, t_foun, t_foun, t_prot,␣
 ↪t_u_prof, bou_con_D, M, N, K, t_c_rel, time_days, start_times_rel,␣
 ↪t_end_EOP, sub_t, t_end, t_0, t, alfa, std_dev, num_simulations, kappa, E_b,␣
 ↪v_b, W_ele_reg, L_ele_reg, a_u, b_u, dx, times)
# Zone_D_spe  = Consolidation_models.final_config('D', "Uniform", ␣
 ↪Parameters_D, 'special', h_spe, t_l_D_spe, d_sc_D_spe, q_rel_D,␣
 ↪F_res_3_spec, Bal_con_req_spec, y_sat_D_u, t_l_D_u_spe, gamma_foun, gamma_w,␣
 ↪gamma_prot, t_foun, t_foun, t_prot, t_u_prof, bou_con_D, M, N, K, t_c_rel,␣
 ↪time_days, start_times_rel, t_end_EOP, sub_t, t_end, t_0, t, alfa, std_dev,␣
 ↪num_simulations, kappa, E_b, v_b, W_ele_spe, L_ele_spe, a_u, b_u, dx, times)
# end = time.perf_counter()

# print(f'Time it takes for Zone D- calculation: {(end-start):.2f} sec or␣
 ↪{(end-start)/60:.2f} minutes')
```

[10]:
```
# start = time.perf_counter()
# Zone_E = Consolidation_models.final_config('E', "Uniform",  Parameters_E,␣
 ↪'regular', h_reg, t_l_E, d_sc_E, q_rel_E, F_res_3_reg, Bal_con_req_reg,␣
 ↪y_sat_E_u, t_l_E_u, gamma_foun, gamma_w, gamma_prot, t_foun, t_foun, t_prot,␣
 ↪t_u_prof, bou_con_E, M, N, K, t_c_rel, time_days, start_times_rel,␣
 ↪t_end_EOP, sub_t, t_end, t_0, t, alfa, std_dev, num_simulations, kappa, E_b,␣
 ↪v_b, W_ele_reg, L_ele_reg, a_u, b_u, dx, times)
# Zone_E_spe = Consolidation_models.final_config('E', "Uniform",  Parameters_E,␣
 ↪'special', h_spe, t_l_E_spe, d_sc_E_spe, q_rel_E, F_res_3_spec,␣
 ↪Bal_con_req_spec, y_sat_E_u, t_l_E_u_spe, gamma_foun, gamma_w, gamma_prot,␣
 ↪t_foun, t_foun, t_prot, t_u_prof, bou_con_E, M, N, K, t_c_rel, time_days,␣
 ↪start_times_rel, t_end_EOP, sub_t, t_end, t_0, t, alfa, std_dev,␣
 ↪num_simulations, kappa, E_b, v_b, W_ele_spe, L_ele_spe, a_u, b_u, dx, times)
# end = time.perf_counter()

# print(f'Time it takes for Zone E- calculation: {(end-start):.2f} sec or␣
 ↪{(end-start)/60:.2f} minutes')
```

[11]:
```
start = time.perf_counter()
Zone_A, Prim_error_A, Sec_error_A, Tot_lay_error_A, Zone_prim_error_A,␣
 ↪Zone_sec_error_A, Zone_tot_error_A, Ini_set_A, Ini_error_A, Set_A,␣
 ↪Set_error_A, Prim_set_A, Sec_set_A, Tot_lay_set_A, Tot_prim_set_A,␣
 ↪Tot_sec_set_A, Tot_zone_set_A, Final_A = Consolidation_models.
 ↪final_config_error(f'simulation_data_A_regular_with_{num_simulations}_simulations_and_dev_{
 ↪pkl')
Zone_A_spe, Prim_error_A_spe, Sec_error_A_spe, Tot_lay_error_A_spe,␣
 ↪Zone_prim_error_A_spe, Zone_sec_error_A_spe, Zone_tot_error_A_spe,␣
 ↪Ini_set_A_spe, Ini_error_A_spe, Set_A_spe, Set_error_A_spe, Prim_set_A_spe,␣
 ↪Sec_set_A_spe, Tot_lay_set_A_spe, Tot_prim_set_A_spe, Tot_sec_set_A_spe,␣
 ↪Tot_zone_set_A_spe, Final_A_spe = Consolidation_models.
 ↪final_config_error(f'simulation_data_A_special_with_{num_simulations}_simulations_and_dev_{
 ↪pkl')
Zone_B, Prim_error_B, Sec_error_B, Tot_lay_error_B, Zone_prim_error_B,␣
 ↪Zone_sec_error_B, Zone_tot_error_B, Ini_set_B, Ini_error_B, Set_B,␣
 ↪Set_error_B, Prim_set_B, Sec_set_B, Tot_lay_set_B, Tot_prim_set_B,␣
 ↪Tot_sec_set_B, Tot_zone_set_B, Final_B = Consolidation_models.
 ↪final_config_error(f'simulation_data_B_regular_with_{num_simulations}_simulations_and_dev_{
 ↪pkl')
Zone_B_spe, Prim_error_B_spe, Sec_error_B_spe, Tot_lay_error_B_spe,␣
 ↪Zone_prim_error_B_spe, Zone_sec_error_B_spe, Zone_tot_error_B_spe,␣
 ↪Ini_set_B_spe, Ini_error_B_spe, Set_B_spe, Set_error_B_spe, Prim_set_B_spe,␣
 ↪Sec_set_B_spe, Tot_lay_set_B_spe, Tot_prim_set_B_spe, Tot_sec_set_B_spe,␣
 ↪Tot_zone_set_B_spe, Final_B_spe = Consolidation_models.
 ↪final_config_error(f'simulation_data_B_special_with_{num_simulations}_simulations_and_dev_{
 ↪pkl')
```

```
Zone_C, Prim_error_C, Sec_error_C, Tot_lay_error_C, Zone_prim_error_C,␣
 ↪Zone_sec_error_C, Zone_tot_error_C, Ini_set_C, Ini_error_C, Set_C,␣
 ↪Set_error_C, Prim_set_C, Sec_set_C, Tot_lay_set_C, Tot_prim_set_C,␣
 ↪Tot_sec_set_C, Tot_zone_set_C, Final_C = Consolidation_models.
 ↪final_config_error(f'simulation_data_C_regular_with_{num_simulations}_simulations_and_dev_{
 ↪pkl')
Zone_D, Prim_error_D, Sec_error_D, Tot_lay_error_D, Zone_prim_error_D,␣
 ↪Zone_sec_error_D, Zone_tot_error_D, Ini_set_D, Ini_error_D, Set_D,␣
 ↪Set_error_D, Prim_set_D, Sec_set_D, Tot_lay_set_D, Tot_prim_set_D,␣
 ↪Tot_sec_set_D, Tot_zone_set_D, Final_D  = Consolidation_models.
 ↪final_config_error(f'simulation_data_D_regular_with_{num_simulations}_simulations_and_dev_{
 ↪pkl')
Zone_D_spe, Prim_error_D_spe, Sec_error_D_spe, Tot_lay_error_D_spe,␣
 ↪Zone_prim_error_D_spe, Zone_sec_error_D_spe, Zone_tot_error_D_spe,␣
 ↪Ini_set_D_spe, Ini_error_D_spe, Set_D_spe, Set_error_D_spe, Prim_set_D_spe,␣
 ↪Sec_set_D_spe, Tot_lay_set_D_spe, Tot_prim_set_D_spe, Tot_sec_set_D_spe,␣
 ↪Tot_zone_set_D_spe, Final_D_spe = Consolidation_models.
 ↪final_config_error(f'simulation_data_D_special_with_{num_simulations}_simulations_and_dev_{
 ↪pkl')
Zone_E, Prim_error_E, Sec_error_E, Tot_lay_error_E, Zone_prim_error_E,␣
 ↪Zone_sec_error_E, Zone_tot_error_E, Ini_set_E, Ini_error_E, Set_E,␣
 ↪Set_error_E, Prim_set_E, Sec_set_E, Tot_lay_set_E, Tot_prim_set_E,␣
 ↪Tot_sec_set_E, Tot_zone_set_E, Final_E  = Consolidation_models.
 ↪final_config_error(f'simulation_data_E_regular_with_{num_simulations}_simulations_and_dev_{
 ↪pkl')
Zone_E_spe, Prim_error_E_spe, Sec_error_E_spe, Tot_lay_error_E_spe,␣
 ↪Zone_prim_error_E_spe, Zone_sec_error_E_spe, Zone_tot_error_E_spe,␣
 ↪Ini_set_E_spe, Ini_error_E_spe, Set_E_spe, Set_error_E_spe, Prim_set_E_spe,␣
 ↪Sec_set_E_spe, Tot_lay_set_E_spe, Tot_prim_set_E_spe, Tot_sec_set_E_spe,␣
 ↪Tot_zone_set_E_spe, Final_E_spe =  Consolidation_models.
 ↪final_config_error(f'simulation_data_E_special_with_{num_simulations}_simulations_and_dev_{
 ↪pkl')
end = time.perf_counter()

print(f'Time it takes for error- calculation: {(end-start):.2f} sec or␣
 ↪{(end-start)/60:.2f} minutes')
```

```
Time it takes for error- calculation: 7.13 sec or 0.12 minutes
```

[12]:
```
superimposed_settlements = [np.sum(Zone_A[2], axis = 0), np.sum(Zone_B[2], axis␣
 ↪= 0), np.sum(Zone_C[2], axis = 0), np.sum(Zone_D[2], axis = 0), np.
 ↪sum(Zone_E[2], axis = 0)]
superimposed_settlements_spe = [np.sum(Zone_A_spe[2], axis = 0), np.
 ↪sum(Zone_B_spe[2], axis = 0), np.sum(Zone_D_spe[2], axis = 0), np.
 ↪sum(Zone_E_spe[2], axis = 0)]
```

```
Cre_values = [np.sum(Zone_A[3], axis = 0), np.sum(Zone_B[3], axis = 0), np.
 ↪sum(Zone_C[3], axis = 0), np.sum(Zone_D[3], axis = 0), np.sum(Zone_E[3],␣
 ↪axis = 0)]
Cre_values_spe = [np.sum(Zone_A_spe[3], axis = 0), np.sum(Zone_B_spe[3], axis =␣
 ↪0), np.sum(Zone_D_spe[3], axis = 0), np.sum(Zone_E_spe[3], axis = 0)]

Tot_prim_set = [Tot_prim_set_A, Tot_prim_set_B, Tot_prim_set_C, Tot_prim_set_D,␣
 ↪Tot_prim_set_E]
prim_mean_superimposed_settlements = [Zone_prim_error_A[0],␣
 ↪Zone_prim_error_B[0], Zone_prim_error_C[0], Zone_prim_error_D[0],␣
 ↪Zone_prim_error_E[0]]
prim_std_superimposed_settlements = [Zone_prim_error_A[1],␣
 ↪Zone_prim_error_B[1], Zone_prim_error_C[1], Zone_prim_error_D[1],␣
 ↪Zone_prim_error_E[1]]
prim_fi_perc_superimposed_settlements = [Zone_prim_error_A[2],␣
 ↪Zone_prim_error_B[2], Zone_prim_error_C[2], Zone_prim_error_D[2],␣
 ↪Zone_prim_error_E[2]]
prim_la_perc_superimposed_settlements = [Zone_prim_error_A[3],␣
 ↪Zone_prim_error_B[3], Zone_prim_error_C[3], Zone_prim_error_D[3],␣
 ↪Zone_prim_error_E[3]]

Tot_prim_set_spe = [Tot_prim_set_A_spe, Tot_prim_set_B_spe, Tot_prim_set_D_spe,␣
 ↪Tot_prim_set_E_spe]
prim_mean_superimposed_settlements_spe = [Zone_prim_error_A_spe[0],␣
 ↪Zone_prim_error_B_spe[0], Zone_prim_error_D_spe[0], Zone_prim_error_E_spe[0]]
prim_std_superimposed_settlements_spe = [Zone_prim_error_A_spe[1],␣
 ↪Zone_prim_error_B_spe[1], Zone_prim_error_D_spe[1], Zone_prim_error_E_spe[1]]
prim_fi_perc_superimposed_settlements_spe = [Zone_prim_error_A_spe[2],␣
 ↪Zone_prim_error_B_spe[2], Zone_prim_error_D_spe[2], Zone_prim_error_E_spe[2]]
prim_la_perc_superimposed_settlements_spe = [Zone_prim_error_A_spe[3],␣
 ↪Zone_prim_error_B_spe[3], Zone_prim_error_D_spe[3], Zone_prim_error_E_spe[3]]

Tot_sec_set = [Tot_sec_set_A, Tot_sec_set_B, Tot_sec_set_C, Tot_sec_set_D,␣
 ↪Tot_sec_set_E]
sec_mean_superimposed_settlements = [Zone_sec_error_A[0], Zone_sec_error_B[0],␣
 ↪Zone_sec_error_C[0], Zone_sec_error_D[0], Zone_sec_error_E[0]]
sec_std_superimposed_settlements = [Zone_sec_error_A[1], Zone_sec_error_B[1],␣
 ↪Zone_sec_error_C[1], Zone_sec_error_D[1], Zone_sec_error_E[1]]
sec_fi_perc_superimposed_settlements = [Zone_sec_error_A[2],␣
 ↪Zone_sec_error_B[2], Zone_sec_error_C[2], Zone_sec_error_D[2],␣
 ↪Zone_sec_error_E[2]]
sec_la_perc_superimposed_settlements = [Zone_sec_error_A[3],␣
 ↪Zone_sec_error_B[3], Zone_sec_error_C[3], Zone_sec_error_D[3],␣
 ↪Zone_sec_error_E[3]]
```

```
Tot_sec_set_spe = [Tot_sec_set_A_spe, Tot_sec_set_B_spe, Tot_sec_set_D_spe,␣
 ↪Tot_sec_set_E_spe]
sec_mean_superimposed_settlements_spe = [Zone_sec_error_A_spe[0],␣
 ↪Zone_sec_error_B_spe[0], Zone_sec_error_D_spe[0], Zone_sec_error_E_spe[0]]
sec_std_superimposed_settlements_spe = [Zone_sec_error_A_spe[1],␣
 ↪Zone_sec_error_B_spe[1], Zone_sec_error_D_spe[1], Zone_sec_error_E_spe[1]]
sec_fi_perc_superimposed_settlements_spe = [Zone_sec_error_A_spe[2],␣
 ↪Zone_sec_error_B_spe[2], Zone_sec_error_D_spe[2], Zone_sec_error_E_spe[2]]
sec_la_perc_superimposed_settlements_spe = [Zone_sec_error_A_spe[3],␣
 ↪Zone_sec_error_B_spe[3], Zone_sec_error_D_spe[3], Zone_sec_error_E_spe[3]]

Tot_set_zone = [Tot_zone_set_A, Tot_zone_set_B, Tot_zone_set_C, Tot_zone_set_D,␣
 ↪Tot_zone_set_E]
Tot_mean_superimposed_settlements =[Zone_tot_error_A[0], Zone_tot_error_B[0],␣
 ↪Zone_tot_error_C[0], Zone_tot_error_D[0], Zone_tot_error_E[0]]
Tot_std_superimposed_settlements = [Zone_tot_error_A[1], Zone_tot_error_B[1],␣
 ↪Zone_tot_error_C[1], Zone_tot_error_D[1], Zone_tot_error_E[1]]
Tot_fi_perc_superimposed_settlements =[Zone_tot_error_A[2],␣
 ↪Zone_tot_error_B[2], Zone_tot_error_C[2], Zone_tot_error_D[2],␣
 ↪Zone_tot_error_E[2]]
Tot_la_perc_superimposed_settlements = [Zone_tot_error_A[3],␣
 ↪Zone_tot_error_B[3], Zone_tot_error_C[3], Zone_tot_error_D[3],␣
 ↪Zone_tot_error_E[3]]

Tot_set_zone_spe = [Tot_zone_set_A_spe, Tot_zone_set_B_spe, Tot_zone_set_D_spe,␣
 ↪Tot_zone_set_E_spe]
Tot_mean_superimposed_settlements_spe =[Zone_tot_error_A_spe[0],␣
 ↪Zone_tot_error_B_spe[0], Zone_tot_error_D_spe[0], Zone_tot_error_E_spe[0]]
Tot_std_superimposed_settlements_spe = [Zone_tot_error_A_spe[1],␣
 ↪Zone_tot_error_B_spe[1], Zone_tot_error_D_spe[1], Zone_tot_error_E_spe[1]]
Tot_fi_perc_superimposed_settlements_spe =[Zone_tot_error_A_spe[2],␣
 ↪Zone_tot_error_B_spe[2], Zone_tot_error_D_spe[2], Zone_tot_error_E_spe[2]]
Tot_la_perc_superimposed_settlements_spe = [Zone_tot_error_A_spe[3],␣
 ↪Zone_tot_error_B_spe[3], Zone_tot_error_D_spe[3], Zone_tot_error_E_spe[3]]

common_times = np.intersect1d(time_days, t)
indices_time_days = np.where(np.isin(time_days, common_times))[0]
indices_t = np.where(np.isin(t, common_times))[0]

Cre_values = np.array(Cre_values)
Cre_values_spe = np.array(Cre_values_spe)
superimposed_settlements = np.array(superimposed_settlements)
superimposed_settlements_spe = np.array(superimposed_settlements_spe)
```

```
Tot_set = np.append(Cre_values[:,indices_t] + superimposed_settlements[:
 ↪,indices_time_days], np.append(superimposed_settlements[:,-2].reshape(-1,␣
 ↪1),superimposed_settlements[:,-1].reshape(-1, 1), axis = 1), axis = 1)
Tot_set = np.append(Tot_set, Cre_values[:,(indices_t[-1])+2:-1], axis = 1)
Tot_set_spe = np.append(Cre_values_spe[:,indices_t] +␣
 ↪superimposed_settlements_spe[:,indices_time_days], np.
 ↪append(superimposed_settlements_spe[:,-2].reshape(-1,␣
 ↪1),superimposed_settlements_spe[:,-1].reshape(-1, 1), axis = 1), axis = 1)
Tot_set_spe = np.append(Tot_set_spe, Cre_values_spe[:,(indices_t[-1])+2:-1],␣
 ↪axis = 1)
```

[13]:
```
Tot_set_ini_A, y_q_A, u_r_q_A, F_ex_q_A, x = Consolidation_models.
 ↪timo_kerr_DEM("Uniform", Parameters_A, t_l_A, kappa, E_b, v_b, W_ele_reg,␣
 ↪h_reg, np.sum(d_sc_A), L_ele_reg, a_u, b_u, Zone_A[7], dx)
Tot_set_ini_A_spe, y_q_A_spe, u_r_q_A_spe, F_ex_q_A_spe, x_spe =␣
 ↪Consolidation_models.timo_kerr_DEM("Uniform", Parameters_A, t_l_A_spe,␣
 ↪kappa, E_b, v_b, W_ele_spe, h_spe, np.sum(d_sc_A_spe), L_ele_spe, a_u, b_u,␣
 ↪Zone_A_spe[7], dx)
Tot_set_ini_B, y_q_B, u_r_q_B, F_ex_q_B, x = Consolidation_models.
 ↪timo_kerr_DEM("Uniform", Parameters_B, t_l_B, kappa, E_b, v_b, W_ele_reg,␣
 ↪h_reg, np.sum(d_sc_B), L_ele_reg, a_u, b_u, Zone_B[7], dx)
Tot_set_ini_B_spe, y_q_B_spe, u_r_q_B_spe, F_ex_q_B_spe, x_spe  =␣
 ↪Consolidation_models.timo_kerr_DEM("Uniform", Parameters_B, t_l_B_spe,␣
 ↪kappa, E_b, v_b, W_ele_spe, h_spe, np.sum(d_sc_B_spe), L_ele_spe, a_u, b_u,␣
 ↪Zone_B_spe[7], dx)
Tot_set_ini_C, y_q_C, u_r_q_C, F_ex_q_C, x = Consolidation_models.
 ↪timo_kerr_DEM("Uniform", Parameters_C, t_l_C,  kappa, E_b, v_b, W_ele_reg,␣
 ↪h_reg, np.sum(d_sc_C), L_ele_reg, a_u, b_u, Zone_C[7], dx)
Tot_set_ini_D, y_q_D, u_r_q_D, F_ex_q_D, x = Consolidation_models.
 ↪timo_kerr_DEM("Uniform", Parameters_D, t_l_D,  kappa, E_b, v_b, W_ele_reg,␣
 ↪h_reg, np.sum(d_sc_D), L_ele_reg, a_u, b_u, Zone_D[7], dx)
Tot_set_ini_D_spe, y_q_D_spe, u_r_q_D_spe, F_ex_q_D_spe, x_spe =␣
 ↪Consolidation_models.timo_kerr_DEM("Uniform", Parameters_D, t_l_D_spe, ␣
 ↪kappa, E_b, v_b, W_ele_spe, h_spe, np.sum(d_sc_D_spe), L_ele_spe, a_u, b_u,␣
 ↪Zone_D_spe[7], dx)
Tot_set_ini_E, y_q_E, u_r_q_E, F_ex_q_E, x = Consolidation_models.
 ↪timo_kerr_DEM("Uniform", Parameters_E, t_l_E, kappa, E_b, v_b, W_ele_reg,␣
 ↪h_reg, np.sum(d_sc_E), L_ele_reg, a_u, b_u, Zone_E[7], dx)
Tot_set_ini_E_spe, y_q_E_spe, u_r_q_E_spe, F_ex_q_E_spe, x_spe =␣
 ↪Consolidation_models.timo_kerr_DEM("Uniform", Parameters_E, t_l_E_spe, ␣
 ↪kappa, E_b, v_b, W_ele_spe, h_spe, np.sum(d_sc_E_spe), L_ele_spe, a_u, b_u,␣
 ↪Zone_E_spe[7], dx)
```

[14]:
```
arrays_prim_con = [
    Zone_A[2],
    Zone_A_spe[2],
```

```
    Zone_B[2],
    Zone_B_spe[2],
    Zone_C[2],
    Zone_D[2],
    Zone_D_spe[2],
    Zone_E[2],
    Zone_E_spe[2],
]

# Concatenate all into one long array
combined_prim_con = np.concatenate(arrays_prim_con)

# Now get min and max
y_min_prim_con = np.min(combined_prim_con)
y_max_prim_con = np.max(combined_prim_con)
```

[15]:
```
fig, axs = plt.subplots(2, 1, figsize=(6,5), constrained_layout = True)
Plotting.primary_consolidation('A', 'regular', Zone_A[4], Zone_A[5], time_days,␣
 ↪Zone_A[2], 0, axs, fig, y_min_prim_con, y_max_prim_con)
Plotting.primary_consolidation('A', 'special', Zone_A_spe[4], Zone_A_spe[5],␣
 ↪time_days, Zone_A_spe[2], 1, axs, fig, y_min_prim_con, y_max_prim_con)
plt.savefig(f'Pictures/Primary_Consolidation_of_separated_layers/
 ↪Primary_Consolidation_of_separated_layers_Zone_A')
plt.close()

fig, axs = plt.subplots(2, 1, figsize=(6,5), constrained_layout = True)
Plotting.primary_consolidation('B', 'regular', Zone_B[4], Zone_B[5], time_days,␣
 ↪Zone_B[2], 0, axs, fig, y_min_prim_con, y_max_prim_con)
Plotting.primary_consolidation('B', 'special', Zone_B_spe[4], Zone_B_spe[5],␣
 ↪time_days, Zone_B_spe[2], 1, axs, fig, y_min_prim_con, y_max_prim_con)
plt.savefig(f'Pictures/Primary_Consolidation_of_separated_layers/
 ↪Primary_Consolidation_of_separated_layers_Zone_B')
plt.close()

fig, axs = plt.subplots(2, 1, figsize=(6,5), constrained_layout = True)
Plotting.primary_consolidation('C', 'regular', Zone_C[4], Zone_C[5], time_days,␣
 ↪Zone_C[2], 0, axs, fig, y_min_prim_con, y_max_prim_con)
axs[1].axis('off')
plt.savefig(f'Pictures/Primary_Consolidation_of_separated_layers/
 ↪Primary_Consolidation_of_separated_layers_Zone_C')
plt.close()

fig, axs = plt.subplots(2, 1, figsize=(6,5), constrained_layout = True)
Plotting.primary_consolidation('D', 'regular', Zone_D[4], Zone_D[5], time_days,␣
 ↪Zone_D[2], 0, axs, fig, y_min_prim_con, y_max_prim_con)
```

```
Plotting.primary_consolidation('D', 'special', Zone_D_spe[4], Zone_D_spe[5],␣
 ↪time_days, Zone_D_spe[2], 1, axs, fig, y_min_prim_con, y_max_prim_con)
plt.savefig(f'Pictures/Primary_Consolidation_of_separated_layers/
 ↪Primary_Consolidation_of_separated_layers_Zone_D')
plt.close()

fig, axs = plt.subplots(2, 1, figsize=(6,5), constrained_layout = True)
Plotting.primary_consolidation('E', 'regular', Zone_E[4], Zone_E[5], time_days,␣
 ↪Zone_E[2], 0, axs, fig, y_min_prim_con, y_max_prim_con)
Plotting.primary_consolidation('E', 'special', Zone_E_spe[4], Zone_E_spe[5],␣
 ↪time_days, Zone_E_spe[2], 1,axs, fig, y_min_prim_con, y_max_prim_con)
plt.savefig(f'Pictures/Primary_Consolidation_of_separated_layers/
 ↪Primary_Consolidation_of_separated_layers_Zone_E')
plt.close()
```

[16]:
```
arrays_sec_con = [
    Zone_A[3],
    Zone_A_spe[3],
    Zone_B[3],
    Zone_B_spe[3],
    Zone_C[3],
    Zone_D[3],
    Zone_D_spe[3],
    Zone_E[3],
    Zone_E_spe[3],
]

# Concatenate all into one long array
combined_sec_con = np.concatenate(arrays_sec_con)

# Now get min and max
y_min_sec_con = np.min(combined_sec_con)
y_max_sec_con = np.max(combined_sec_con)
```

[17]:
```
# fig, axs = plt.subplots(2, 1, figsize=(7,7), constrained_layout = True)

# Plotting.secondary_consolidation('A', 'regular', Zone_A[4], Zone_A[5], t,␣
 ↪Zone_A[3], 0, axs, fig, Zone_A[6], y_min_sec_con, y_max_sec_con)
# Plotting.secondary_consolidation('A', 'special', Zone_A_spe[4],␣
 ↪Zone_A_spe[5], t, Zone_A_spe[3], 1, axs, fig, Zone_A_spe[6], y_min_sec_con,␣
 ↪y_max_sec_con)

# plt.savefig(f'Pictures/Secondary_Consolidation_of_separated_layers/
 ↪Secondary_Consolidation_of_separated_layers_Zone_A')
# plt.close()
```

```
# fig, axs = plt.subplots(2, 1, figsize=(7,7), constrained_layout = True)

# Plotting.secondary_consolidation('B', 'regular', Zone_B[4], Zone_B[5], t,␣
 ↪Zone_B[3], 0, axs, fig,  Zone_B[6], y_min_sec_con, y_max_sec_con)
# Plotting.secondary_consolidation('B', 'special', Zone_B_spe[4],␣
 ↪Zone_B_spe[5], t, Zone_B_spe[3], 1, axs, fig,  Zone_B_spe[6], y_min_sec_con,␣
 ↪y_max_sec_con)

# plt.savefig(f'Pictures/Secondary_Consolidation_of_separated_layers/
 ↪Secondary_Consolidation_of_separated_layers_Zone_B')
# plt.close()

# fig, axs = plt.subplots(2, 1, figsize=(7,7), constrained_layout = True)

# Plotting.secondary_consolidation('C', 'regular', Zone_C[4], Zone_C[5], t,␣
 ↪Zone_C[3], 0, axs, fig, Zone_C[6], y_min_sec_con, y_max_sec_con)
# axs[1].axis('off')

# plt.savefig(f'Pictures/Secondary_Consolidation_of_separated_layers/
 ↪Secondary_Consolidation_of_separated_layers_Zone_C')
# plt.close()

# fig, axs = plt.subplots(2, 1, figsize=(7,7), constrained_layout = True)

# Plotting.secondary_consolidation('D', 'regular', Zone_D[4], Zone_D[5], t,␣
 ↪Zone_D[3], 0, axs, fig, Zone_D[6], y_min_sec_con, y_max_sec_con)
# Plotting.secondary_consolidation('D', 'special', Zone_D_spe[4],␣
 ↪Zone_D_spe[5], t, Zone_D_spe[3], 1, axs, fig, Zone_D_spe[6], y_min_sec_con,␣
 ↪y_max_sec_con)

# plt.savefig(f'Pictures/Secondary_Consolidation_of_separated_layers/
 ↪Secondary_Consolidation_of_separated_layers_Zone_D')
# plt.close()

# fig, axs = plt.subplots(2, 1, figsize=(7,7), constrained_layout = True)

# Plotting.secondary_consolidation('E', 'regular', Zone_E[4], Zone_E[5], t,␣
 ↪Zone_E[3], 0, axs, fig, Zone_E[6], y_min_sec_con, y_max_sec_con)
# Plotting.secondary_consolidation('E', 'special', Zone_E_spe[4],␣
 ↪Zone_E_spe[5], t, Zone_E_spe[3], 1, axs, fig, Zone_E_spe[6], y_min_sec_con,␣
 ↪y_max_sec_con)

# plt.savefig(f'Pictures/Secondary_Consolidation_of_separated_layers/
 ↪Secondary_Consolidation_of_separated_layers_Zone_E')
# plt.close()
```

```
# plt.savefig('Pictures/Secondary Settelement of seperated layers')
```

[18]:
```
# fig, axs = plt.subplots(2, 1, figsize = (8,8), constrained_layout = True)

# Plotting.set_time('Primary', Zone_lay, 'regular', time_days,␣
 ↪superimposed_settlements, start_times_rel_prot, fig, axs, 0)
# Plotting.set_time('Primary', Zone_lay_spe, 'special', time_days,␣
 ↪superimposed_settlements_spe, start_times_rel_prot, fig, axs, 1)

# plt.savefig('Pictures/
 ↪Primary_Secondary_and_Total_Consolidation_per_defined_zone/
 ↪Primary_Consolidation_per_defined_zone')
# plt.close()

# fig, axs = plt.subplots(2, 1, figsize = (8,8), constrained_layout = True)

# Plotting.set_time('Secondary', Zone_lay, 'regular', t/365, Cre_values,␣
 ↪start_times_rel_prot,fig, axs, 0)
# Plotting.set_time('Secondary', Zone_lay_spe, 'special', t/365,␣
 ↪Cre_values_spe, start_times_rel_prot,fig, axs, 1)

# plt.savefig('Pictures/
 ↪Primary_Secondary_and_Total_Consolidation_per_defined_zone/
 ↪Secondary_Consolidation_per_defined_zone')
# plt.close()

# fig, axs = plt.subplots(2, 1, figsize = (8,8), constrained_layout = True)

# Plotting.set_time('Total', Zone_lay, 'regular', t/365, Tot_set,␣
 ↪start_times_rel_prot,fig, axs, 0)
# Plotting.set_time('Total', Zone_lay_spe, 'special', t/365, Tot_set_spe,␣
 ↪start_times_rel_prot, fig, axs, 1)

# plt.savefig('Pictures/
 ↪Primary_Secondary_and_Total_Consolidation_per_defined_zone/
 ↪Total_Consolidation_per_defined_zone')
```

[19]:
```
arrays_prim_con_lay = [
    Prim_error_A[2],
    Prim_error_A[3],
    Prim_error_A_spe[2],
    Prim_error_A_spe[3],
    Prim_error_B[2],
    Prim_error_B[3],
    Prim_error_B_spe[2],
    Prim_error_B_spe[3],
```

```
    Prim_error_C[2],
    Prim_error_C[3],
    Prim_error_D[2],
    Prim_error_D[3],
    Prim_error_D_spe[2],
    Prim_error_D_spe[3],
    Prim_error_E[2],
    Prim_error_E[3],
    Prim_error_E_spe[2],
    Prim_error_E_spe[3],
]

# Concatenate all into one long array
combined_prim_con_lay = np.concatenate(arrays_prim_con_lay)

# Now get min and max
y_min_prim_con_lay = np.min(combined_prim_con_lay)
y_max_prim_con_lay = np.max(combined_prim_con_lay)
```

[20]:
```
arrays_sec_con_lay = [
    Sec_error_A[2],
    Sec_error_A[3],
    Sec_error_A_spe[2],
    Sec_error_A_spe[3],
    Sec_error_B[2],
    Sec_error_B[3],
    Sec_error_B_spe[2],
    Sec_error_B_spe[3],
    Sec_error_C[2],
    Sec_error_C[3],
    Sec_error_D[2],
    Sec_error_D[3],
    Sec_error_D_spe[2],
    Sec_error_D_spe[3],
    Sec_error_E[2],
    Sec_error_E[3],
    Sec_error_E_spe[2],
    Sec_error_E_spe[3],
]

# Concatenate all into one long array
combined_sec_con_lay = np.concatenate(arrays_sec_con_lay)

# Now get min and max
y_min_sec_con_lay = np.min(combined_sec_con_lay)
y_max_sec_con_lay = np.max(combined_sec_con_lay)
```

[21]:
```
arrays_tot_con_lay = [
    Tot_lay_error_A[2],
    Tot_lay_error_A[3],
    Tot_lay_error_A_spe[2],
    Tot_lay_error_A_spe[3],
    Tot_lay_error_B[2],
    Tot_lay_error_B[3],
    Tot_lay_error_B_spe[2],
    Tot_lay_error_B_spe[3],
    Tot_lay_error_C[2],
    Tot_lay_error_C[3],
    Tot_lay_error_D[2],
    Tot_lay_error_D[3],
    Tot_lay_error_D_spe[2],
    Tot_lay_error_D_spe[3],
    Tot_lay_error_E[2],
    Tot_lay_error_E[3],
    Tot_lay_error_E_spe[2],
    Tot_lay_error_E_spe[3],
]

# Concatenate all into one long array
combined_tot_con_lay = np.concatenate(arrays_tot_con_lay)

# Now get min and max
y_min_tot_con_lay = np.min(combined_tot_con_lay)
y_max_tot_con_lay = np.max(combined_tot_con_lay)
```

[22]:
```
# fig, axs = plt.subplots(2,1,figsize=(12,8))

# Plotting.errorbar_lay('A', 'Primary', 'regular', Zone_A, time_days,␣
 ↪Prim_error_A[0], Prim_error_A[1], Prim_error_A[2], Prim_error_A[3], t_l_A,␣
 ↪fig, axs, 0)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,8))
# Plotting.errorbar_lay('A', 'Secondary', 'regular', Zone_A, t/365,␣
 ↪Sec_error_A[0], Sec_error_A[1], Sec_error_A[2], Sec_error_A[3], t_l_A, fig,␣
 ↪axs, 0)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,8))
# Plotting.errorbar_lay('A', 'Total', 'regular', Zone_A, t/365,␣
 ↪Tot_lay_error_A[0], Tot_lay_error_A[1], Tot_lay_error_A[2],␣
 ↪Tot_lay_error_A[3], t_l_A, fig, axs, 0)
```

```
# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,8))
# Plotting.errorbar_lay('A', 'Primary', 'special', Zone_A_spe, time_days,␣
 ↪Prim_error_A_spe[0], Prim_error_A_spe[1], Prim_error_A_spe[2],␣
 ↪Prim_error_A_spe[3], t_l_A_spe, fig, axs, 0)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,8))
# Plotting.errorbar_lay('A', 'Secondary', 'special', Zone_A_spe, t/365,␣
 ↪Sec_error_A_spe[0], Sec_error_A_spe[1], Sec_error_A_spe[2],␣
 ↪Sec_error_A_spe[3], t_l_A_spe, fig, axs, 0)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,8))
# Plotting.errorbar_lay('A', 'Total', 'special', Zone_A_spe, t/365,␣
 ↪Tot_lay_error_A_spe[0], Tot_lay_error_A_spe[1], Tot_lay_error_A_spe[2],␣
 ↪Tot_lay_error_A_spe[3], t_l_A_spe, fig, axs, 0)

# plt.close()
```

[23]:
```
# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('B', 'Primary', 'regular', Zone_B, time_days,␣
 ↪Prim_error_B[0], Prim_error_B[1], Prim_error_B[2], Prim_error_B[3],␣
 ↪Prim_set_B, t_l_B, fig, axs, 0, y_min_prim_con_lay, y_max_prim_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('B', 'Secondary', 'regular', Zone_B, t/365,␣
 ↪Sec_error_B[0], Sec_error_B[1], Sec_error_B[2], Sec_error_B[3], Sec_set_B,␣
 ↪t_l_B, fig, axs, 0, y_min_sec_con_lay, y_max_sec_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('B', 'Total', 'regular', Zone_B, t/365,␣
 ↪Tot_lay_error_B[0], Tot_lay_error_B[1], Tot_lay_error_B[2],␣
 ↪Tot_lay_error_B[3], Tot_lay_set_B, t_l_B, fig, axs, 0, y_min_tot_con_lay,␣
 ↪y_max_tot_con_lay)
```

```
# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('B', 'Primary', 'special', Zone_B_spe, time_days,␣
 ↪Prim_error_B_spe[0], Prim_error_B_spe[1], Prim_error_B_spe[2],␣
 ↪Prim_error_B_spe[3], Prim_set_B_spe, t_l_B_spe, fig, axs, 0,␣
 ↪y_min_prim_con_lay, y_max_prim_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('B', 'Secondary', 'special', Zone_B_spe, t/365,␣
 ↪Sec_error_B_spe[0], Sec_error_B_spe[1], Sec_error_B_spe[2],␣
 ↪Sec_error_B_spe[3], Sec_set_B_spe, t_l_B_spe, fig, axs, 0,␣
 ↪y_min_sec_con_lay, y_max_sec_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('B', 'Total', 'special', Zone_B_spe, t/365,␣
 ↪Tot_lay_error_B_spe[0], Tot_lay_error_B_spe[1], Tot_lay_error_B_spe[2],␣
 ↪Tot_lay_error_B_spe[3], Tot_lay_set_B_spe, t_l_B_spe, fig, axs, 0,␣
 ↪y_min_tot_con_lay, y_max_tot_con_lay)

# plt.close()
```

[24]:
```
# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('C', 'Primary', 'regular', Zone_C, time_days,␣
 ↪Prim_error_C[0], Prim_error_C[1], Prim_error_C[2], Prim_error_C[3],␣
 ↪Prim_set_C, t_l_C, fig, axs, 0, y_min_prim_con_lay, y_max_prim_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('C', 'Secondary', 'regular', Zone_C, t/365,␣
 ↪Sec_error_C[0], Sec_error_C[1], Sec_error_C[2], Sec_error_C[3], Sec_set_C,␣
 ↪t_l_C, fig, axs, 0, y_min_sec_con_lay, y_max_sec_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))
```

```
# Plotting.errorbar_lay('C', 'Total', 'regular', Zone_C, t/365,␣
 ↪Tot_lay_error_C[0], Tot_lay_error_C[1], Tot_lay_error_C[2],␣
 ↪Tot_lay_error_C[3], Tot_lay_set_C, t_l_C, fig, axs, 0, y_min_tot_con_lay,␣
 ↪y_max_tot_con_lay)

# plt.close()
```

[25]:
```
# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('D', 'Primary', 'regular', Zone_D, time_days,␣
 ↪Prim_error_D[0], Prim_error_D[1], Prim_error_D[2], Prim_error_D[3],␣
 ↪Prim_set_D, t_l_D, fig, axs, 0, y_min_prim_con_lay, y_max_prim_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('D', 'Secondary', 'regular', Zone_D, t/365,␣
 ↪Sec_error_D[0], Sec_error_D[1], Sec_error_D[2], Sec_error_D[3], Sec_set_D,␣
 ↪t_l_D, fig, axs, 0, y_min_sec_con_lay, y_max_sec_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('D', 'Total', 'regular', Zone_D, t/365,␣
 ↪Tot_lay_error_D[0], Tot_lay_error_D[1], Tot_lay_error_D[2],␣
 ↪Tot_lay_error_D[3], Tot_lay_set_D, t_l_D, fig, axs, 0, y_min_tot_con_lay,␣
 ↪y_max_tot_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('D', 'Primary', 'special', Zone_D_spe, time_days,␣
 ↪Prim_error_D_spe[0], Prim_error_D_spe[1], Prim_error_D_spe[2],␣
 ↪Prim_error_D_spe[3], Prim_set_D_spe, t_l_D_spe, fig, axs, 0,␣
 ↪y_min_prim_con_lay, y_max_prim_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('D', 'Secondary', 'special', Zone_D_spe, t/365,␣
 ↪Sec_error_D_spe[0], Sec_error_D_spe[1], Sec_error_D_spe[2],␣
 ↪Sec_error_D_spe[3], Sec_set_D_spe, t_l_D_spe, fig, axs, 0,␣
 ↪y_min_sec_con_lay, y_max_sec_con_lay)
```

```
# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('D', 'Total', 'special', Zone_D_spe, t/365,␣
 ↪Tot_lay_error_D_spe[0], Tot_lay_error_D_spe[1], Tot_lay_error_D_spe[2],␣
 ↪Tot_lay_error_D_spe[3], Tot_lay_set_D_spe, t_l_D_spe, fig, axs, 0,␣
 ↪y_min_tot_con_lay, y_max_tot_con_lay)

# plt.close()
```

[26]:
```
# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('E', 'Primary', 'regular', Zone_E, time_days,␣
 ↪Prim_error_E[0], Prim_error_E[1], Prim_error_E[2], Prim_error_E[3],␣
 ↪Prim_set_E, t_l_E, fig, axs, 0, y_min_prim_con_lay, y_max_prim_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('E', 'Secondary', 'regular', Zone_E, t/365,␣
 ↪Sec_error_E[0], Sec_error_E[1], Sec_error_E[2], Sec_error_E[3], Sec_set_E,␣
 ↪t_l_E, fig, axs, 0, y_min_sec_con_lay, y_max_sec_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('E', 'Total', 'regular', Zone_E, t/365,␣
 ↪Tot_lay_error_E[0], Tot_lay_error_E[1], Tot_lay_error_E[2],␣
 ↪Tot_lay_error_E[3], Tot_lay_set_E, t_l_E, fig, axs, 0, y_min_tot_con_lay,␣
 ↪y_max_tot_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('E', 'Primary', 'special', Zone_E_spe, time_days,␣
 ↪Prim_error_E_spe[0], Prim_error_E_spe[1], Prim_error_E_spe[2],␣
 ↪Prim_error_E_spe[3], Prim_set_E_spe, t_l_E_spe, fig, axs, 0,␣
 ↪y_min_prim_con_lay, y_max_prim_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))
```

```
# Plotting.errorbar_lay('E', 'Secondary', 'special', Zone_E_spe, t/365,␣
 ↪Sec_error_E_spe[0], Sec_error_E_spe[1], Sec_error_E_spe[2],␣
 ↪Sec_error_E_spe[3], Sec_set_E_spe, t_l_E_spe, fig, axs, 0,␣
 ↪y_min_sec_con_lay, y_max_sec_con_lay)

# plt.close()

# fig, axs = plt.subplots(2,1,figsize=(12,9))

# Plotting.errorbar_lay('E', 'Total', 'special', Zone_E_spe, t/365,␣
 ↪Tot_lay_error_E_spe[0], Tot_lay_error_E_spe[1], Tot_lay_error_E_spe[2],␣
 ↪Tot_lay_error_E_spe[3], Tot_lay_set_E_spe, t_l_E_spe, fig, axs, 0,␣
 ↪y_min_tot_con_lay, y_max_tot_con_lay)
# plt.close()
```

[27]:
```
# print(f'primary regular {np.array(Prim_error_E)[2][3][-1]*1000:.1f}')
# print(f'primary regular {np.array(Prim_error_E)[3][3][-1]*1000:.1f}')
# print(f'primary special {np.array(Prim_error_E_spe)[2][3][-1]*1000:.1f}')
# print(f'primary special {np.array(Prim_error_E_spe)[3][3][-1]*1000:.1f}')
# print(f'secondary regular {np.array(Sec_error_E)[2][3][-1]*1000:.1f}')
# print(f'secondary regular {np.array(Sec_error_E)[3][3][-1]*1000:.1f}')
# print(f'secondary special {np.array(Sec_error_E_spe)[2][3][-1]*1000:.1f}')
# print(f'secondary special {np.array(Sec_error_E_spe)[3][3][-1]*1000:.1f}')
```

[28]:
```
# print(f'primary regular {prim_fi_perc_superimposed_settlements[4][-1]*1000:.
 ↪1f}')
# print(f'primary regular {prim_la_perc_superimposed_settlements[4][-1]*1000:.
 ↪1f}')
# print(f'primary special␣
 ↪{prim_fi_perc_superimposed_settlements_spe[3][-1]*1000:.1f}')
# print(f'primary special␣
 ↪{prim_la_perc_superimposed_settlements_spe[3][-1]*1000:.1f}')
print(f'secondary regular B {sec_fi_perc_superimposed_settlements[1][-1]*1000:.
 ↪1f}')
print(f'secondary regular B {sec_la_perc_superimposed_settlements[1][-1]*1000:.
 ↪1f}')
print(f'secondary special B␣
 ↪{sec_fi_perc_superimposed_settlements_spe[1][-1]*1000:.1f}')
print(f'secondary special B␣
 ↪{sec_la_perc_superimposed_settlements_spe[1][-1]*1000:.1f}')
print(f'secondary regular C {sec_fi_perc_superimposed_settlements[2][-1]*1000:.
 ↪1f}')
print(f'secondary regular C {sec_la_perc_superimposed_settlements[2][-1]*1000:.
 ↪1f}')
```

```
print(f'secondary regular D {sec_fi_perc_superimposed_settlements[3][-1]*1000:.
↪1f}')
print(f'secondary regular D {sec_la_perc_superimposed_settlements[3][-1]*1000:.
↪1f}')
print(f'secondary special D␣
↪{sec_fi_perc_superimposed_settlements_spe[2][-1]*1000:.1f}')
print(f'secondary special D␣
↪{sec_la_perc_superimposed_settlements_spe[2][-1]*1000:.1f}')
print(f'secondary regular E {sec_fi_perc_superimposed_settlements[4][-1]*1000:.
↪1f}')
print(f'secondary regular E {sec_la_perc_superimposed_settlements[4][-1]*1000:.
↪1f}')
print(f'secondary special E␣
↪{sec_fi_perc_superimposed_settlements_spe[3][-1]*1000:.1f}')
print(f'secondary special E␣
↪{sec_la_perc_superimposed_settlements_spe[3][-1]*1000:.1f}')
# print(f'initial regular {- Ini_error_E[2][0]*1000:.1f}')
# print(f'initial regular {- Ini_error_E[3][0]*1000:.1f}')
# print(f'initial special {- Ini_error_E_spe[2][0]*1000:.1f}')
# print(f'initial special {- Ini_error_E_spe[3][0]*1000:.1f}')
```

```
secondary regular B 26.5
secondary regular B 129.1
secondary special B 18.5
secondary special B 118.9
secondary regular C 5.2
secondary regular C 34.3
secondary regular D -37.9
secondary regular D 47.1
secondary special D -55.7
secondary special D 34.4
secondary regular E -44.8
secondary regular E 44.6
secondary special E -46.1
secondary special E 44.2
```

[29]:
```
fig, axs = plt.subplots(2, 2, figsize = (16,9), constrained_layout = True)

# Plotting.errorbar_zone('Primary', Zone_lay, 'regular', time_days,␣
↪prim_mean_superimposed_settlements, prim_std_superimposed_settlements,␣
↪start_times_rel_prot, fig, axs, 0, 0)
# Plotting.errorbar_zone('Primary', Zone_lay_spe, 'special', time_days,␣
↪prim_mean_superimposed_settlements_spe,␣
↪prim_std_superimposed_settlements_spe, start_times_rel_prot, fig, axs, 1, 0)
```

```
Plotting.perc_zone('Primary', Zone_lay, 'regular', time_days,␣
 ↪prim_mean_superimposed_settlements, prim_std_superimposed_settlements,␣
 ↪prim_fi_perc_superimposed_settlements,␣
 ↪prim_la_perc_superimposed_settlements, Tot_prim_set, start_times_rel_prot,␣
 ↪fig, axs, 0)
Plotting.perc_zone('Primary', Zone_lay_spe, 'special', time_days,␣
 ↪prim_mean_superimposed_settlements_spe,␣
 ↪prim_std_superimposed_settlements_spe,␣
 ↪prim_fi_perc_superimposed_settlements_spe,␣
 ↪prim_la_perc_superimposed_settlements_spe, Tot_prim_set_spe,␣
 ↪start_times_rel_prot, fig, axs, 1)
plt.savefig('Pictures/
 ↪Variability_of_Primary_Secondary_and_Total_consolidation_per_defined_zone/
 ↪Variability_of_primary_consolidation_per_defined_zone')
plt.close()

fig, axs = plt.subplots(2, 2, figsize = (16,9), constrained_layout = True)

# Plotting.errorbar_zone('Secondary', Zone_lay, 'regular', t/365, ␣
 ↪sec_mean_superimposed_settlements, sec_std_superimposed_settlements,␣
 ↪start_times_rel_prot, fig, axs, 0, 0)
# Plotting.errorbar_zone('Secondary', Zone_lay_spe, 'special', t/365,␣
 ↪sec_mean_superimposed_settlements_spe, sec_std_superimposed_settlements_spe,␣
 ↪start_times_rel_prot, fig, axs, 1, 0)

Plotting.perc_zone('Secondary', Zone_lay, 'regular', t/365, ␣
 ↪sec_mean_superimposed_settlements, sec_std_superimposed_settlements,␣
 ↪sec_fi_perc_superimposed_settlements, sec_la_perc_superimposed_settlements,␣
 ↪Tot_sec_set, start_times_rel_prot, fig, axs, 0)
Plotting.perc_zone('Secondary', Zone_lay_spe, 'special', t/365,␣
 ↪sec_mean_superimposed_settlements_spe, sec_std_superimposed_settlements_spe,␣
 ↪sec_fi_perc_superimposed_settlements_spe,␣
 ↪sec_la_perc_superimposed_settlements_spe, Tot_sec_set_spe,␣
 ↪start_times_rel_prot, fig, axs, 1)

plt.savefig('Pictures/
 ↪Variability_of_Primary_Secondary_and_Total_consolidation_per_defined_zone/
 ↪Variability_of_secondary_consolidation_per_defined_zone')
plt.close()

fig, axs = plt.subplots(2, 2, figsize = (16,9), constrained_layout = True)

# Plotting.errorbar_zone('Total', Zone_lay, 'regular', t/365,␣
 ↪Tot_mean_superimposed_settlements, Tot_std_superimposed_settlements,␣
 ↪start_times_rel_prot, fig, axs, 0,0)
```

```
# Plotting.errorbar_zone('Total', Zone_lay_spe, 'special', t/365,␣
 ↪Tot_mean_superimposed_settlements_spe, Tot_std_superimposed_settlements_spe,␣
 ↪start_times_rel_prot, fig, axs, 1,0)

Plotting.perc_zone('Total', Zone_lay, 'regular', t/365,␣
 ↪Tot_mean_superimposed_settlements, Tot_std_superimposed_settlements,␣
 ↪Tot_fi_perc_superimposed_settlements, Tot_la_perc_superimposed_settlements,␣
 ↪Tot_set_zone, start_times_rel_prot, fig, axs, 0)
Plotting.perc_zone('Total', Zone_lay_spe, 'special', t/365,␣
 ↪Tot_mean_superimposed_settlements_spe, Tot_std_superimposed_settlements_spe,␣
 ↪ Tot_fi_perc_superimposed_settlements_spe,␣
 ↪Tot_la_perc_superimposed_settlements_spe, Tot_set_zone_spe,␣
 ↪start_times_rel_prot, fig, axs, 1)

plt.savefig('Pictures/
 ↪Variability_of_Primary_Secondary_and_Total_consolidation_per_defined_zone/
 ↪Variability_of_total_consolidation_per_defined_zone')
plt.close()
print('done')
```

```
done
```

[30]:
```
# fig, axs = plt.subplots(6, 1, figsize = (13,5*10), constrained_layout = True)

# Plotting.perc_zone('Primary', Zone_lay, 'regular', time_days,␣
 ↪prim_mean_superimposed_settlements, prim_fi_perc_superimposed_settlements,␣
 ↪prim_la_perc_superimposed_settlements, start_times_rel_prot, fig, axs, 0)
# Plotting.perc_zone('Primary', Zone_lay_spe, 'special', time_days,␣
 ↪prim_mean_superimposed_settlements_spe,␣
 ↪prim_fi_perc_superimposed_settlements_spe,␣
 ↪prim_la_perc_superimposed_settlements_spe, start_times_rel_prot, fig, axs, 1)
# Plotting.perc_zone('Secondary', Zone_lay, 'regular', t/365, ␣
 ↪sec_mean_superimposed_settlements, sec_fi_perc_superimposed_settlements,␣
 ↪sec_la_perc_superimposed_settlements, start_times_rel_prot, fig, axs, 2)
# Plotting.perc_zone('Secondary', Zone_lay_spe, 'special', t/365,␣
 ↪sec_mean_superimposed_settlements_spe,␣
 ↪sec_fi_perc_superimposed_settlements_spe,␣
 ↪sec_la_perc_superimposed_settlements_spe, start_times_rel_prot, fig, axs, 3)
# Plotting.perc_zone('Total', Zone_lay, 'regular', t/365,␣
 ↪Tot_mean_superimposed_settlements, Tot_fi_perc_superimposed_settlements,␣
 ↪Tot_la_perc_superimposed_settlements, start_times_rel_prot, fig, axs, 4)
# Plotting.perc_zone('Total', Zone_lay_spe, 'special', t/365,␣
 ↪Tot_mean_superimposed_settlements_spe,␣
 ↪Tot_fi_perc_superimposed_settlements_spe,␣
 ↪Tot_la_perc_superimposed_settlements_spe, start_times_rel_prot, fig, axs, 5)

# plt.savefig('Pictures/5th and 95th Percentile Settelement per Defined Zone')
```

[31]:
```
arrays_q = [
    Tot_set_ini_A,
    Tot_set_ini_A_spe,
    Tot_set_ini_B,
    Tot_set_ini_B_spe,
    Tot_set_ini_C,
    Tot_set_ini_D,
    Tot_set_ini_D_spe,
    Tot_set_ini_E,
    Tot_set_ini_E_spe
]

# Concatenate all into one long array
combined_q = np.concatenate(arrays_q)

# Now get min and max
y_min_q = np.min(combined_q)
y_max_q = np.max(combined_q)
```

[32]:
```
# fig, axs = plt.subplots(5, 2, figsize=(10,12))

# Plotting.plotting_q('A', 'regular', Zone_A[7], y_q_A, dx, L_ele_reg, u_r_q_A,␣
 ↪F_ex_q_A, 0, fig, axs, y_min_q, y_max_q)
# Plotting.plotting_q('A', 'special', Zone_A_spe[7], y_q_A_spe, dx, L_ele_spe,␣
 ↪u_r_q_A_spe, F_ex_q_A_spe, 0, fig, axs, y_min_q, y_max_q)
# Plotting.plotting_q('B', 'regular', Zone_B[7], y_q_B, dx, L_ele_reg, u_r_q_B,␣
 ↪F_ex_q_B, 1 , fig, axs, y_min_q, y_max_q)
# Plotting.plotting_q('B', 'special', Zone_B_spe[7], y_q_B_spe, dx, L_ele_spe,␣
 ↪u_r_q_B_spe, F_ex_q_B_spe, 1 , fig, axs, y_min_q, y_max_q)
# Plotting.plotting_q('C', 'regular', Zone_C[7], y_q_C, dx, L_ele_reg, u_r_q_C,␣
 ↪F_ex_q_C, 2 , fig, axs, y_min_q, y_max_q)

# axs[2,1].axis('off')

# Plotting.plotting_q('D', 'regular', Zone_D[7], y_q_D, dx, L_ele_reg, u_r_q_D,␣
 ↪F_ex_q_D, 3 , fig, axs, y_min_q, y_max_q)
# Plotting.plotting_q('D', 'special', Zone_D_spe[7], y_q_D_spe, dx, L_ele_spe,␣
 ↪u_r_q_D_spe, F_ex_q_D_spe, 3 , fig, axs, y_min_q, y_max_q)
# Plotting.plotting_q('E', 'regular', Zone_E[7], y_q_E, dx, L_ele_reg, u_r_q_E,␣
 ↪F_ex_q_E, 4 , fig, axs, y_min_q, y_max_q)
# Plotting.plotting_q('E', 'special', Zone_E_spe[7], y_q_E_spe, dx, L_ele_spe,␣
 ↪u_r_q_E_spe, F_ex_q_E_spe, 4 , fig, axs, y_min_q, y_max_q)


# plt.savefig('Pictures/Soil structure interaction over length complete beam')
# plt.close()
```

[33]:
```
# arrays_ini = [
#     -Ini_error_B[2],
#     -Ini_error_B[3],
#     -Ini_error_B_spe[2],
#     -Ini_error_B_spe[3],
#     -Ini_error_C[2],
#     -Ini_error_C[3],
#     -Ini_error_D[2],
#     -Ini_error_D[3],
#     -Ini_error_D_spe[2],
#     -Ini_error_D_spe[3],
#     -Ini_error_E[2],
#     -Ini_error_E[3],
#     -Ini_error_E_spe[2],
#     -Ini_error_E_spe[3],
# ]

# # Concatenate all into one long array
# combined_ini = np.concatenate(arrays_ini)

# # Now get min and max
# y_min_ini = np.min(combined_ini)
# y_max_ini = np.max(combined_ini)

# Plotting.errorbar_ini('B', 'regular', x, Ini_error_B, Ini_set_B, y_min_ini,␣
 ↪y_max_ini)
# Plotting.errorbar_ini('B', 'special', x_spe, Ini_error_B_spe, Ini_set_B_spe,␣
 ↪y_min_ini, y_max_ini)
# Plotting.errorbar_ini('C', 'regular', x, Ini_error_C, Ini_set_C, y_min_ini,␣
 ↪y_max_ini)
# Plotting.errorbar_ini('D', 'regular', x, Ini_error_D, Ini_set_D, y_min_ini,␣
 ↪y_max_ini)
# Plotting.errorbar_ini('D', 'special', x_spe, Ini_error_D_spe, Ini_set_D_spe,␣
 ↪y_min_ini, y_max_ini)
# Plotting.errorbar_ini('E', 'regular', x, Ini_error_E, Ini_set_E, y_min_ini,␣
 ↪y_max_ini)
# Plotting.errorbar_ini('E', 'special', x_spe, Ini_error_E_spe, Ini_set_E_spe,␣
 ↪y_min_ini, y_max_ini)
```

[34]:
```
# fig, axs = plt.subplots(nrows=2, ncols=2,  figsize=(8,8))
# Plotting.Set_zone('A', 'regular', times,  Set_error_A, x, 0, fig, axs)
# Plotting.Set_zone('A', 'special', times,  Set_error_A_spe, x_spe, 1, fig, axs)
# plt.savefig('Pictures/Variability_initial_deformation_per_time_and_zone/
 ↪Variability_initial_deformation_per_time_and_zone_A')
# plt.close()

# fig, axs = plt.subplots(nrows=2, ncols=2,  figsize=(8,8))
```

```
# Plotting.Set_zone('B', 'regular', times,  Set_error_B, x, 0, fig, axs)
# Plotting.Set_zone('B', 'special', times,  Set_error_B_spe, x_spe, 1, fig, axs)
# plt.savefig('Pictures/Variability_initial_deformation_per_time_and_zone/
 ↪Variability_initial_deformation_per_time_and_zone_B')
# plt.close()

# fig, axs = plt.subplots(nrows=2, ncols=2,  figsize=(8,8))
# Plotting.Set_zone('C', 'regular', times,  Set_error_C, x, 0, fig, axs)
# axs[1,0].axis('off')
# axs[1,1].axis('off')
# plt.savefig('Pictures/Variability_initial_deformation_per_time_and_zone/
 ↪Variability_initial_deformation_per_time_and_zone_C')
# plt.close()

# fig, axs = plt.subplots(nrows=2, ncols=2,  figsize=(8,8))
# Plotting.Set_zone('D', 'regular', times,  Set_error_D, x, 0, fig, axs)
# Plotting.Set_zone('D', 'special', times,  Set_error_D_spe, x_spe, 1, fig, axs)
# plt.savefig('Pictures/Variability_initial_deformation_per_time_and_zone/
 ↪Variability_initial_deformation_per_time_and_zone_D')
# plt.close()

# fig, axs = plt.subplots(nrows=2, ncols=2,  figsize=(8,8))
# Plotting.Set_zone('E', 'regular', times,  Set_error_E, x, 0, fig, axs)
# Plotting.Set_zone('E', 'special', times,  Set_error_E_spe, x_spe, 1, fig, axs)
# plt.savefig('Pictures/Variability_initial_deformation_per_time_and_zone/
 ↪Variability_initial_deformation_per_time_and_zone_E')
# plt.close()
```

[35]:
```
# fig, axs = plt.subplots(8, 2*len(times), figsize = (13*len(times),6*10))

# # Plotting.errorbar_set('A', 'regular', times, Set_error_A[0],␣
 ↪Set_error_A[1], Set_error_A[2], Set_error_A[3], x, fig, axs, 0)
# # Plotting.errorbar_set('A', 'special', times, Set_error_A_spe[0],␣
 ↪Set_error_A_spe[1], Set_error_A_spe[2], Set_error_A_spe[3], x_spe, fig, axs,␣
 ↪0)
# Plotting.errorbar_set('B', 'regular', times, Set_error_B[0], Set_error_B[1],␣
 ↪Set_error_B[2], Set_error_B[3], x, fig, axs, 0)
# Plotting.errorbar_set('B', 'special', times, Set_error_B_spe[0],␣
 ↪Set_error_B_spe[1], Set_error_B_spe[2], Set_error_B_spe[3], x_spe, fig, axs,␣
 ↪0)
# Plotting.errorbar_set('C', 'regular', times, Set_error_C[0], Set_error_C[1],␣
 ↪Set_error_C[2], Set_error_C[3], x, fig, axs, 2)
# for col in range(6, 12):
#     axs[2, col].axis('off')
#     axs[3, col].axis('off')
```

```
# Plotting.errorbar_set('D', 'regular', times, Set_error_D[0], Set_error_D[1],␣
 ↪Set_error_D[2], Set_error_D[3], x, fig, axs, 4)
# Plotting.errorbar_set('D', 'special', times, Set_error_D_spe[0],␣
 ↪Set_error_D_spe[1], Set_error_D_spe[2], Set_error_D_spe[3], x_spe, fig, axs,␣
 ↪4)
# Plotting.errorbar_set('E', 'regular', times, Set_error_E[0], Set_error_E[1],␣
 ↪Set_error_E[2], Set_error_E[3], x, fig, axs, 6)
# Plotting.errorbar_set('E', 'special', times, Set_error_E_spe[0],␣
 ↪Set_error_E_spe[1], Set_error_E_spe[2], Set_error_E_spe[3], x_spe, fig, axs,␣
 ↪6)

# # # Create legends for each column
# # for col in range(2*len(times)):
# #     handles, labels = axs[0, col].get_legend_handles_labels()
# #     fig.legend(handles, labels, loc='lower center', ncol=3,␣
 ↪bbox_to_anchor=(0.5, -0.05 - col * 0.05))

# plt.subplots_adjust(hspace=0.4, wspace=0.4, bottom=0.2)

# plt.savefig('Pictures/Error_total_settlement_seperate_tunnelparts')
# # plt.close()
```

[36]:
```
x = np.arange(0,L_ele_reg+dx,dx)
x_spe = np.arange(0,L_ele_spe+dx,dx)
#[5, 10, 20, 25, 50, 100]

x_tot_A = np.concatenate([x, x + x[-1] + dx, x_spe + 2 * (x[-1] + dx), x + 2 *␣
 ↪(x[-1] + dx) + x_spe[-1] + dx, x + 3 * (x[-1] + dx) + x_spe[-1] + dx])
x_tot_B = x_tot_A
x_tot_C = np.concatenate([x, x + x[-1] + dx, x + 2 * (x[-1] + dx), x + 2 *␣
 ↪(x[-1] + dx) + x[-1] + dx, x + 3 * (x[-1] + dx) + x[-1] + dx])
x_tot_D = x_tot_A
x_tot_E = x_tot_A

Set_A_comp = np.concatenate([Set_A, Set_A, Set_A_spe, Set_A, Set_A], axis=1)
Set_B_comp = np.concatenate([Set_B, Set_B, Set_B_spe, Set_B, Set_B], axis=1)
Set_C_comp = np.concatenate([Set_C, Set_C, Set_C, Set_C, Set_C], axis=1)
Set_D_comp = np.concatenate([Set_D, Set_D, Set_D_spe, Set_D, Set_D], axis=1)
Set_E_comp = np.concatenate([Set_E, Set_E, Set_E_spe, Set_E, Set_E], axis=1)

fig, axs = plt.subplots(nrows=2, ncols=1, sharex=True, figsize=(10, 10))
Plotting.plot_settlement(axs[0], x_tot_A, Set_A_comp, times, 'A')
Plotting.plot_settlement(axs[1], x_tot_B, Set_B_comp, times, 'B')
plt.savefig('Pictures/Total_deformation_over_length_of_tunnel/
 ↪Total_deformation_over_length_of_tunnel_zone_A_B')
plt.close()
```

```
fig, axs = plt.subplots(nrows=2, ncols=1, sharex=True, figsize=(10, 10))
Plotting.plot_settlement(axs[0], x_tot_C, Set_C_comp, times, 'C')
axs[1].axis('off')
plt.savefig('Pictures/Total_deformation_over_length_of_tunnel/
 ↪Total_deformation_over_length_of_tunnel_zone_C')
plt.close()

fig, axs = plt.subplots(nrows=2, ncols=1, sharex=True, figsize=(10, 10))
Plotting.plot_settlement(axs[0], x_tot_D, Set_D_comp, times, 'D')
Plotting.plot_settlement(axs[1], x_tot_E, Set_E_comp, times, 'E')
plt.savefig('Pictures/Total_deformation_over_length_of_tunnel/
 ↪Total_deformation_over_length_of_tunnel_zone_D_E')
plt.close()
```

[37]:
```
arrays_tot = [
    Set_error_B[2],
    Set_error_B[3],
    Set_error_C[2],
    Set_error_C[3],
    Set_error_D[2],
    Set_error_D[3],
    Set_error_E[2],
    Set_error_E[3],
]


arrays_tot_spe = [
    Set_error_B_spe[2],
    Set_error_B_spe[3],
    Set_error_D_spe[2],
    Set_error_D_spe[3],
    Set_error_E_spe[2],
    Set_error_E_spe[3]
]

# Concatenate all into one long array
combined_tot = np.concatenate(arrays_tot)
combined_tot_spe = np.concatenate(arrays_tot_spe)

# Now get min and max
y_min_tot = np.min(combined_tot)
y_max_tot = np.max(combined_tot)
y_min_tot_spe = np.min(combined_tot_spe)
y_max_tot_spe = np.max(combined_tot_spe)

y_min_tot = np.min([y_min_tot, y_min_tot_spe])
```

```
y_max_tot = np.max([y_max_tot, y_max_tot_spe])
```

[38]:
```
# Plotting.Tot_error('A', x_tot_A, times, Set_error_A, Set_error_A_spe, 0, fig,␣
 ↪axs)

fig, axs = plt.subplots(nrows=4, ncols=2, figsize=(10, 15))
Plotting.Tot_error('B', x_tot_B, times, Final_B, Final_B_spe, Set_error_B,␣
 ↪Set_error_B_spe, 0, fig, axs, y_min_tot, y_max_tot)
plt.subplots_adjust(hspace=0.5, wspace=0.4, bottom=0.2)
plt.savefig('Pictures/Total_settlement_over_length_of_tunnel_including_errors/
 ↪Total_settlement_over_length_of_tunnel_including_errors_for_zone_B')
plt.close()

fig, axs = plt.subplots(nrows=4, ncols=2, figsize=(10, 15))
Plotting.Tot_error('C', x_tot_C, times, Final_C, 0, Set_error_C, 0, 0, fig,␣
 ↪axs, y_min_tot, y_max_tot)
plt.subplots_adjust(hspace=0.5, wspace=0.4, bottom=0.2)
axs[3,1].axis('off')
plt.savefig('Pictures/Total_settlement_over_length_of_tunnel_including_errors/
 ↪Total_settlement_over_length_of_tunnel_including_errors_for_zone_C')
plt.close()

fig, axs = plt.subplots(nrows=4, ncols=2, figsize=(10, 15))
Plotting.Tot_error('D', x_tot_D, times, Final_D, Final_D_spe, Set_error_D,␣
 ↪Set_error_D_spe, 0, fig, axs, y_min_tot, y_max_tot)
plt.subplots_adjust(hspace=0.5, wspace=0.4, bottom=0.2)
plt.savefig('Pictures/Total_settlement_over_length_of_tunnel_including_errors/
 ↪Total_settlement_over_length_of_tunnel_including_errors_for_zone_D')
plt.close()

fig, axs = plt.subplots(nrows=4, ncols=2, figsize=(10, 15))
Plotting.Tot_error('E', x_tot_E, times, Final_E, Final_E_spe, Set_error_E,␣
 ↪Set_error_E_spe, 0, fig, axs, y_min_tot, y_max_tot)
plt.subplots_adjust(hspace=0.5, wspace=0.4, bottom=0.2)
plt.savefig('Pictures/Total_settlement_over_length_of_tunnel_including_errors/
 ↪Total_settlement_over_length_of_tunnel_including_errors_for_zone_E')
plt.close()

print('done')
```

```
done
```

[39]:
```
print(f'regular {np.array(Set_error_E)[2][5][-1]*1000 :.1f}')
print(f'regular {np.array(Set_error_E)[3][5][-1]*1000 :.1f}')
print(f'special {np.array(Set_error_E_spe)[2][5][-1]*1000 :.1f}')
print(f'special {np.array(Set_error_E_spe)[3][5][-1]*1000 :.1f}')
```

```
regular -87.0
```

```
regular 60.3
special -91.1
special 60.4
```