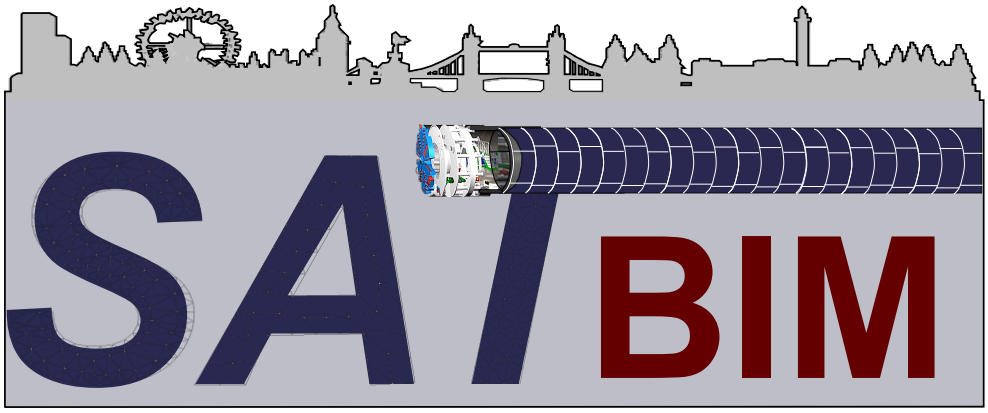
TECHNICAL REPORT

January 26, 2019



"SATBIM"
Simulations for multi-level Analysis of interactions in Tunnelling based on the Building Information Modelling technology

Jelena Ninić

Acknowledgement

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No 702874. This support is gratefully acknowledged.

Abstract

With increasing urbanization and mobility, the need for underground facilities and consequently efficient and safe design and construction techniques grows. The goal of the SATBIM project is to develop a multi-level simulation model for tunnel structure interaction integrated in the framework of Building Information Modelling to support engineering decisions during the project life cycle and to allow for the evaluation and minimization of risks on the existing infrastructure. SATBIM is an integrated platform for structural analysis, visualisation and optimization of the mechanized tunnelling process from early stages of the design over to the construction and the operation phase. The complete concept will be validated using industrial data with reference to the real tunnel project in Germany. The output will have wide implication on technology with expected high academic and industrial impact.

Author :	Dr Jelena Ninić Centre for Structural Engineering and Informatics, The University of Nottingham, UK
Project Coordinator:	Dr Christian Koch Centre for Structural Engineering and Informatics, The University of Nottingham, UK
Collaboration :	MSc. Hoang-Giang Bui[a] Prof. Günther Meschke[a] Prof. Markus König[b] [a]Institute for Structural Mechanics, [b]Chair of Computing in Engineering, Ruhr University Bochum, Germany

CONTENTS

1 Concept

Building and construction information modelling for decision making during the life cycle of infrastructure projects is a vital tool for the analysis of complex, integrated and multi-disciplinary systems. In sate of the art in engineering practice, the design is proven by analytical, empirical, or very complex numerical models Meschke et al. (2013). The generation of such numerical models, based on design and reports requires demanding manual intervention of experts and high computational costs. In recent years a BIM is increasingly gaining attention in infrastructure projects. Recent advances are allowing for multi-level representation of the built environment with the adequate Level of Detail/Definition (LoD) to support planning and analysis tasks of large tunnel projects.

Considering the fact that each project from the early design stage over to construction and the operation phase requires both information management and numerical analysis on different LoD, the need for a platform for information and numerical models becomes evident. However a solid link between such numerical models and BIM does not exist. This unique platform for information and numerical modelling would serve to support and optimize design, give reliable predictions and all in favour of minimization of risks and increased safety (see Fig 1).

The goal of the SATBIM is to develop a multi-level simulation model for tunnel-structure interaction integrated in the framework of Building Information Modelling (BIM) to support engineering decisions during the project life cycle and to allow for the evaluation and minimization of risks on existing infrastructure. This enables the general evaluation of the efficacy of such a multi-level simulation approach in other fields related to engineering design and risk assessment. SATBIM will provide an integrated platform for structural analysis, visualisation and optimization of the mechanized tunnelling process from early stages of the design over to the construction and the operation phase.

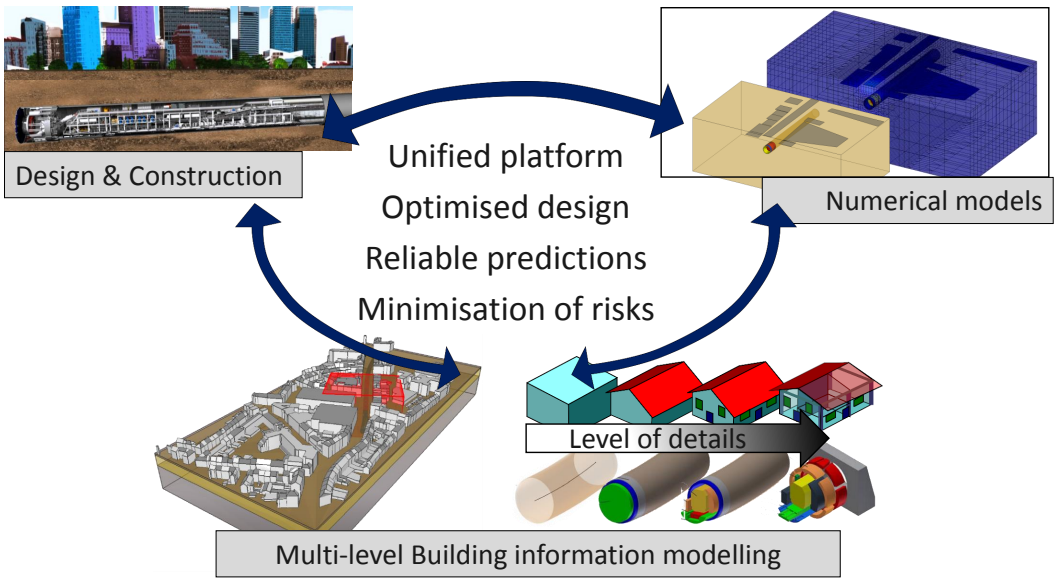


Figure 1: Concept for integrated platform for design and numerical analysis on different level pf details- SATBIM

This project deals with the multi-level information modelling of infrastructure and environment for simulating behaviour of such complex systems at various levels as required. The concept will be applied to assess the effects of underground construction, in particular mechanized tunnelling, onto existing infrastructure. This topic will serve as a feasibility study to use the concept more widely for other engineering design problems subsequently.

Why different level of details? The appraisal of different design alternatives is essential for ensuring optimal designs. In assessing the effects of various alternatives for tunnelling projects on the surrounding environment is a multi-disciplinary and complex problem. The current state of the art process is cumbersome and requires significant computing resources and time (sophisticated simulations including all details can take days or weeks to complete). This often leads to sub-optimal solutions and could possibly results in a solution that is not ideal in terms of its effect on the existing infrastructure. However, in the conceptual phase, a designer often only needs approximate estimations for many different scenarios, e.g. tunnel track alternatives. To ensure a seamless workflow, the computation time should be minimised. If preliminary analysis (with consideration of uncertainties) indicates the potential for hazards, a more detailed evaluation of the model is required.

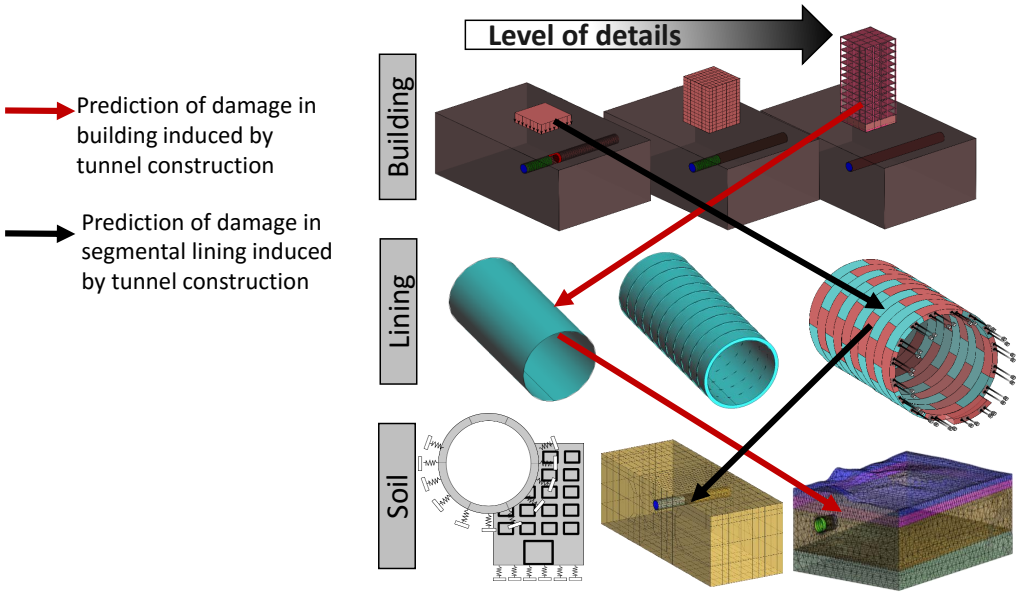


Figure 2: Alternatives for selection of LoDs for individual components based on objective of numerical analysis

The approach of the SATBIM project fulfils these requirements by dynamically generating simulation models from BIM models at the required level of detail for the specific problem to be solved. For example, minimising the overall risk of damage to buildings induced by tunnelling needs high LoD for structures and topology of the soil, however for the lining structure and it installation process the lower LoD is sufficient to achieve high accuracy of the solution (see 2(→)). On the other hand, estimating stresses in the tunnel structure

needs low LoD for buildings and high LoD for lining and it installation process while surface topology of the soil is not crucial for the accuracy of the results (see 2(→)).

1.1 Tools

SATBIM is platform for information modelling, numerical analysis and visualisation for optimisation of the tunnel design. To perform all those tasks different tools are to be employed and implemented in a user friendly SATBIM platform with high level of automation. To this end exiting software for Building Information Modelling Revit® and Dynamo, pre/-postprocessor GiD and open source platform KRATOS will be connected through a new software `SatBim Modeller` to perform tasks of modelling, analysis and visualisation. In the following section the main features of all individual software as well as the concept of their iteration is presented.

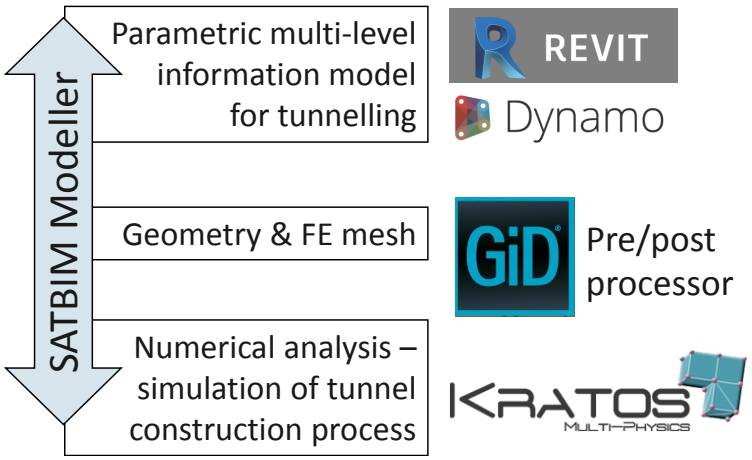


Figure 3: Tools in the SATBIM project

1.1.1 Revit® and Dynamo

Revit® is Software for BIM including features for architectural design, MEP and structural engineering and construction AUTODESK (2017).

It is possible to define custom Families for structural components. Typically, the custom loadable families are standard sizes and configurations of common components and symbols used in a building design, and are created using a family template that is provided in Revit®. One of the great features of Revit® is possibility to define parametric Families, i.e. assigning geometric or material properties as parameters which are than later changeable in Information model. This feature is used in Tunnel information model when creating custom families on different level of details for individual structural components.

Dynamo is Add-on for Building Information Modelling modelling in Revit® Jezyk and the Dynamo Development team at Autodesk (2016). A visual programming tool for designers that make use of external libraries or any Autodesk product that has an

API. Dynamo uses textual programming language DesignScript. DesignScript is dynamic, garbagecollected and associative language, and provides strong support for visual programming environment. Dynamo for Revit® extends building information modelling with the data and logic environment of a graphical algorithm editor. Its flexibility, coupled with a robust Revit® database, offers a new perspective for BIM.

Another advantage of Dynamo is that the Python programming language and can be embedded into existing applications. Python offers much more achievable methods for writing conditional statements and looping, or other operations in order to write algorithms for generation on custom nodes.

1.1.2 GiD

GiD is a universal, adaptive and user-friendly pre and post processor for numerical simulations in science and engineering CIMNE International Center for Numerical Methods in Engineering (2016). It has been designed to cover all the common needs in the numerical simulations field from pre to post-processing: a) Geometrical modelling (CAD); b) Mesh generation; c) Definition of analysis data; d) Data transfer to analysis software; e) Postprocessing operations and ; f) Visualization of results.

The GiD pre- and post-processor is used to perform the complete geometrical modelling of the tunnelling simulation model, including the generation of a geometrical representation of the ground, the tunnel, the lining system and the shield based on geometry and project parameters provided by Tunnel Information model. The geometrical entities imported as ACIS files (Volumes, surfaces, lines and points) are assigned to logical layers, to which the individual conditions and boundary conditions are defined. Furthermore, GiD is used to generate a suitable finite element mesh for the simulation model.

GiD allows for a customisation of its user interface by means of so-called *problemtype*s. These problem types define a set of application-specific conditions to be applied to all geometrical entity types as points, lines, surfaces, or volumes. Moreover, materials can be defined and assigned to individual volumes of the model. By means of an input file template, the mesh and the conditions can be translated to a user-defined format that can be read by the simulation kernel.

The definitions of a *problemtype* are stored in a number of different files. In general part, where the options of the underlying computation kernel are defined, are stored simulation parameters (e.g. linear solver category, the time integration scheme, gravity settings, parameters of the contact algorithm, output options and the choice of palletisation method),material models and boundary conditions (finite element conditions DIRICHLET and NEUMANN boundary conditions, water pressures, tying, contact etc.).

During the mesh generation, GiD automatically assigns the conditions to the mesh, according to their definition of layers. Thus, the definition of model conditions is independent from the actual mesh. This approach allows for a large amount of flexibility for the automatically generated meshes, since the user does not know about the mesh topology prior to meshing.

1.1.3 KRATOS

KRATOS is a framework for building multi-disciplinary finite element programs. It provides several tools for fast implementation of finite element applications. KRATOS is a modular, object-oriented framework for the development of finite element applications for multi-physics simulations Dadvand et al. (2010); International Center for Numerical Methods in Engineering (CIMNE) (2014). The aim of KRATOS is to allow for a simple yet robust and efficient implementation of various finite element algorithms. The main requirements are: generality in design and implementation, flexibility and extensibility, a good level of reusability of libraries and tools and good performance and memory efficiency.

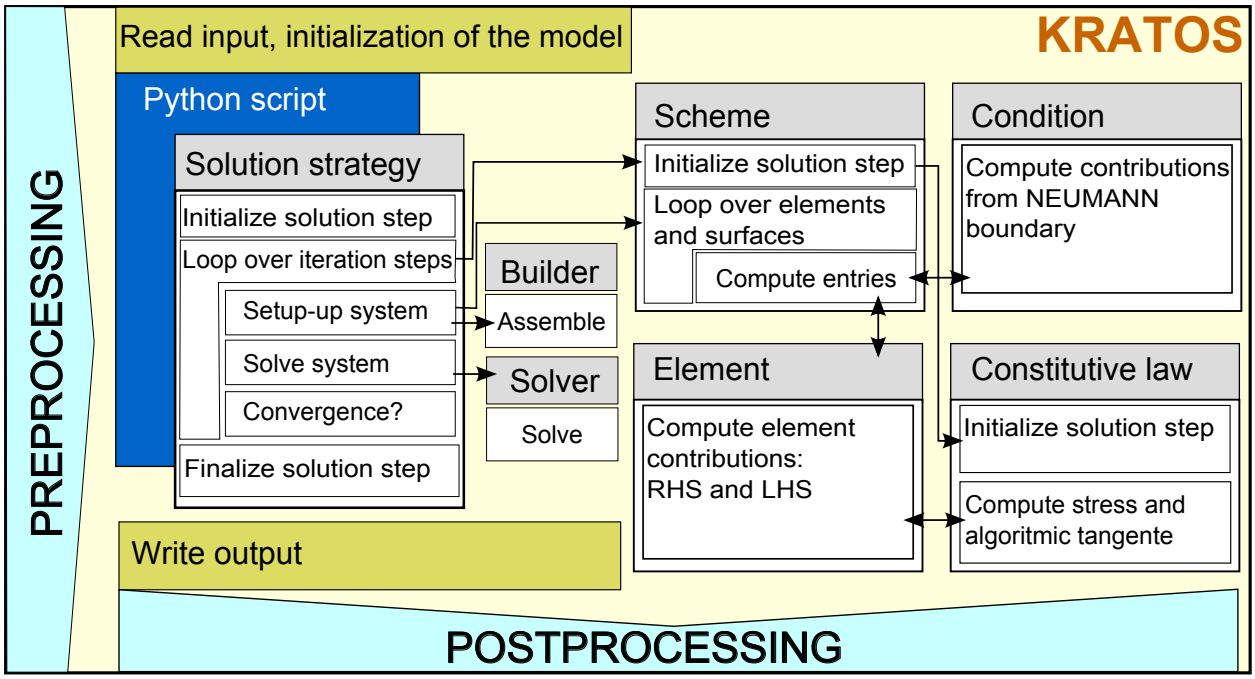


Figure 4: Object-oriented implementation of the multi-phase formulation for partially saturated soil using the FE package KRATOS Dadvand et al. *(2010)*

The components of KRATOS can be roughly grouped into two categories: the Kernel and the Applications. The Kernel is the numerical core, while the Applications are the actual implementations of the FE algorithms for different types of analysis (fluid, structural mechanics, mortar etc.). The Kernel provides the basic infrastructure and general numeric tools such as prototypes for the classes used in KRATOS (*Element*, *Node*, *ConstitutiveLaw*, etc.), the central database used to store the mesh, the global variables, and the solution step variables. The Kernel is the core based on which the different applications are built. Moreover, providing a common infrastructure for all applications, it also enables communication between the applications.

The different algorithmic levels of a FE analysis are separated from each other and hierarchically organized. This layered solution scheme is illustrated in Figure 4. The outermost loop is directly controlled by the user from a Python script, where the number of load or time steps (solution steps) and boundary conditions are defined as well as all manipulations

of the model. Within the solver method of the strategy object, a NEWTONs loop is realized including the assembly of the computed elemental and surface entities, the solution of the given linear equation system, the incremental update of the Degree of Freedom (DOF) and a check of the convergence of the iteration.

In order to allow for a sufficiently flexible design of the simulation workflow, natively compiled libraries written in high-level languages (e.g. `C++` or `FORTRAN`) are imported into Python scripts. In contrast to this flexibility, the computationally parts of the software—data handling, assembling, solving—are implemented as natively compiled software libraries.

Applications For the simulation of mechanized tunnelling, several parts of KRATOS have been utilized. The core of the simulation software is formed by the *Structural Application* and *Mortar Application*, while some additional algorithm are defined in **ekate**'s *Auxiliary Application*. The *Structural Application* includes the implementation of elements, constitutive laws, solution strategies, specific conditions and utilities for general applications in structural mechanics, *Mortar Application* contains formulation of interface including tying and contact, while in the **ekate** *Auxiliary Application* those components are specifically defined for shield tunnelling problems STASCHEIT (2010).

1.1.4 `SatBim Modeller`

A fully automatic modeller for arbitrary tunnel alignment provides a high degree of automation for generation, setup and execution of the simulation model connecting the Multi-level TIM with simulation software KRATOS trough GiD pre/post processor. All information about the geometry, process parameters and material parameters are obtained from TIM as will be explained in Section 2.7. The core of the modeller is the Python script `SatBim Modeller`, which reads all defined specifications of the model from ACSI files and model, material and semantic parameters and automatically generates an FE simulation model and a simulation script for the tunnelling problem. This modeller will be in details explained in Section 3.1.

A Python script is used to define the actual simulation model for tunnelling. The Python script `SatBim Modeller` generates the geometry of the model and FE mesh with respective boundary conditions and logical layers. Moreover, another Python script called *Simulation script* is used to control the tunnel simulation workflow. Python scripts serve as an interface between the actual FE model and the simulation software, providing a customized setup of the Kernel and preparing the model for the simulation. The Python script is designed as an own module that defines all necessary functions needed for the simulation workflow.

1.2 How all this works together?

In Section 1.1 the tools for generation of Information and Numerical simulation model for urban tunnelling are described. In order to have a consistent modelling on different level of details for all individual components and the system, a tool for the automation in data exchange and the model generation is developed - a `SatBim Modeller`.

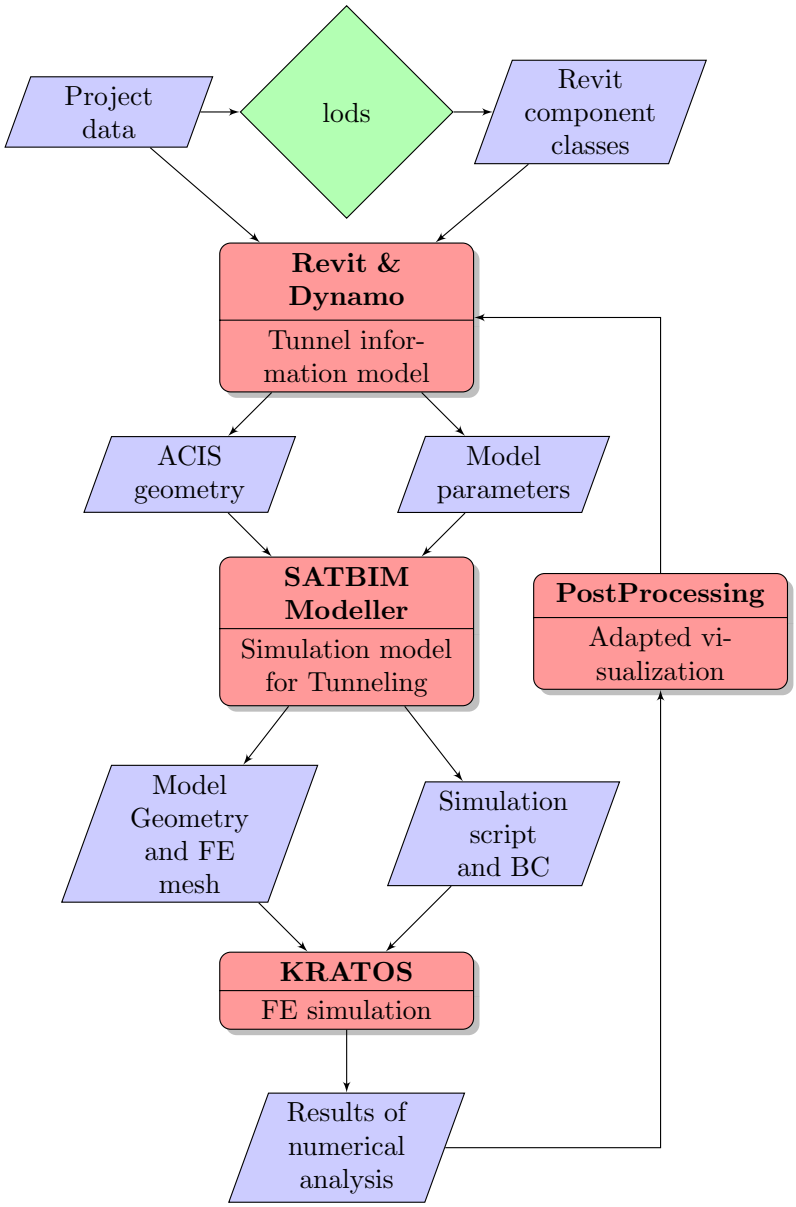


Figure 5: Algorithm of SATBIM work-flow

The Algorithm 5 shows the flow of the data from Tunnel Project where the specific requirements of the project are defined, over Information model generated using Revit® and Dynamo to GiD and `SatBim Modeller` where Finite Element model is generated together with simulations script and finally up to simulation software KRATOS where the numerical model is calculated. The results of the numerical simulations (e.g. settlements) and their impact on the existing environment (e.g. risk of damage) are than visualised within the TIM to enable a comprehensive and quick understanding of effects of design actions on the stability and safety of the existing environment.

The User defines parameters based on Tunnel Project requirements (tunnel diameter, alignment, thickness of lining structure, materials, semantic data etc.) in Dynamo interface

and makes decision about the LoDs of the component. Pressing ***start*** the rest of the model generations and calculations flows without interoperation or additional assignment of parameters. From the information model generated in REVIT®, using DYNAMO the geometry of complete model will be exported in ACIS format, while global model parameters, as well as material and semantic parameters of each component will be exported as text files. `SatBim Modeller` reads those files, executes GID and perform all operations in order to create tunnel model with geometry identical to TIM model, however, assigning proper boundary conditions according to model parameters and generating FE mesh. In addition, `SatBim Modeller` generates Python script to prescribe boundary condition and geometry change during tunnel contraction process. GID employs a simple script language that allows for the generation of arbitrary ASCII files that contain all necessary data for the simulation (the mesh information, boundary and initial conditions, material properties, and model parameters) in the form adjured to simulation kernel KRATOS.

Naturally, the User may edit any of the intermediate steps and modify some parameters if, based on it engineering judging may improve the results (e.g. FE model for re-meshing).

2 Tunnel information model

In this section, a concept for parametric and multi-level modelling of shield bored tunnels within urban environments using parametric BIM tools, particularly Revit® and Dynamo, is presented and employed as a basis for structural analysis on different LoDs. To this end a parametric representation of each system component (soil with excavation, tunnel lining with grouting, TBM and buildings) is developed in the information model for three LoDs (high, medium and low) and used for automated generation of the numerical models for the tunnel construction process and soil-structure interaction. The platform enables flexible, user-friendly generation of the tunnel structure for arbitrary alignments based on predefined structural families for each component, supporting the design process and at the same time providing an insight into stability and safety of the design. This model, with selected optimal LoDs for each component w.r.t. the objective of the analysis, will be further used for efficient design and process optimization in mechanized tunnelling.

2.1 Multi-level approach

For planning of large infrastructure projects such as inner-city subway tracks, engineers have to conduct investigations and develop models with consideration widely differing scales, ranging from the kilometre scale for the general routing of the track down to the centimetre scale for detailed design of connection points, as illustrated in Figure 6. Planning and design phase require analysis, modelling, visualization, and numerical analysis which are performed using different tools such as Geographic Information Systems (GIS), Building Information Modelling (BIM) and Numerical analysis tools. However, to enable all those tasks with high consistency when changing the scales a sound foundation of handling multi-scale representations is required. Although multi-scale modelling is already well established in the GIS field van Oosterom and Schenkelaars (1995), it has been addressed just recently in BIM Borrmann et al. (2014) to support planning and analysis tasks of large tunnel projects. However, multi-scale concepts are also much needed in the Numerical Analysis context, as the planning process typically provides only rough information in the early stages and increasingly detailed and fine-grained information in later stages and the proof of design has to be conducted through all those phases.

To meet this demand, this section presents a comprehensive concept for incorporating multi-scale representations with building information models, with a particular focus on the geometric-semantic modelling of shield tunnels, in the form suitable for direct application of the model for further numerical analysis.

Employing a multi-scale representation is particularly important in the context of planning tunnelling projects as they typically have a very large extent (several kilometres) and at the same time are subject to design decisions in the range of only a few centimetres to provide the desired connections and avoid spatial conflicts. Although the multi-level approach is well established for modelling of buildings and despite the evident multi-scale characteristics of tunnel planning process, there are only limited number of models enabled multi-level approach for support of tasks of planning data, design and analysis. The concept of describing buildings and infrastructure facilities using multiple levels of detail is well established in the

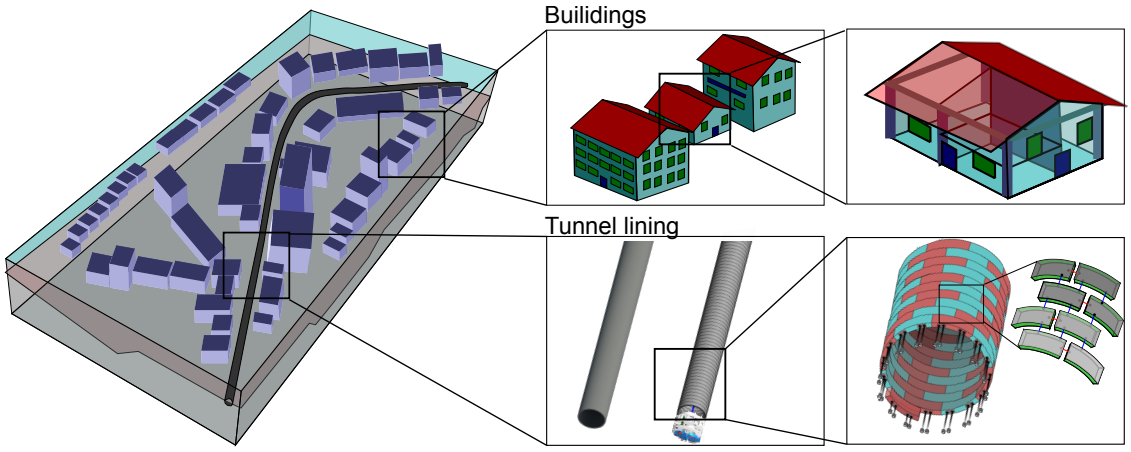


Figure 6: In large infrastructural projects planning, design and analysis is performed on different scales (from kilometre to centimetre), which requires modelling on different LoDs.

GIS field van Oosterom and Schenkelaars (1995), where in general two approaches are used:

- bottom-up approach: coarser representations are abstracted from detailed data by a process called generalizationForberg (2007)
- top-down: dynamic generation of coarse representations from finer ones and replace finite LoDs by continuous onesDöllner and Buchholz (2005)

Moreover, multi-level modelling has become standard for modelling buildings Xie et al. (2012); Biljecki et al. (2016) as well as an integral part of CityGML Kolbe and Groeger (2004). However, there is only a very limited number of examples where the multi-level approach is used to support the planning, design, and analysis of large tunnel projects Borrmann et al. (2014).

2.2 Generation of information model

2.2.1 Components

Mechanized tunnelling is an established and flexible technology for the construction of tunnels in urban areas. The tunnel construction process in soft soils causes short and long-term ground deformations, resulting from a disturbance of the virgin stress state of the soil and the changing pore water conditions due to the heading face support, the shield skin friction and the gap grouting. Furthermore, depending on the tunnel advancement procedure and the development of soil deformations around the tunnel, the influence of the tunnelling on structural deformations of the tunnel lining and the machine itself can vary significantly. Therefore, numerical modelling of the shield tunnelling process requires both material and process to be properly modelled.

The shield-supported tunnel advance beneath the groundwater table in soft soil requires permanent support of the surrounding underground to prevent the groundwater from flowing into the construction site. A realistic model to be applied during construction and in the design phase needs to represent all components of the tunnelling process relevant for the prognosis of the response of the surrounding soil during excavation. These components include (see Fig. 7):

- proper model for the soil (Section 2.3),
- the TBM and the hydraulic jacks(Section 2.6),
- the segmental lining with the support measures applied at the tunnel face and at the tail void (Section 2.4) and,
- existing infrastructure (Section 2.5).

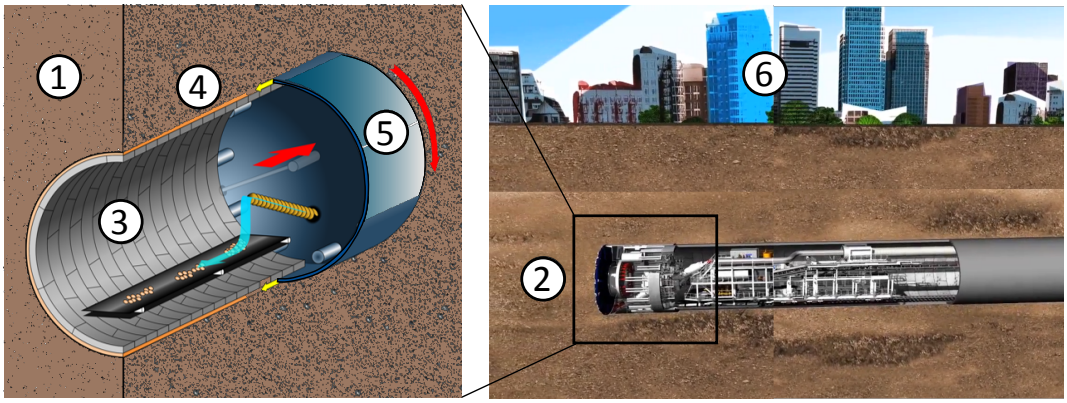


Figure 7: Main components of urban tunnelling process: 1) Surrounding soil; 2)Tunnel alignment and excavated soil; 3)Segmental tunnel lining; 4)Tail gap grouting; 5)Tunnel boarding machine (TBM); 6)Existing Infrastructure

Besides geometric data, semantic data as well can be represented on different level of detail. The face pressure and the grouting of the tail gap strongly interact with the soil skeleton and the pore fluids. Consequently, these components have considerable influence on the spatial distribution and temporal evolution of the settlement trough and on a possible failure of the soil. Therefore, the proper multi-level representation of :

- heading face support,
- pressurization of the tail gap with grouting and,
- TBM advance rate

is to be considered in the SATBIM framework.

2.2.2 Multi-level model structure

In all components of the simulation model can be represented with different level of details. The Example given in Fig 8a shows how both, a building and a TBM can be represented with geometries characterized from only basic volumetric representation, up to highest details.

The concept different LODs refers to different levels of abstraction for a linear infrastructure facility following the well-defined multi-scale concepts of the GIS. In this concept the information model of the component contains all the different LODs at once, allowing the planner to dynamically switch between them and choose the appropriate level of abstraction for a particular planning task.

In the SATBIM Project for each component of urban tunnelling process defined in Section 2.2.1, a three level of details are defined:

- Low
- Medium
- High

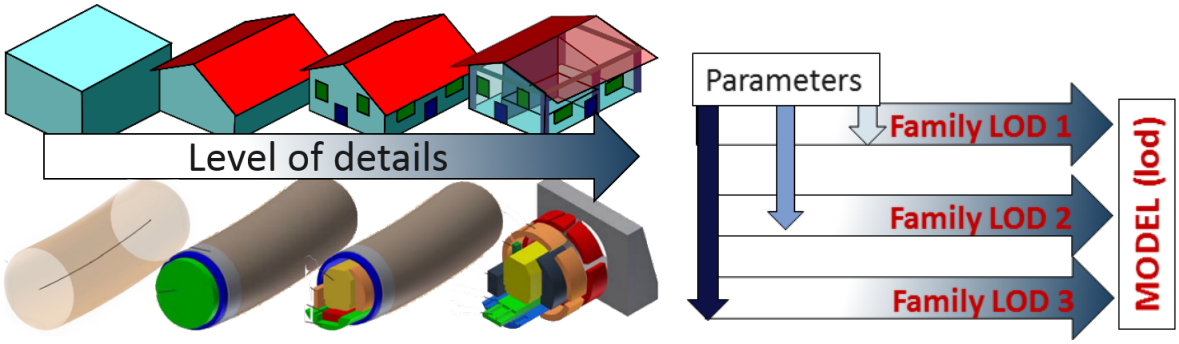
In general, in the lowest level of detail, non-volumetric representation of the component is assumed, since in corresponding numerical model a components are not represented with structural models but instead with the analytical or empirical models assigned through the sets of boundary conditions. In medium LoD for each component the volumetric representation is established, where the component is "occupying" the exact volume, however the geometry is simplified, i.e. approximated. Finally the highest LoD includes more details about the actual geometry of the component. However, the components such TBM do not include the details of the machinery and the equipment inside the shield, and therefore, the even higher representation can be introduced as an extension.

It must be noted, that the Information model structure as well as the `SatBim Modeller` is very flexible, and if user requires, a further level of details can be added without interrupting the existing model structure.

For each component, on each level of detail, a Family for the corresponding component is defined. In order to keep consistency between different level of detail, a parametric consistency between Families is defined as shown in Fig 8b. The full set of parameters defining a component is needed for definition of the highest LoD, while only a portion of the same list is used for definition of lower LoD. This parameter dependency allows for automated preservation of the consistency of the multi-scale model.

Families Families are classes of elements in a category. A family groups elements with a common set of parameters (properties), identical use, and similar graphical representation. Different elements in a family may have different values for some or all properties, but the set of propertiestheir names and meaningis the same. Revit® uses the following kinds of families:

- **Loadable families** can be loaded into a project and created from family templates. One can determine the set of properties and the graphical representation of the family.
- **System families** are not available for loading or creating as separate files. One can use the predefined types to generate new types that belong to this family within the project. For example, the behaviour of a level is predefined in the system. However, it is possible to create different types of levels with different compositions. System families can be transferred between projects.



a b

Figure 8: (a) Multi level representation of the geometry ; (b) Parameter dependency between different level of details for individual components

- **In-place families** define custom elements that are created in the context of a project. Unlike system and standard component families, you cannot duplicate in-place family types to create multiple types.

For the Tunnel information model, a Loadable families for different components are used. When establishing new Families of different tunnel components we took advantage of the parametric modelling, where family geometry and material properties are described either as Family type or instance parameters, which allows to access and change those parameters trough Dynamo interface, and make the model fully parametric.

Types of level-based data As shown in Fig. 9, not only geometry, but all different types of data characterizing Tunnel model can be represented on different level of details.

Simulation	Level of details
Geometric representation	• 1D (analytical solutions) • 2D (plain strain) • 3D
Structural Components	Soil, lining, buildings, TBM
Material Models	• Linear elastic • standard plasticity • advances damage and failure models
Time Discretization	• One time step • Consolidation • Real-time simulation
Support Measures	• Loading • Time-variant boundary conations • Structural models
Excavation method	• Volume loss • Step -wise excavation • Additivity

Figure 9: Different types data represented in level-based structure: geometry, structural components, material models, semantic data, excavation model

For instance, in terms of information modelling, not only geometry but also semantic information can be represented on different LoDs, where for instance time-variant data can be represented either by average quantities per amount of time or in finer (full) scale. In terms of numerical modelling, first of all we can decide about the dimension of the analysis, choosing full three-dimensional model, two-dimensional approximation or even reducing to analytical 1D solutions. Moreover, materials can be described using models of different complexity, from smile linear elastic models, and then stepwise increasing the complexity by including different non-linearities. Finally, in simulation model itself the process can be represented with different amount of approximations and simplifications.

2.3 Lining

2.3.1 Introduction

For shield tunnelling technologies, the use of segmental lining as the final tunnel support and lining is a worldwide standard (Maidl et al., 2012). First, the use of segmental lining is essential to fulfil the required construction speed due to the short available build time. Secondly, the shield thrust is enabled by using the segmental lining as an abutment for the hydraulic jacks, where the thrust forces are resisted in the tunnel lining.

In order to support the excavated soil and to ensure the tunnel stability behind the shield the precast lining segments are installed in subsequent rings. The gap between the segmental lining external diameter and the boundary of excavated soil is continuously filled with grouting mortar to prevent soil deformation towards the tunnel and stabilize the tunnel

tube. A complete segment ring is cone-shaped in order to accommodate curve radii of up to around 150 meters. The narrowest spot on the segment ring is usually at the so-called key segment, which, as the last wedge element, closes and braces the ring. The key elements of the segments can be adjusted by means of groove and tongue or cam and plug systems. By contrast, segments with flat sides are of less complicated design (her, 2016).

Segmental lining is characterized with a high degree of jointing, where the joints can be differentiated into longitudinal joints (between the segments in a ring) and ring joints (between the rings) (Maidl et al., 2012). As reported by many authors, the joints in a segmental lining have a considerable influence on the structural behaviour of the tunnel lining (Cavalaro et al., 2011; Hefny and Chua, 2006; Teachavorasinskun and Chub-Uppakarn, 2010). In fact, the joints have smaller stiffness in comparison with the segment itself, and therefore, undergo more motions and have a reduced bending moment bearing capacity. The design code of the Japanese Society of Civil Engineering empirically recommends that segmental joints should be designed to carry only 60–80% of the maximum bending moment carried by the main segment. Hence, the lateral confinement from the surrounded soil has to be adjusted according to the adopted moment reduction factor (Koyama, 2003).

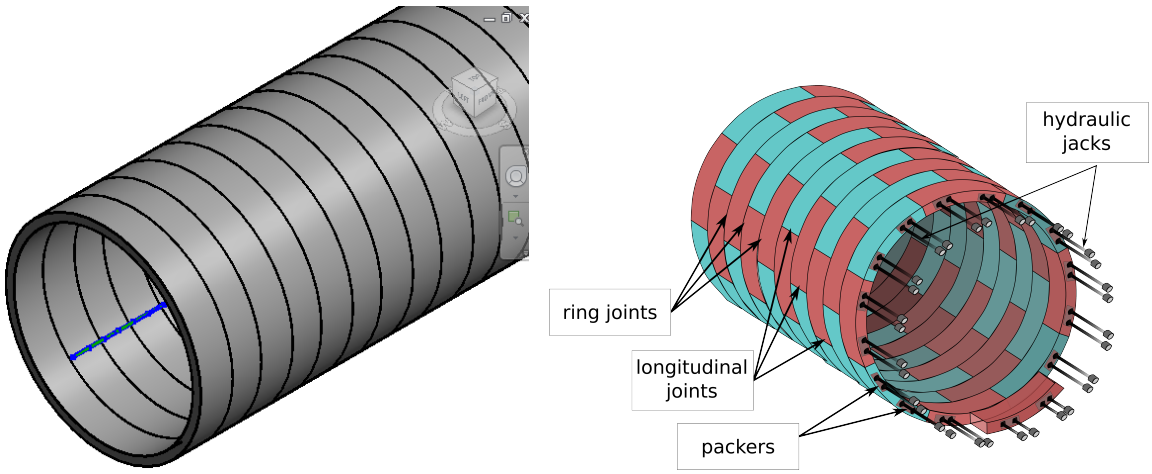


Figure 10: (a) Segmental Lining; (b) Design of segmental lining and basic characteristics

The universal ring Each tunnel project has special tunnel lining requirements, depending on the diameter, the soil conditions and the routing, so that a safe and durable tunnel structure can be produced in order to withstand demanding use for up to 100 years. The geometry of the entire ring and its individual segments as well as the arrangement of the joints and their sealing must be designed such that the can be easily moulded for designed tunnel alignment. Moreover, perfectly manufactured segments must be supplied just-in-time for tunnel production. Hence, in order to have modular segments and to fulfil the requirement of just-in-time for production, the solution is to go with a universal rings (see fig. 11). In most cases, the universal segment ring is made of several segments of the same size and of one smaller segment at the end -the key-stone closing the ring. The universal-ring sides are

not made parallel but tend to form an angle that allows the tunnel to bend according to the relative angular position of each segment. The Universal ring is characterized with average ring length L_r, inner and outer radius of the ring (r_{inner} and r_{outer}), angle describing tapered geometry of the ring α, number of segments and their division within ring.

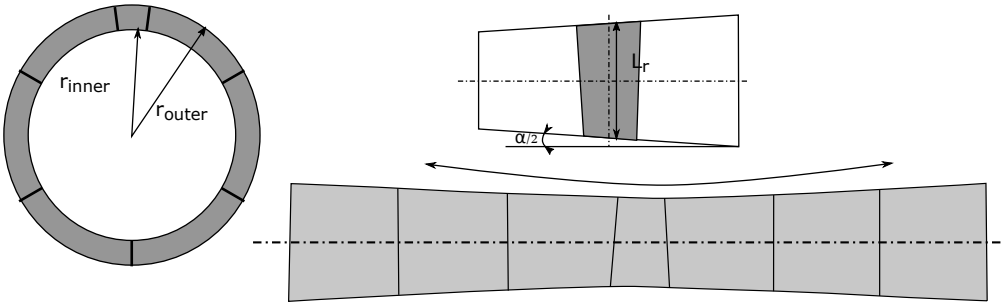


Figure 11: Tapered geometry of universal ring (6+1)

2.3.2 Alignment

The designed alignment of the tunnel is formed by adjusting the rotations of the rings as shown in Fig 12(a). For the curved parts the rings are placed by lining up the key, for straight parts the rings are switched from upward key to downward key. The relative positioning of keys can be played upon to modify the curved radius. In Fig. 12(b) the deviation for the alignment which can be achieved for the given geometry of the universal ring from the designed tunnel alignment.

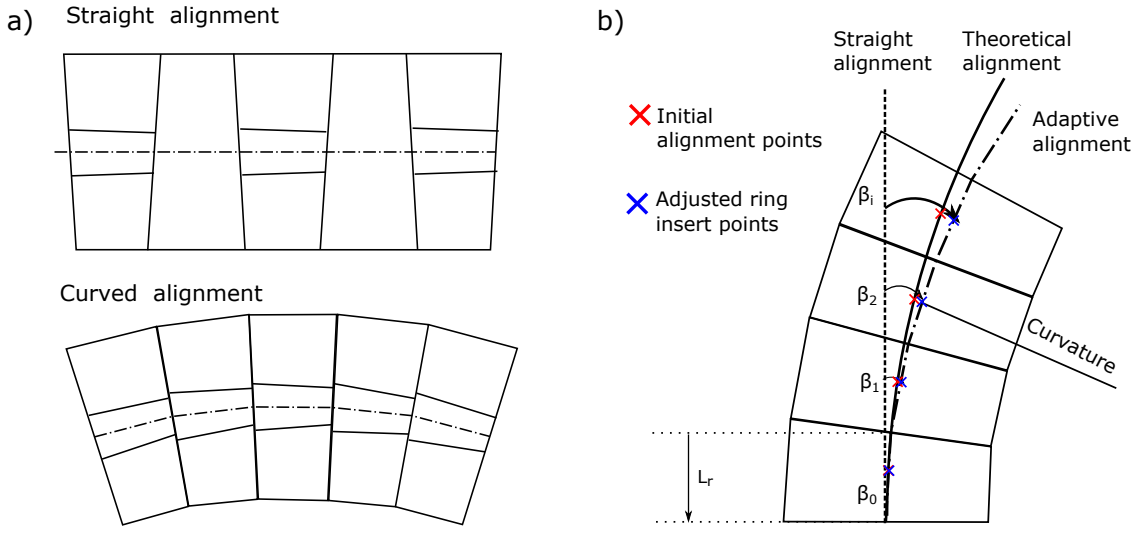


Figure 12: Forming the tunnel alignment based on the universal ring geometry: a) Straight and Curved alignment; b) Design Alignment vs. Adaptive alignment

The algorithm for calculation of adaptive alignment is implemented in DYNAMO as shown in Fig. 13. The main parameters for determination of the actual tunnel alignment are: 1)

actual design path; 2) tunnel ring length L_r and 3) curvature (κ) of the ring (optionally the angle describing tapered geometry of lining can be assigned α related to L_r

$$\alpha = \arccos \frac{1 - L_r^2}{2 \cdot \kappa^2}$$

). The actual algorithm calculating the best position of the lining rings w.r.t. design path, giving the new adjusted tunnel alignment is implemented as Python script (see Algorithm 1). The model for calculation of the alignment is parametrised, and therefore all parameters can be changed and Dynamo will automatically recalculate the adaptive alignment.

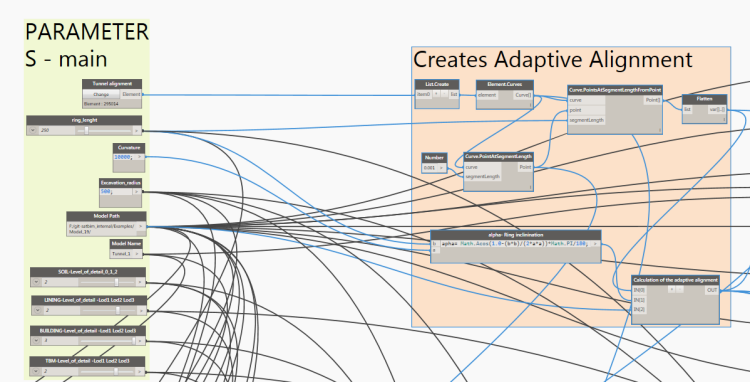


Figure 13: Implementation of generation of adaptive alignment in Dynamo. *The actual algorithm for the determination of the adaptive alignment is given in 1*

Briefly described the procedure works as follows:

- The User defines Design tunnel alignment in Revit®
- In Dynamo in parameter field User defines L_r and κ
- Along the defined Design tunnel alignment a set of points will be defined as initial lining ring centre points
- Algorithm 1 reads initial lining ring centre points and based on defined lining parameters generates the list of adjusted centre points of lining rings.

In the Algorithm shown above, based on the set of initial lining-ring-centre-points and parameters L_r and κ, a new adjusted lining-ring-centre-points list will be created by determining the rotation of the lining ring such that the adjusted lining-ring-centre-point has minimal distance from initial lining-ring-centre-points. As an output the list of adjusted centre points of lining rings ($list\_points$) as well as list where the rotations of the rings in the plain normal to alignment ($list\_or$) are stored.

The algorithm 1 is for the determination of ring rotations in 2D, since the alternative rotations are only 0 or 180°, i.e. we rotate the rings such that it turns left or right. Hoverer, since in reality, the tunnel alignment is mostly three-dimensional (3D), the different segmentation of lining rings will determine the number of possible rotations and therefore enable adjustment to 2D alignment. In other to achieve 3D alignment,an additional variable θ which determines the rotation in ring plane is introduced. Therefore, if the segmentation of the

Listing 1: Algorithm for calculation of adaptive tunnel alignment in 2D

```
a=curvature #kappa
L_r=ring_length
apha= math.acos(1.0-(L_r*L_r)/(2*a*a))*Math.PI/180
beta=initial angle
for i in range (0,len(list_points_initial)):
        lx.append(list_points_initial[i].X)
        ly.append(list_points_initial[i].Y)
        lz.append(c=list_points_initial[i].Z)

#calculate adapted alignment
for i in range (2,len(list_points_initial)):
        x1_n=lx_new[i-1]+L_r*math.cos(beta)
        y1_n=ly_new[i-1]+L_r*math.sin(beta)
        if list_or[i-2]==1:
                x2_n=lx_new[i-1]+L_r*math.cos(beta-alpha)
        else:
                x2_n=lx_new[i-1]+L_r*math.cos(beta+alpha)
        if list_or[i-2]==1:
                y2_n=ly_new[i-1]+L_r*math.sin(beta-alpha)
        else:
                y2_n=ly_new[i-1]+dist*math.sin(beta+alpha)
        dist1=  math.sqrt((x1_n-lx[i])*(x1_n-lx[i])+(y1_n-ly[i])*(y1_n-ly[i]))
        dist2=  math.sqrt((x2_n-lx[i])*(x2_n-lx[i])+(y2_n-ly[i])*(y2_n-ly[i]))
        if dist1<dist2:
                lx_new.append(x1_n)
                ly_new.append(y1_n)
                beta=beta
                if list_or[i-2]==1:
                        list_or.append(0)
                else:
                        list_or.append(1)
        else:
                lx_new.append(x2_n)
                ly_new.append(y2_n)
                if list_or[i-2]==1:
                        list_or.append(1)
                        beta=beta-alpha
                else:
                        list_or.append(0)
                        beta=beta+alpha
        list_beta.append(beta)

        #create list of points for new adapted alignment
        for i in range (0, len(lx_new)):
        point=Point.ByCoordinates(lx_new[i],ly_new[i],lz[i])
        list_points.append(point)
        ##############################################################
```

rings is $6 + 1$, there are six possible rotations in ring plain, and $\Delta\theta = 60°$. Also, there are alternatives in ring installation strategy, such that for instance the next ring can be turned only for one $\Delta\theta$ clockwise or anticlockwise, or it can be turned alternatively in any of six positions in plain.

Regardless, of the ring rotation strategy, for any 3D design alignment, it is possible now to determine the adjusted alignment flowing the geometrical transformation outlined below. Starting with the initial ring and it centreline coordinate $x_{n-1}, y_{n-1}, z_{n-1}$, and adding a new ring on it, we move to the new alignment point for certain differential displacement:

$$x_n = x_{n-1} + \Delta x_n \qquad y_n = y_{n-1} + \Delta y_n \qquad z_n = z_{n-1} + \Delta z_n \tag{1}$$

This differential displacement depends on ring geometrical properties L_r and α, as well as the rotation of the ring in ring plain θ as follows:

$$\Delta x_n = L_r \cdot cos(\beta_{n-1} + \frac{\alpha}{2} \cdot cos(\theta_{n-1}) + \frac{\alpha}{2} \cdot cos(\theta_n)) \cdot cos(\gamma_{n-1} + \frac{\alpha}{2} \cdot sin(\theta_{n-1}) + \frac{\alpha}{2} \cdot sin(\theta_n)) \tag{2}$$

$$\Delta y_n = L_r \cdot sin(\beta_{n-1} + \frac{\alpha}{2} \cdot cos(\theta_{n-1}) + \frac{\alpha}{2} \cdot cos(\theta_n)) \cdot cos(\gamma_{n-1} + \frac{\alpha}{2} \cdot sin(\theta_{n-1}) + \frac{\alpha}{2} \cdot sin(\theta_n)) \tag{3}$$

$$\Delta z_n = L_r L_r \cdot cos(\beta_{n-1} + \frac{\alpha}{2} \cdot cos(\theta_{n-1}) + \frac{\alpha}{2} \cdot cos(\theta_n)) \cdot sin(\gamma_{n-1} + \frac{\alpha}{2} \cdot sin(\theta_{n-1}) + \frac{\alpha}{2} \cdot sin(\theta_n)) \tag{4}$$

With that, we achieve new inclination of the ring in global coordinate system in XY plain β and YZ plain γ:

$$\beta_{n+1} = \beta_n + \frac{\alpha}{2} \cdot cos(\theta_{n-1}) + \frac{\alpha}{2} \cdot cos(\theta_n) \tag{5}$$

$$\gamma_{n+1} = \gamma_n + \frac{\alpha}{2} \cdot sin(\theta_{n-1}) + \frac{\alpha}{2} \cdot sin(\theta_n) \tag{6}$$

Finally, having the start point of the ring mid point of the ring is calculated as:

$$x_n^{mid} = x_n + \frac{L_r}{2} \cdot cos(\frac{\alpha}{4}) \cdot cos(\beta_n + \frac{\alpha}{4} \cdot cos(\theta^r)) \cdot cos(\gamma_n + \frac{\alpha_n}{4} \cdot sin(\theta_n)) \tag{7}$$

$$y_n^{mid} = y_n + \frac{L_r}{2} \cdot cos(\frac{\alpha}{4}) \cdot sin(\beta_n + \frac{\alpha}{4} \cdot cos(\theta^r)) \cdot cos(\gamma_n + \frac{\alpha_n}{4} \cdot sin(\theta_n)) \tag{8}$$

$$z_n^{mid} = y_n + \frac{L_r}{2} \cdot cos(\frac{\alpha}{4}) \cdot cos(\beta_n + \frac{\alpha}{4} \cdot cos(\theta^r)) \cdot sin(\gamma_n + \frac{\alpha_n}{4} \cdot sin(\theta_n)) \tag{9}$$

$$\tag{10}$$

The algorithm for calculation of 3D tunnel alignment and it Python implementation is given in Appendix in listing 14. Having this implementation in Dynamo now we are able to achieve any tunnel trace in space using only one universal ring. The agreement between design and adapted tunnel is shown in Figure 14. Form this comparison is clear that using the algorithm and formulation described above we can achieve any design alignment with high accuracy.

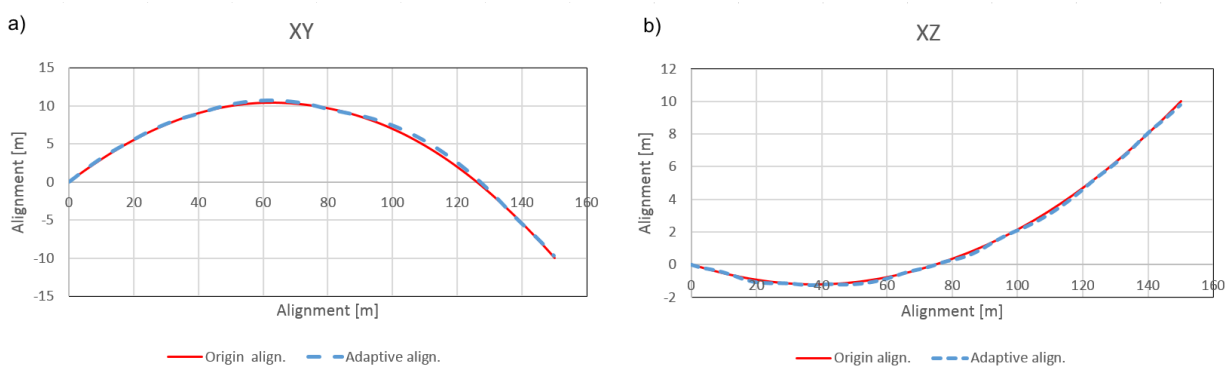


Figure 14: Comparison between design alignment and newly calculated adapted alignment based on universal ring in D for: a) XY plain; b)XZ plain

2.3.3 Families

As described in Section 2.2.2 for each individual component the custom families are defined for each level of detail. In Fig. 15a a families for lining component are shown as well as logic of the parameter dependency between different LoDs. Since the main idea is that each LoD occupies the same space in the information model (either volume or surface in footprint) the different LoDs are sharing geometry and material parameters such that the highest level of detail will be defined by the complete parameter list while each lover LoD will be defined by just a portion of the parameter list. For example in the tunnel lining model, each LoD will be defined by the parameter outer diameter r_{outter}, however, since only highest level knows about the segmentation of ring, it will contain parameter "number of lining segments".

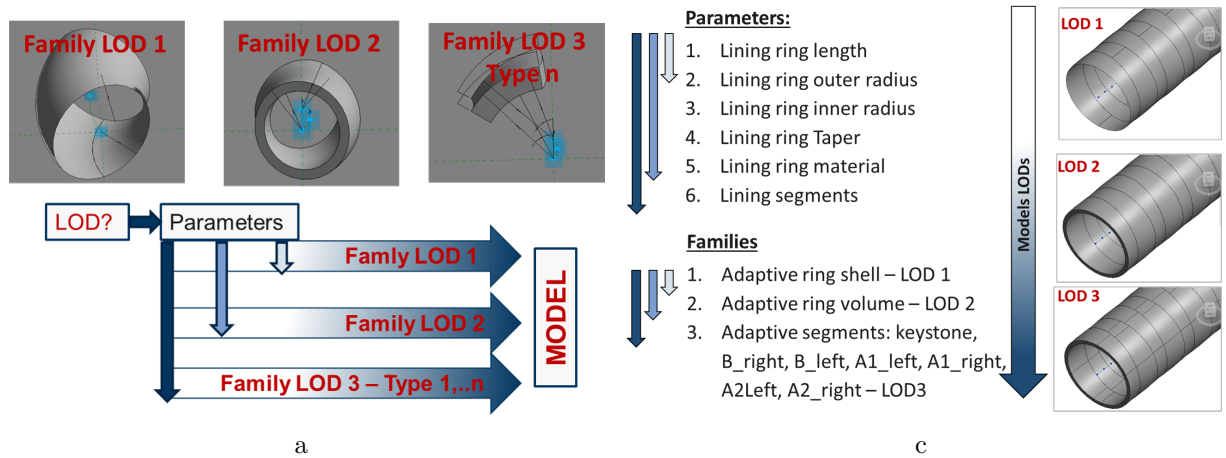


Figure 15: Lining multi-level information model: (a) Families for different LoDs; (b) Parameter dependences

To define custom Loadable families of the Lining component on different LoDs as a template an "metric generic adaptive family" is used. This type of family allows to define insert points

as parameters which later will be used to define family Instance. This approach is adopted in order to be able to use previously calculated list of adjusted centre points of lining rings (see Alg. 1) to inset family instances.

2.3.4 Grouting

The gap that evolves behind the shield between the erected lining tube and the soil is grouted using a pressurized mortar simultaneously to the advance of the shield to prevent the soil from moving into this gap (see Figure 16). Moreover, grouting is securely connecting the tunnel lining with the surrounding ground. Therefore, the tail void is filled with pressurised grouting mortar. After hydration of the mortar it forms a solid connection between the lining and the ground. The grouting mortar is prevented from flowing into the shield by means of sealings between the lining and the shield tail. The tail void grouting has a considerably large effect on the change in the initial stress state of the soil around the tail, which finally causes surface settlements.

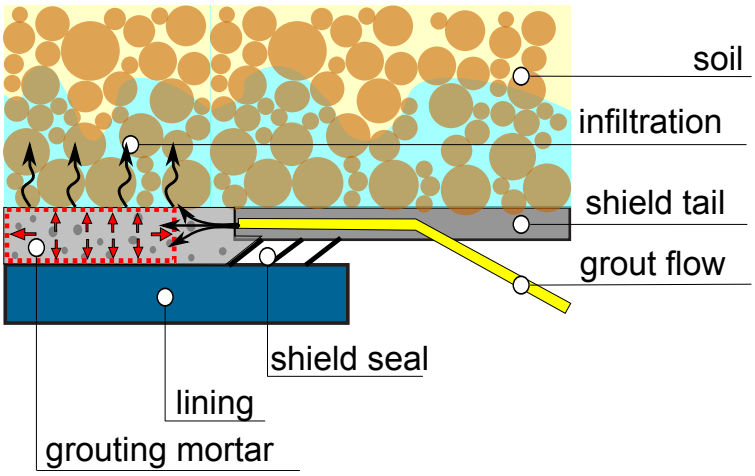


Figure 16: Tail gap grouting: sketch of the grouting of the annular gap between the lining and the surrounding soil with a pressurized mortar inflow through the shield skin

From the discretion of the grouting mortar we can see that this is basically a volume between lining and the soil, and therefore in terms of information modelling it is a dependent component because it exists only for certain levels of details of the soil and lining component. For that reason it is decided not to implement separate families for the grouting component, but regather a geometry representing the grouting. This geometry is defined based on the lining parameter - outer radius (r_{outer}), soil parameter excavation radius (r_{exc}) and calculated tunnel alignment. The geometry representing grouting is a volume ring filling exactly the gap between the lining ring outer surface and the soil excavation surface.

2.3.5 Dynamo lining and grouting generator

After the adjusted tunnel alignment is calculated (see Alg. 1), and custom system families for lining on different LoDs are defined (see Fig. 15a), the lining structure is generated

importing Family instances of the Lining for the selected LoD along the alignment (see Fig. 13). In Fig. 17 (a) the DYNAMO node shows how the families are loaded into current project and, based on given parameters, lining structure and grouting volumes are generated (Figs. 17 (b) and (c)). The user can dynamically change any model parameter (alignment, lining parameters, soil) and the tunnel information model is automatically adjusted.

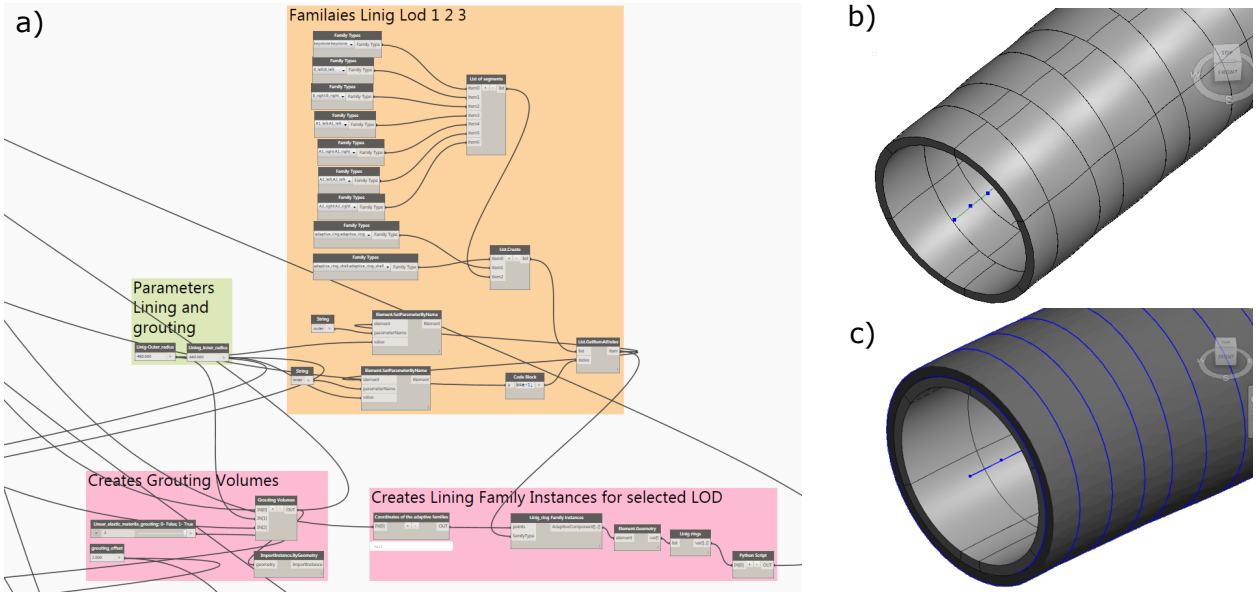


Figure 17: Generation of lining structure and grouting volumes: a) Dynamo nodes; b) Lining (LoD3); c) Grouting

The calculation of three-dimensional alignment described in Section 2.3.2 and shown in Figure 14 enables generation of arbitrary alignments using universal ring approach, as shown in Figure 18.

2.4 Soil

2.4.1 Introduction

In tunnelling project often very complex geological conditions are met. It is very common that the tunnel construction is conducted through the different geological layers under ground water level, and that those layers are also not heterogeneous, but the soil characteristic within one layer also may vary a lot 19. Therefore, the underground projects have vast uncertainty. Geology strongly affects almost every major decision that must be made in the planning, design, and construction of a tunnel determining the cost, and the behaviour of the completed structure.

The design of major construction works requires an understanding of the local ground conditions and, the better the ground conditions are known, there is more control in execution of the project and less uncertainty when assessing risk, safety, design and financial costs.

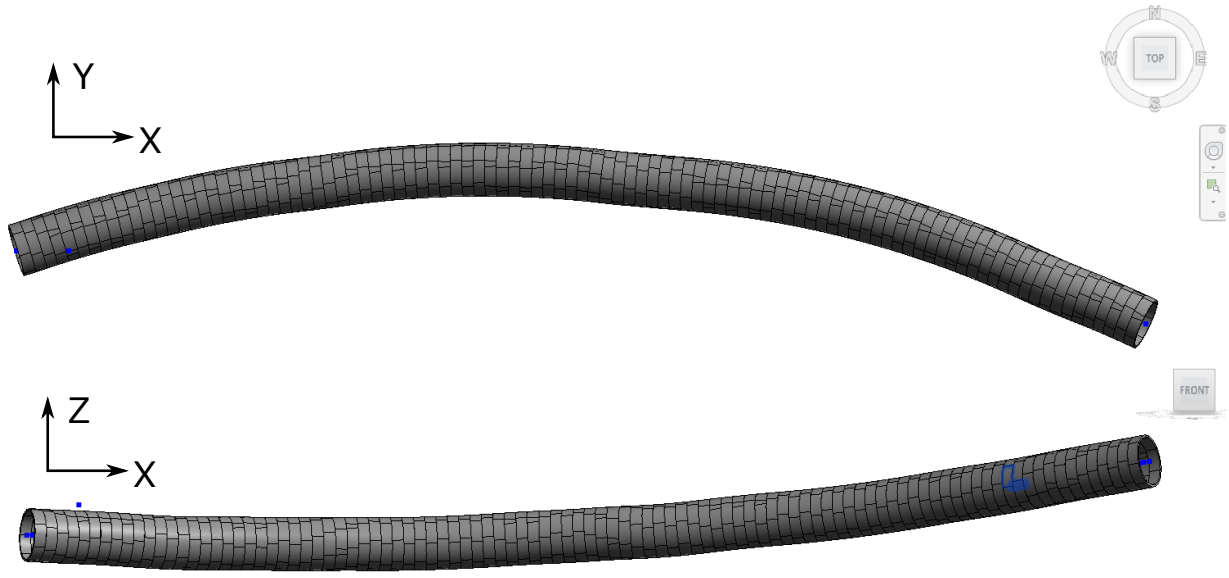


Figure 18: Three-dimensional segmented tunnel structure corresponding to alignment shown in Figure 14

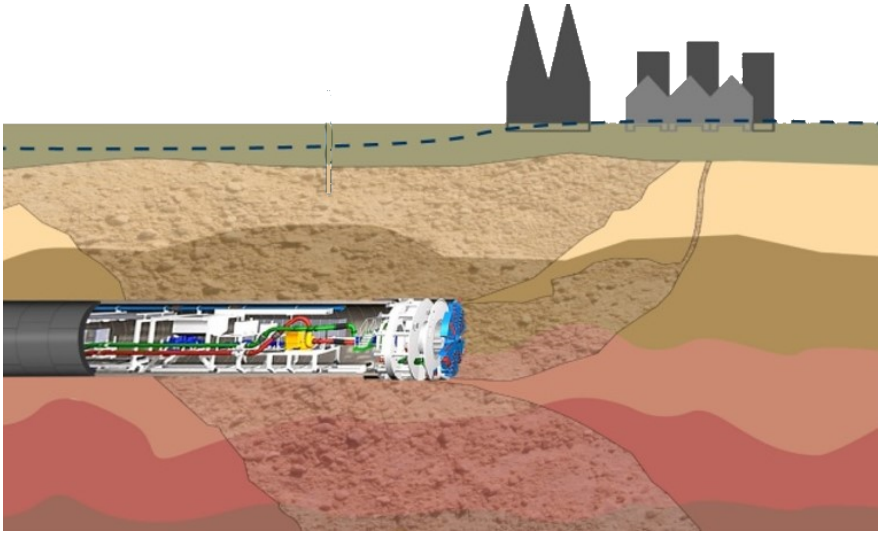


Figure 19: Excavation of tunnel through various soil layers

Ground model is developed based on a desk study of the local geological sequences and proceeds by means of ground investigation using boreholes and trial pits, commonly complemented by in situ testing and geophysical surveys, as appropriate to local needs and circumstances. Nowadays, all processed data is stored in 3D GIS models. 3D GIS applications and 3D visualization technology has become one of the most effective techniques to represent 3D modelling of the geological bodies (including: formation lithology, geological structure, etc.) and reconstruct the 3D geotransects in the virtual circuit tunnel projects Aldiss et al. (2012). Recently Geo Builidng Information Models (GeoBIM) have been developed to enable not only the management of subsurface construction along but also all

geo-related (subsurface) data, such as geological, hydro-geological and geotechnical objects and properties Zobl and Marschallinger (2008).

In terms of numerical simulations, the probably the most important feature is the proper modelling of the surrounding ground. Not only because the ground is the domain where the complete process takes place but also because it is the part of the model where most of the effects of tunnelling happen (excavation, application of support measures, consolidation). Moreover, the behaviour of the ground has predominating influence on the surface settlements and the structural stability of the complete system.

The `SatBim Modeller` is able to handle volumetric data in the ACIS format; a file format that is well-established in 3D modelling and can be imported to and exported from other programs. Therefore, in this project, the soil on highest LoD uses external Loadable models in ACIS format.

2.4.2 Families

For the TIM model developed for this project, we allowed for representations of the soil on different level of accuracy (details). CAD models of relevant geological features containing soil or rock layers, their boundaries, and their geotechnical properties are standard format for tunnel ground models. Since Rivet allows to import already generated CAD formats, the highest level of detail for soil representation is (LoD3) actual CAD geometry developed by geo-science experts (see Fig. 20). This geological model consisting of geotechnical layers represented with a single or multiple volume and material description corresponding to layer is used to construct a geometrical representation of the surrounding ground and later on will be used for the tunnelling simulation.

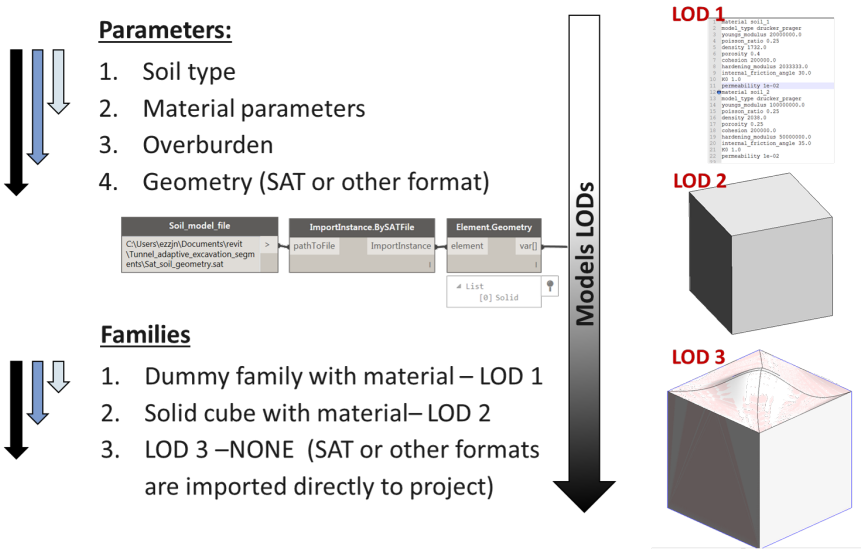


Figure 20: Families and their parameter dependencies for Soil model on different level of detail

If detailed 3D geological model is not available, but only an estimation about the geometry and composition of the soil, for a second level of detail (LoD2) a rectangular bounding box,

representing substitute soil volume is defined. The bounding box is defined as a custom family described as a rectangular volume with material. This bounding box is used to delimit the simulation model and is chosen large enough that the model boundaries do not affect the simulation results while being small enough not to unnecessarily expand the complexity of the model. By experience, the distance of the model boundaries from the tunnel axis should be chosen a five to eight tunnel diameters in radial direction and about ten to fifteen diameters in axial direction. Therefore, the size of bounding box depends on Tunnel alignment, overburden and excavation radius.

Finally if there is no information of the geometry of the soil layers, and only rough information about the material trough which the tunnel will be constructed, in the lowest LoD the soil is represented with Dummy family which has no geometry but only parametrically assigned elastic material parameter values.

2.4.3 Excavation

For given soil component (LoD2 and LoD3) along the projected tunnel alignment a soil volumes that are to be removed (excavated have to be defined). Again, similar as for Grouting (see Section 2.3.4) this geometry is not "independent component" since it depends on the soil component and therefore will not be represented with custom families, but instead with geometrical entities representing excavated soil volumes.Those volume entities defined based on soil parameter - excavation radius (r_{exc}), and calculated tunnel alignment. Since the the rings of certain length L_r are installed, the TBM moves forward in steps of the ring length and subsequently removes the soil. Therefore, the created excavation volumes are extruded from the circle with radius r_{exc} along the alignment path for advancement step L_r.

2.4.4 Dynamo lining and grouting generator

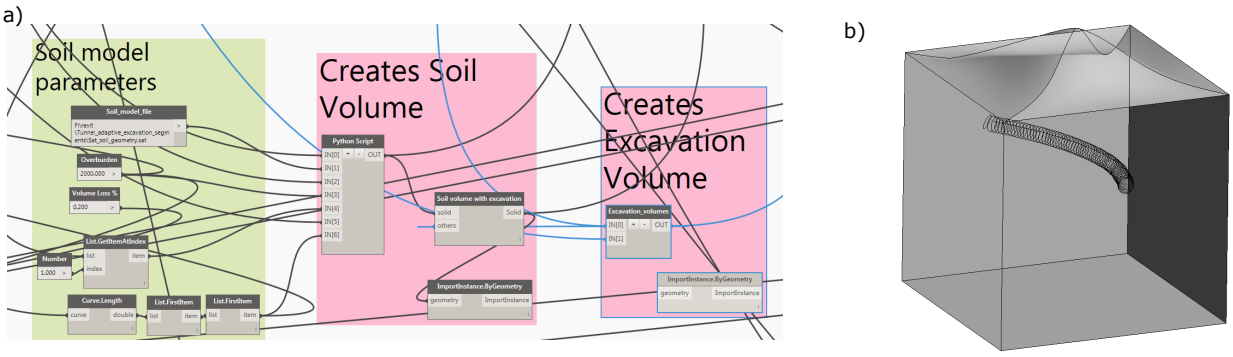


Figure 21: Generation of Soil and Excavation volumes: a) Dynamo nodes; b) Soil with excavation

In Fig. 21 (a) a Dynamo node for generation of the soil is shown. In this node first of all soil volume is either imported for LoD3 or generated for LoD2. This is done to custom Python script. With another python script, Excavation volumes are generated based on new adjusted tunnel alignment and excavation volumes. From the imported or generated

Soil volumes(s) the excavation volumes are extracted leaving the soil volume with a hole. Finally, those two models are merged creating final soil and excavation structure as shown in Fig. 21 (b). The user can dynamically change any model parameter (alignment, excavation radius, soil) and the information model of the soil and excavation is automatically generated and adjusted to new requirements.

2.5 Building

2.5.1 Introduction

Tunnelling-induced surface and subsurface displacements in urban environments may cause significant damage to existing structures and foundations, in particular in the case of a tunnel advance with low overburden. The response of existing buildings to tunnelling-induced ground movement in urban tunnelling is a fully coupled 3D problem of tunnel-soil-structure interaction. While the unavoidable volume loss associated with tunnel construction causes deformations of buildings, the buildings, by virtue of their structural stiffness, on the other hand, are influencing the tunnelling-induced soil deformations. Understanding of this mutual interaction may be highly relevant for the economic efficiency of a project. Precaution measures taken in order to prevent potential damage due to the projected advancement of tunnels often turn out to be inadequate, leading to an unnecessary increase of costs. Evidently, these interactions are strongly dependent on the position of the buildings with respect to the tunnel axis and on the shape-modes of tunnel-induced settlement Potts and Addenbrooke (1997).

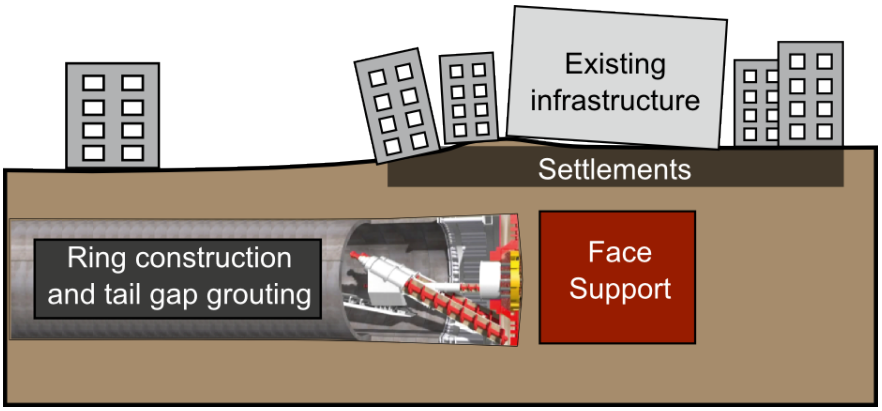


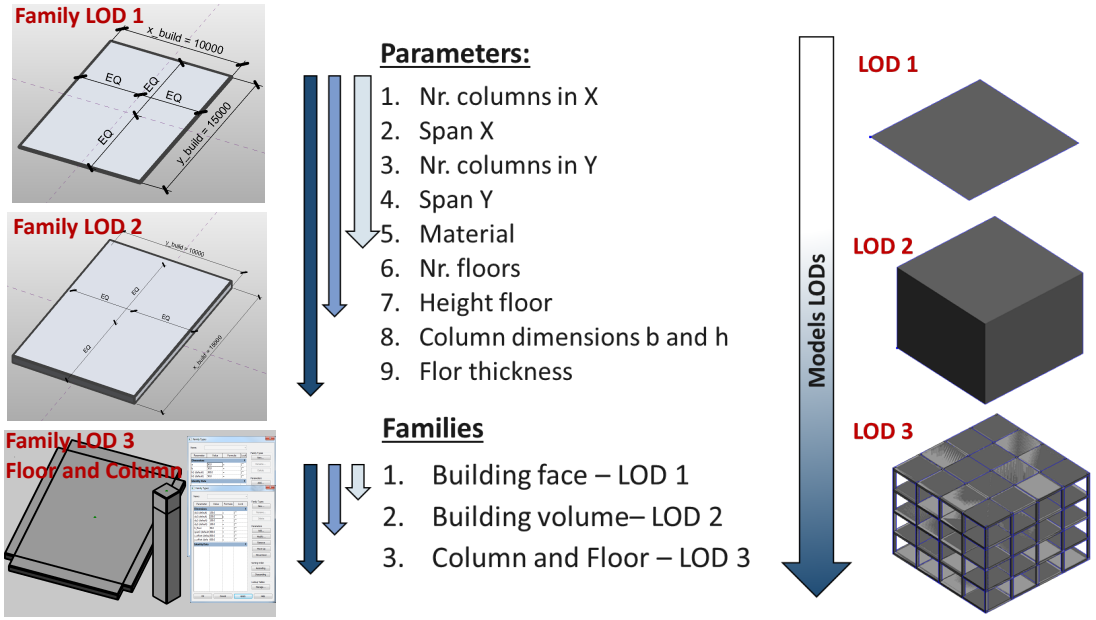
Figure 22: Interaction of existing infrastructure with tunnelling process

In some cases, for design purposes, the interest is not in the detailed distribution of stresses within the structural members, but in the distribution of aver-aged quantities such as bending moments, axial and shear forces, while clearly keeping the accuracy of the global structural response. If the focus is on tunnelling-induced settlements due to interaction with existing structures rather than the influence of settlements on the damage of the structures, the reduced models for structures can be used. Otherwise, if the target is to access the effect of tunnelling on structural behaviour, the detailed models of structures are required. This

shows the necessity for multi-level approach when modelling the soil-structure interaction during tunnelling process.

2.5.2 Families

Buildings can be represented by the equivalent load of the dead weight, using substitute models or using detailed structural models for buildings. Although Revit® offers a number of Loadable families for all structural components for modelling of buildings (columns, floors, faades, etc.) for this project new custom families are developed to match the requirements of the multi-level modelling approach (see Fig. 23a).



a c

Figure 23: Building multi-level information model: (a) Families for different LoDs; (b) Parameter dependences

For the lowest LoD, only a surface representation of building is adopted, representing the building footprint area, and corresponding custom family is defined based on parameters describing this surface area. This model corresponds to numerical model where the building is represented by the equivalent surface dead load of the building. In LoD2 building is represented with a volume occupied by the building, and corresponding custom family now includes parameters defining number of floors, floor floor height. etc. Finally, for LoD3 of the building custom loadable families for floors and columns are are developed including complete set of parameters for building discretion (see Fig. 24b).

2.5.3 Dynamo lining and grouting generator

In Fig. 24 a Dynamo node for generation of the building is shown. Aside of loading custom families for different LoDs, a nodes for defining material parameters are added. Based on

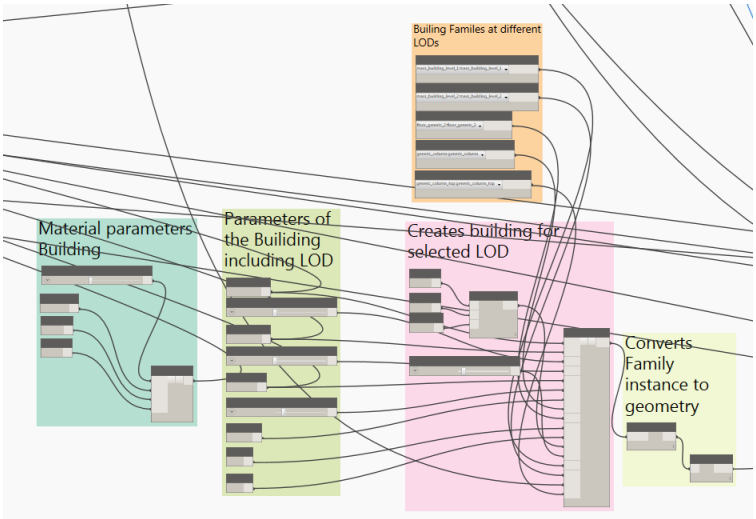


Figure 24: Dynamo node for generation of building in different LoDs

building geometry parameters, material parameters and insert point of building and rotation, using custom Python node a building is generated and can be changed dynamically.

2.6 TBM

2.6.1 Introduction

A tunnel boring machine (TBM), also known as a "mole", is a machine used to excavate tunnels with a circular cross section through a variety of soil and rock strata. They can bore through anything from hard rock to sand. Tunnel diameters can range from a metre (done with micro-TBMs) to 19.25 metres to date. In soft ground, there are three main types of TBMs: Earth Pressure Balance Machines (EPB), Slurry Shield (SS) and open-face type. Both types of closed machines operate like Single Shield TBMs, using thrust cylinders to advance forward by pushing off against concrete segments.

In shield tunnelling process the TBM is pushed forward by elongation of the hydraulic jacks, excavating the soil by a rotating cutting wheel. The excavation proceeds until the length of one lining ring has been reached. Subsequently, the machine stops and the next lining ring is erected using precast concrete segments. Simultaneously, grouting mortar is injected into the annular gap. After the ring installation has finished, the excavation starts again. While the shield skin secures the material around the excavation area, the heading face, where the actual excavation takes place, is not sealed. In order to prevent material from flowing into the tunnel and to decrease settlements due to movement of material, supporting measures are taken.

The geometry, components and main parameters defining the EPB shield are shown in Fig. 25

The TBM does not fit perfectly into the volume excavated by the cutting wheel, but a gap exists between the undeformed soil and the shield skin to allow for a proper control of the

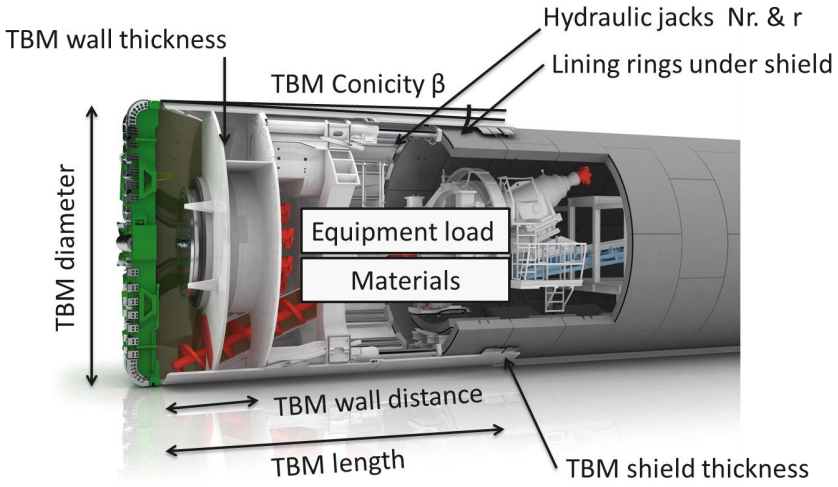


Figure 25: Tunnel Boring Machine and main it main characteristics

TBM movements, to reduce the jack forces and to minimise the wear of the shield skin. This gap, which is called the steering gap, is caused by overcutting at the shield front end and by taper of the TBM diameter from its front to its tail typically around 0:4 % leading to an increasing gap width towards the tail of the TBM . Thus, the soil surrounding the TBM can deform until it gets, at least partly, into frictional contact with the TBMs steel skin. In addition to the relaxation of the soil into the steering gap its deformations are affected by deformation of the shield tail when it is subjected to the earth pressure, and by the weight of the TBM.

2.6.2 Families

In terms of numerical modelling there are different approaches of representing the shield machine. Since the main function of the shield is preventing that the material around the excavation area converges towards excavation, some authors represented TBM exactingly as boundary condition limiting deformation of the soil Founta et al. (2013). However, TBM is also an deformable body and the taper of the TBM and the frictional contact of the shield skin with the surroundings play an important role for the re-distribution of stresses and pore pressures in the soil. Therefore TBM can be represented with a 3D model interaction with surrounding soil with frictional interface. Additional advanced modelling feature is to account for hydraulic jacks that are attached to the TBM and use the previously erected lining segments as thrust bearings. In order to prevent divergence of the machine from the alignment, the thrust jacks are also used to steer the shield by setting different jack pressures.

Similar as for buildings using Dynamo node TBM for specific LoD is generated.

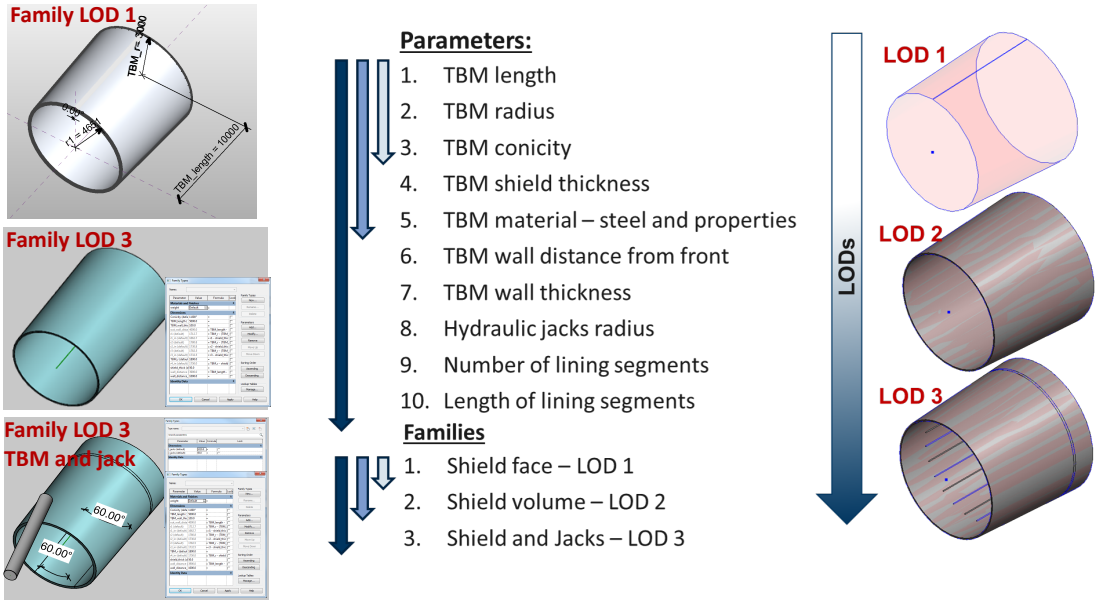


Figure 26: TBM multi-level information model: (a) Families for different LoDs; (b) Parameter dependences

2.7 Multi-level information model for tunnelling

Combining all individual components (lining with it alignment and grouting, soil with excavation, TBM and buildings) the complete tunnel model is generated as shown in Fig. 27. As already mentioned for generation of individual components, local parameters, characteristic for each individual component, are defined, however there are number of parameters which are common for multiple components. Those parameters are defined as Global parameters (see Fig. 27a). Among the global parameters, such ring length L_r, excavation radius r_{exc}, number of steps/slices, overburden, etc., a parameters defining the location of the model *model_path* and *model_name* are assigned. On this location, a directory *model_name* is defined, and all model output (geometry and global and local parameters) are stored. Based on this data the simulation model is generated afterwords.

The complete DYNAMO model for generation of multi-level information model for tunnelling is shown in Fig. 28. Here it can be seen how the definition of global parameters and local parameters is defined and how all individual components stay in relation with each other.

In order to generate the Simulation model for tunnelling with identical geometry, material and semantic parameters, the generated TIM is exported by Dynamo nodes shown in Fig. 29. Geometry of all individual components is exported in ACIS format. Each volume of each individual component is exported separately, in order to enable flexible manipulation with those volumes when assigning custom layer entities needed for numerical simulation (e.g. each lining ring or segment is exported as separate file *lining_volume_n.sat*). Global parameters of the model and the local parameters of individual components are exported as text files, *main_model.tex* and *component.tex* respectively. Moreover, in separate text files

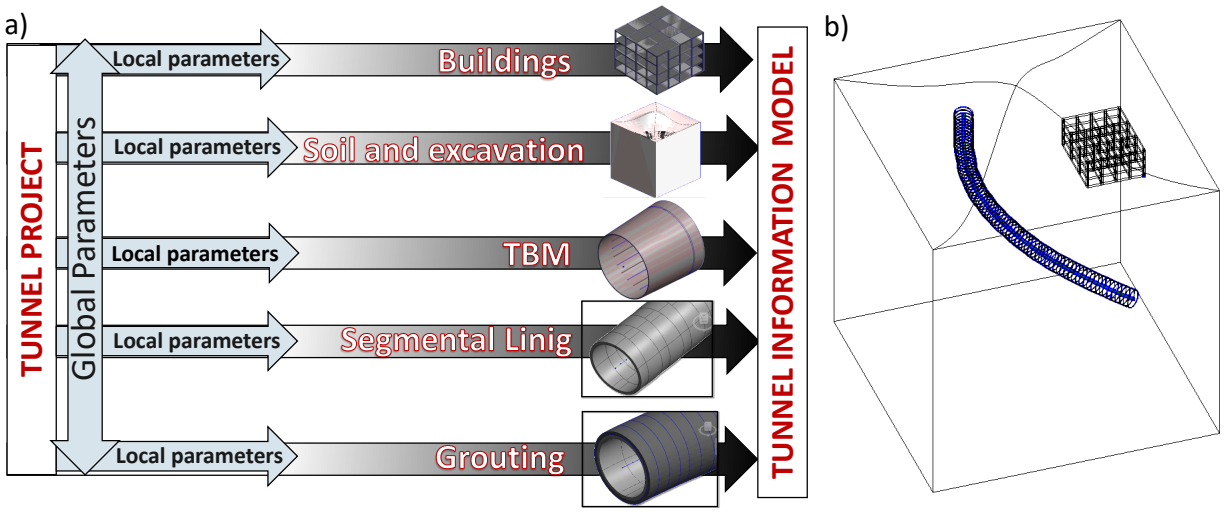


Figure 27: Tunnel information model: a) Combining sub-models and their dependency on local and global parameters; b) Complete tunnel Information model.

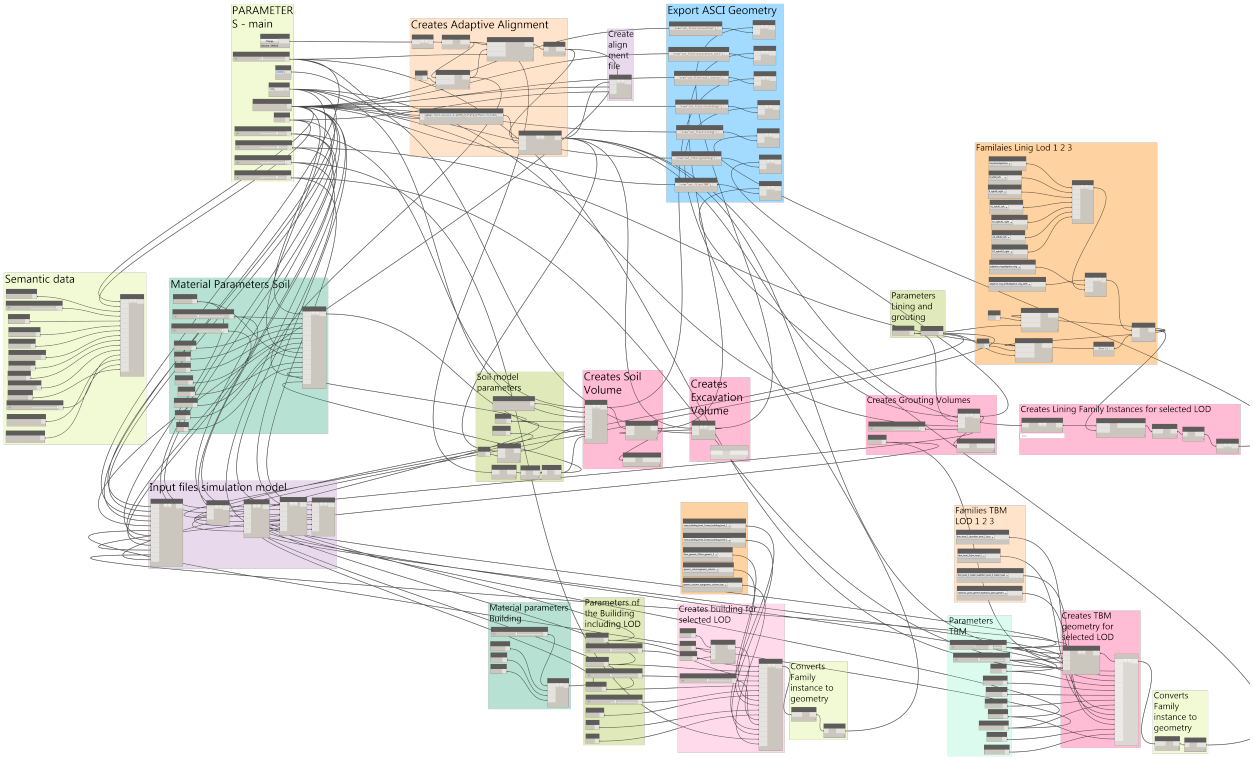


Figure 28: Structure of the complete Multi-level tunnel information model

are exported, material properties, semantic data and tunnel alignment.

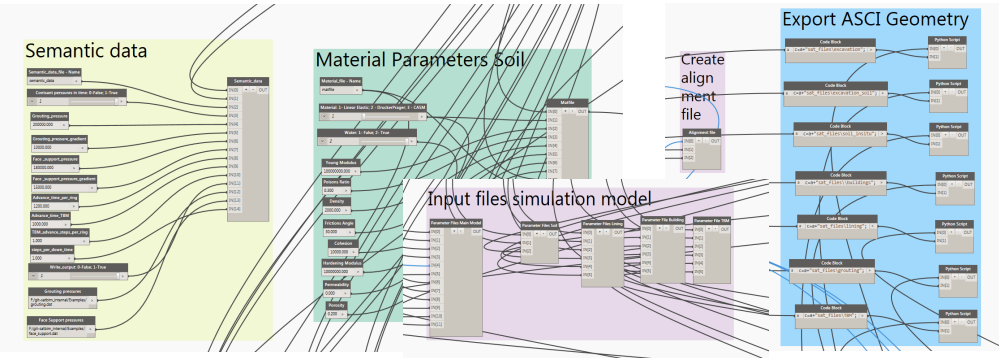


Figure 29: Dynamo node for exporting the model parameters and model geometry

3 Simulation models

In previous section the the development of the multi-level information model for tunnelling has been presented while in this section the development of the multi-level numerical simulation model for tunnel-soil-structure interactions. As a reminder, the tools and structure of the concept is shown in Fig.30.

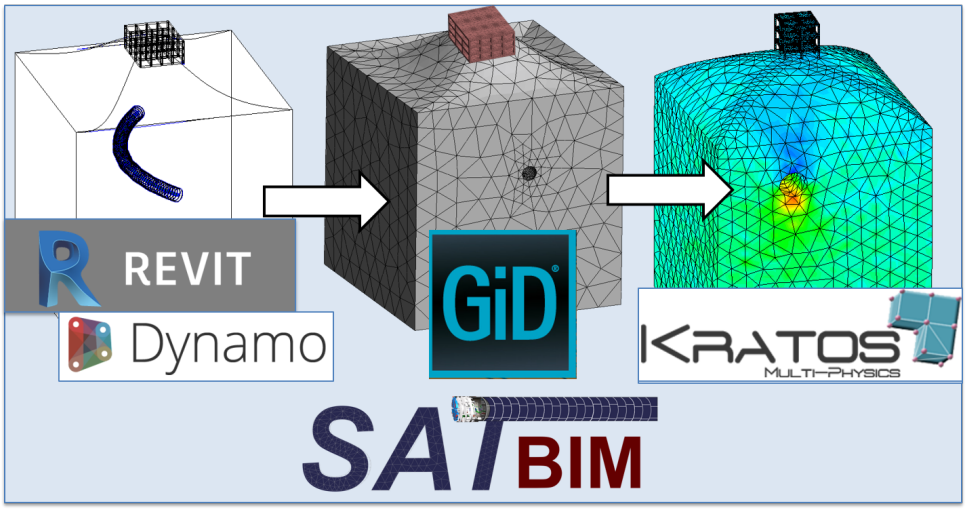


Figure 30: The flow of SATBIM multi-level model generator

Why do we need automation in model generation? In practice, numerical simulations are usually carried out to gain information on a specific project. If the investigation of different design variants for a project is required, very probably the generation of different numerical models is needed which leads to high user workload. In this case, the model generation becomes the bottleneck of efficiency. Then, the demand for an efficient simulation software is to minimise the effort needed for the model generation as well as the evaluation of the simulation results generation as well as the evaluation of the simulation results. `SatBim Modeller` fulfils those requirements.

Dependencies between component on different LoDs In the Section 2.2.1 describing generation of TIM, it has already been mentioned that the main tunnel model comments are: soil, lining, TBM and buildings. Those main components are models considering different LoDs. On the other hand, grouting and excavation are defied as dependent structural components, since their existence in the model depends on the LoDs of the lining and soil respectively, and they are modelled on only one LoD. In table given in Fig. 31 is shown how different main components on different LoDs, as well as dependent components can be combined together. With $\times$ are denoted the impossible combinations. An example is combination of soil LoD1 (soil is represented with elastic springs) and Lining LoD1 (Volume loss method). In this case, both, soil and lining geometry does not exist, but they are described

only with parameters. An alternative for this combination would be implementation of analytical model for determination of tunnelling induced surface settlements and structural forces in lining, based on model parameters.

											Dependent components	
		soil	Lining			TBM			building			grouting	excavation
soil	Lod1		✗	✓	✓	✗	✗	✗	✗	✓	✓	✗	✗
	Lod2		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
	Lod3		✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
Lining	Lod1					✗	✗	✗	✓	✓	✓	✗	✓
	Lod2					✓	✓	✓	✓	✓	✓	✓	✓
	Lod3					✓	✓	✓	✓	✓	✓	✓	✓
TBM	Lod1								✓	✓	✓	✓	✓
	Lod2								✓	✓	✓	✓	✓
	Lod3								✓	✓	✓	✓	✓
building													

Figure 31: Numerical models for tunnelling combing different level of details for subcomponents

3.1 SatBim Modeller

One aspect of software efficiency is the effort a user has to invest in learning and using the software. Therefore, particular attention has been paid to automation of input generation for the tunnelling simulation software using the Tunnel information model. Hence, the simulation software shall be able to automatically process model input from another software, leaving as little workload as possible to the user.

Modular structure SatBim Modeller software is organised in a highly modular way in order to offer a high flexibility not only for further extensions, but also to adapt to any changes in the simulation software. This not only facilitates debugging but also allows to still use a large share of the complete modeller even for situations where the fully automatic model generation is intractable. Therefore, the modeller can be seen as a collection of tools for assistance in model setup for a large variety of situations rather than a fully automatic software for a constricted scope of situations.

3.1.1 Program structure

SatBim Modeller model generator is realised as a library of Python scripts which are used to generate both particular parts and the assembled global simulation model. The basic task of the model generator is the creation of a number of batch files that are used to control the

GiD preprocessor. These batch files contain every instruction that is necessary to read and generate the geometry, apply the boundary conditions and generate the mesh.

To provide a flexibility in terms of model generation, and further changes and extensions, `SatBim Modeller` is implemented with highly modular structure (see Fig. 32). The top Class is *SatBimModeller*. It contains Python modules: *ReadParams*,*IvokeGiD*, *Run*, *CompileSystem* and *CreateSimulationScript*. All individual components (soil, lining, grouting, building) are defined as classes, inherited from the *ModelComponent* class. The *ModelComonent*, and naturally their inherited classes, includes modules: *ReadParams*, *PrepareModel* and *AddToSimulatioScript*.

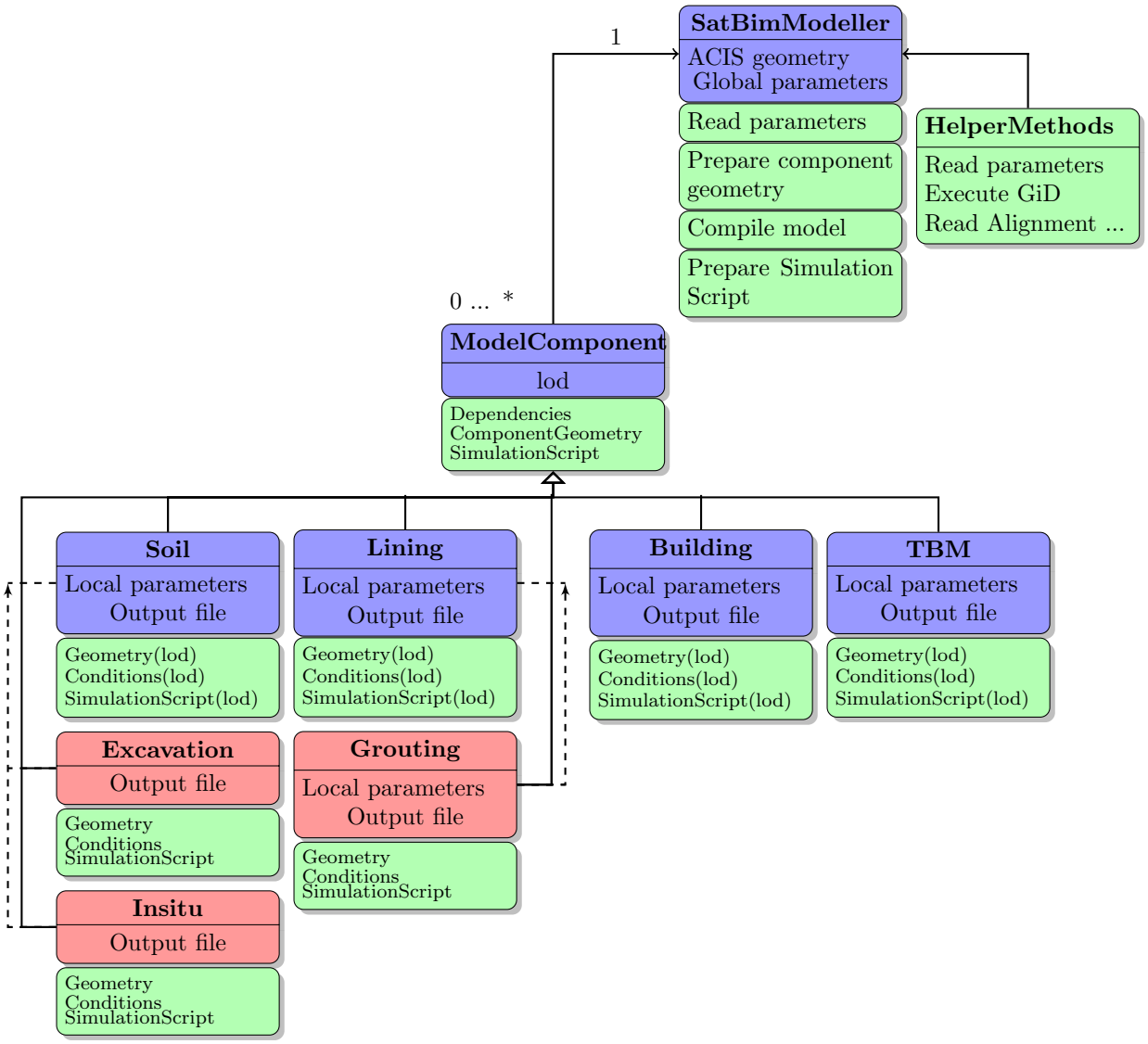


Figure 32: ***SatBim Modeller*** *structure*

When initialising and executing the following tasks are preformed:

- Crete model directory,

- read global parameters,
- establish dependencies between model components,
- read local parameters of each component,
- prepare geometry of each component,
- compile system merging all geometries, assigning materials and boundary conditions, generate FE mesh and prepare model output,
- create Simulation script and,
- run the Model.

3.1.2 Input parameters

In Section 2.2.2 it was explained how the geometry and parameters defining the tunnel project are exported for TIM in a form adjusted to be used as an input for a simulation model.

Briefly summarizing the structure of TIM output and `SatBim Modeller` input, all information about geometry is stored in directory $model\_path/ACIS\_files/component\_name$, while while the local and global parameters defining the model are exported to $model\_path/parameter\_files/m$ and $model\_path/parameter\_files/component\_name.dat$ respectively. It is important to mention that LoD is a stored as a local parameter for each component and than dependences between different instances are defined in a top class $SatBimModeller$. Material parameters are stored in $model\_path/parameter\_files/matfile.dat$ while semantic data is stored in $model\_path/parameter\_files/semantic.dat$. Finally, the coordinates of the calculated tunnel alignment are stored in $model\_path/parameter\_files/alignment.dat$.

$HelperMethods$: $ReadParamFile$, $ReadMaterilaFile$ and $ReadAlignmentFile$ are used to read parameters and define them as local variables in the $SatBimModellers$ for generation of simulation model.

3.1.3 Generation of FE model

For each of the sub-models a dedicated Python class is available that contains a modules $PrepareModel_LoD$ to generates specific batch files for GiD. For the structural components (soil, lining, TBM and building), a handlers for LoDs are defined to decide which $PrepareModel_LoD$ module to call depending on the local variable lod. On the other hand, the dependent components (excavation, insitu and grouting) are defined with only one LoD, and are called based on defined dependencies. When executing $SatBimModelrar$ module Run, after reading global parameters and establishing dependencies between components, the individual models will be prepared and then compiled to complete simulation model as shown in 2.

Listing 2: Prepare individual components of the tunnel model and compile system

```
# Prepare models for all model components
print "Run: Preparing model components ..."
for component in self.model_components:
        # Prepare model
        output_path = self.working_dir + self.params['model_name'] + "_"
        + component.component_name + ".bch"

        # Open file
        bch_handle = open(output_path, 'w')

        # Prepare component
        component.PrepareModel(bch_handle)

        # Close file
        bch_handle.close()

        # invoke GID
        self.InvokeGID(output_path)

        # Compile all model Components into simulation model
        self.CompileSystem()
```

Basically, for each individual component a module $PrepareModel_Lod$ for selected LoD, a batch file will be generated to execute following tasks in GiD:

- read local parameters and assigned LoD,
- insert geometry and assign layers to entitles,
- prepare material file,
- prepare boundary condition file.

Afterwords in $CompileSystem$, all individual components are imported into $model\_name\_system$ file and all material parameters and boundary conditions are assigned. GiD is invoked from the command line in silent mode and executes these batch files. Finally, FE mesh is generated and executing $CalculateModel$ GiD module ASCII files that contain all necessary data for the simulation, including the mesh information, boundary and initial conditions, material properties, and model parameters are generated in the form that can be read by computation kernel KRATOS.

3.1.4 Generation of simulation scripts

Apart from the generation of input files that can be read by the simulation software, a Python script is generated that is used to control the simulation workflow. This script serves as an interface between the FE model and the simulation software, providing a customised setup of the kernel and preparing the model for the simulation. The Python script is designed as an own module that defines all necessary functions needed for the simulation workflow, taking actual FE model as an object. In this Python script the functions $Solve()$ and $WtiteOutput()$ are called to solve the model, however, also number of different functions to assign boundary conditions (loads, displacements, contact interfaces, deactivation, activation, etc) to each individual component in order to simulate the tunnel excavation process.

Listing 3: Add contribution of each component to simulation script

```
self.CreateSimulationScript()

for component in self.model_components:
        # Write simulation script model
        output_path = self.working_dir + self.params['model_name'] + "_simulation.py"

        # Open file
        bch_handle = open(output_path, 'a')

        # Add component contribution to simulation script
        component.AddToSimScript(bch_handle)

        # Close file
        bch_handle.close()

#coppy soil_properties utility and material file
shutil.copy(self.problem_path+self.params['matfile']+".dat", self.working_dir)
shutil.copy('soil_properties_utility.py', self.working_dir)
```

Depending on the LoD of each component and mutual dependency between continents different simulation models are generated and each of them requires spatial *SimulationScript*. Therefore each component class contains the Python module *AddToSimScript* for each LoD where the "contribution" of the respective LoD of the component is added to the base simulation script called *SimulationTempllate.py* (see Alg. 3). *SimulationTempllate.py* contains just basic initialisation of the simulation software kernel. As a result a Python script *model_name_simulation.py* is generated, and executing this script, the numerical analysis can be performed.

3.1.5 Invoke of simulation software

In order to run *SatBimModeller* for a specific TIM model, a simple shell executable file *run_modeller.sh* (see Alg. 4) is to be executed in a command prompt.

3.1.6 Post-processing and result data base

The simulation results can be stored both in result files and read by the post-processing module of GiD that provides various visualisations the results. In the cases where a user invokes the simulation manually, this is the customary way of post-processing. Results such as displacements, stresses, or internal variables can be easily illustrated. Also, the construction sequence can be animated. However, visualisation of "raw" numerical analysis output (e.g. simulation results stored in each node or Gauss Point of FE model) might not be

Listing 4: Execution of model generator

```
modeller_path=('D:/git-satbim_internal/SatBimModeller/')
problems_path=('D:/git-satbim_internal/Examples/Model_19/')
gid_path=('C:/Program_Files/GiD/GiD_12.0.10/')
JOB_ID=1
echo "running_modeller..."
python "${modeller_path}"/SatBimModeller.py "$modeller_path" "$problems_path" "$gid_path" $JOB_ID$
```

efficient in order to draw conclusions about phenomena occurring during tunnel constriction. Therefore, these results should be be processed by arbitrary evaluation algorithms. As an example, a search algorithm can be applied that returns risk on damage of structures due to tunnelling in simplified way. To enable this a new visualisation method will be developed within WP of the SATBIM project.

3.2 Numerical Models on different LoDs

3.2.1 Soil Models

In numerical simulations of the mechanized tunnelling process, one of the most important requirements is the proper modelling of the soil behaviour, including complex hydraulic conditions. This includes modelling of the multi-phase composition of the soil and the ability of the pore fluids (water and air) to flow through the voids between the soil grains (Nagel, 2009), or even the transportation process of fine particles through the soil pore network (Schaufler et al., 2013). Besides establishing a proper constitutive framework for the description of the hydraulic behaviour of the soil mixture, a key feature of a good soil model is a realistic description of the material (stress-strain) response of the soil skeleton. A large number of constitutive models have been developed including elastic-plastic behaviour, hardening, softening, small strain stiffness, anisotropy, creep and cyclic behaviour, etc. Still, according to Potts and Zdravkovic (1999), "due to the complexity of real soil behaviour, a single constitutive model that can describe all facets of behaviour, with a reasonable number of input parameters, does not yet exist". In terms of the application of the constitutive models for simulations of a real tunnelling process, the proper calibration of the model with laboratory tests is required, but not always an easy task for very complex soil models containing a large number of parameters, which not all a necessarily have physical meaning. Therefore, it is key to find a balance between the complexity of the soil model and available laboratory testing necessary to properly calibrate the model.

A numerical analysis of the influence of the joints of the segmental lining on the overall behaviour of the tunnel structure is typically performed without consideration of the complete tunnel construction analysis, but rather applying sophisticated models for lining and joints, and observing the behaviour under design loads, while the resistance of the soil is modelled by subgrade reaction springs (Arnau and Molins, 2012; Teachavorasinskun and Chub-Uppakarn, 2010). Moreover, for modelling of buildings, Limiting Tensile Strain Method (LTSM) Burland and Wroth (1975) or Winkelr beam method is currently used in the engineering practice to predict building damage due to excavation induced ground deformations. LTSM prediction methods is based on the calculation of induced tensile strains in a structure due to imposed differential vertical and horizontal ground deformations. However, the soil-structure interaction is neglected in the empirical analytical LTSM, that assumes full transfer of the greenfield ground movements to the building. Therefore, for simplest representation of the soil the subgrade reaction method is chosen in order to have simulation model with minimal number of DOFs (computationally very cheap), however, at least to some extent include soil structure-interaction effect. For more detailed representation of the TBM advance and tunnel construction with excavation of the soil, an advanced FE model for fully and partially saturated soils is employed (Nagel, 2009).

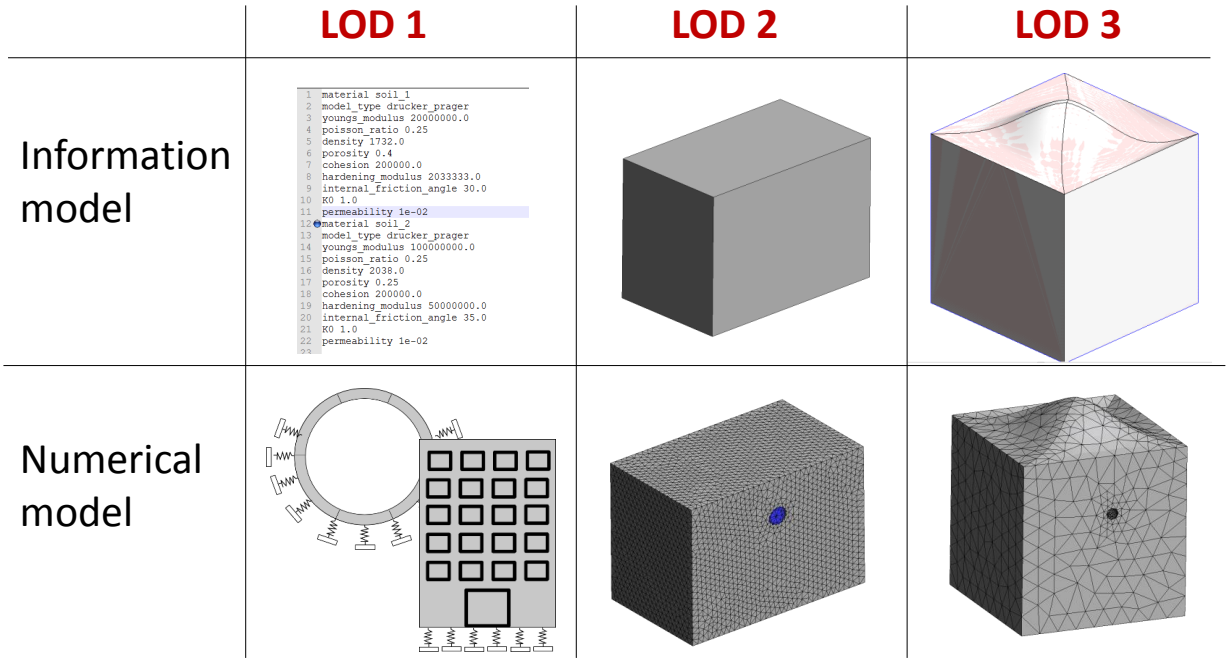


Figure 33: Soil information and numerical models on different level of details

LoD1 For representation of the soil a subgrade reaction model is adopted where the soil is represented by infinitely thin, uncoupled springs neglecting the soil-structure interaction and the weight of the excavated soil (Fig. 33 LoD1). Linear elastic subgrade reaction is given if springs are liner ($p = K\dot{u}$), where p is pressure between building and soil and K is subgrade reaction modulus. Subgrade reaction approach permits development of elegant analytical solutions for determining the deformation of buildings, using WINKLER equation:

$$EI\frac{\partial^4 w}{\partial x^4} = q_0(x) - r(x) \qquad \text{where} \qquad r(x) = B * K_h * w(x) \tag{11}$$

where EI is the beam stiffness, B is the beam width, K_h is the coefficient of the horizontal subgrade reaction, while $w(x)$ and $q_o(x)$ are deflection of the beam and load functions, respectively.

However, the challenge is determining the subgrade reaction coefficient K which cannot be measured. In a simple model proposed in VESIC (1963) this coefficient is given as:

$$K_s = \frac{Es}{B\dot{I}_p(1-\nu^2)} \tag{12}$$

Where the I_p is shape factor of foundation.

When determining the subgrade reaction modulus of the springs for lining model, according to KOLYMBAS (1998), the stiffness of the spring is assumed to depend on the stiffness of the soil E, the Poison's ratio ν and the radius of the tunnel lining r:

$$K_s = \frac{E}{r}\frac{1-\nu}{(1+\nu)(1-2\nu)} \tag{13}$$

LoD2 In this LoD soil is treated as structural FE model, and geometry, determined as a bounding box, is used to delimit the simulation model (see Section 2.4). The soil is modelled as a two-phase fully saturated material, accounting for the soil matrix and the pore water as distinct phases according to the theory of porous media (see Nagel and Meschke (2010) for details).

For fully saturated soft soils a two-field finite element formulation is used. The governing equations of this model are given by the weak formulation of the mass balance equation δW_w for the water flow in the pore space of the soil and the grout, respectively, and the weak form of the equilibrium equation δW_m:

$$\delta W_w = \delta W_{w,int} - \delta W_{w,ext} = 0\,, \qquad \delta W_m = \delta W_{m,int} - \delta W_{m,ext} = 0\,, \tag{14}$$

with

$$\begin{aligned} \delta W_{w,int} &= \int_\Omega \delta p_w \mathbf{I} : \dot{\boldsymbol{\varepsilon}}\, \mathrm{d}\Omega\,, \qquad \delta W_{w,ext} = \int_\Omega \delta \nabla p_w \cdot \mathbf{q}\, \mathrm{d}\Omega - \int_{\Gamma_q} \delta p_w q^*\, \mathrm{d}\Gamma_q \\ \delta W_{m,int} &= \int_\Omega \delta \boldsymbol{\varepsilon} : (\boldsymbol{\sigma}' + \mathbf{I} p_w)\, \mathrm{d}\Omega\,, \quad \delta W_{m,ext} = \int_\Omega \delta \mathbf{u} \cdot \rho \mathbf{g}\, \mathrm{d}\Omega - \int_{\Gamma_\sigma} \delta \mathbf{u} \cdot \mathbf{t}^*\, \mathrm{d}\Gamma_\sigma\,. \end{aligned} \tag{15}$$

$\boldsymbol{\varepsilon}$ denotes the strain tensor, $\mathbf{q}$ the pore water flow and q^* the prescribed flow through the boundary. $\boldsymbol{\sigma}$ is the total stress tensor, ρ the density of the mixture, $\mathbf{g}$ the gravitational acceleration and $\mathbf{t}^*$ the traction vector. p_w and $\mathbf{u}$ denote the pore water pressure and the displacements of the soil and the grouting, respectively.

In order to allow for a user friendly input of the soil layers and their material properties, a routine denoted a *MaterialPropertiesUtility* is implemented. Through this simulation script, the material properties are assigned directly to the finite element mesh, storing the respective values at the element Gauss points inside of the polygon defining the respective soil layer. This allows to assign different soil layers with arbitrary topologies, even if the specific soil layers are not separated in distinguished soil volumes and assigned to logical GiD layers.

LoD3 In the highest LoD the same soil representation in terms of FE formulation is used as for LoD2 (two phase soil model for fully saturated soils), however, geometry is defined using actual CAD geometry developed as shown in Section 2.4. Therefore, for representation of individual layers, distinct volumes are available, and therefore distinct FE meshes are generated. In a prospective, interface conditions can be assigned between distinct soil layers, to model interactions, sliding and redistribution of pore water pressures on the soil layer interfaces.

3.2.2 Soil material models in terms of LoDs

In numerical simulations of the mechanized tunneling process, one of the most important requirements is the proper modeling of the soil behavior, including complex hydraulic conditions. Besides establishing a proper constitutive framework for the description of the hydraulic behavior of the soil mixture, a key feature of a good soil model is a realistic description

of the material (stress-strain) response of the soil skeleton. A large number of constitutive models have been developed including elastic-plastic behavior, hardening, softening, small strain stiffness, anisotropy, creep and cyclic behavior, etc. Still, according to POTTS AND ZDRAVKOVIC (1999), "due to the complexity of real soil behavior, a single constitutive model that can describe all facets of behavior, with a reasonable number of input parameters, does not yet exist". In terms of the application of the constitutive models for simulations of a real tunneling process, the proper calibration of the model with laboratory tests is required, but not always an easy task for very complex soil models containing a large number of parameters, which not all a necessarily have physical meaning. Therefore, it is key to find a balance between the complexity of the soil model and available laboratory testing necessary to properly calibrate the model.

Depending on the type of the soil and available material parameters different material models can be applied. If there is no knowledge about the material behaviour, the slimiest soil model winch can be applied is linear elastic. Since elastic behaviour is unrealistic for soils, different elasto-plastic constitutive models are available in KRATOS: the Mohr Coulomb and the Drucker-Prager model, which are preferably used for sandy soils and the more general Clay and Sand model, characterized by non-associative plasticity and Lode-angle dependent yield surfaces YU (1998) which is well suited for clayey soil. In general in geotechnical engineering it is convenient to work with alternative invariant quantities which are combinations of the principal effective stresses. A convenient choice of these invariants is:

$$\begin{array}{ll} \text{Mean effective stress:} & p' = \frac{1}{3}(\sigma'_1 + \sigma'_2 + \sigma'_3+) \\ \text{Deviatoric stress:} & J_2 = \frac{1}{\sqrt{6}}\sqrt{(\sigma'_1 - \sigma'_2)^2 + (\sigma'_2 - \sigma'_3)^2 + (\sigma'_3 - \sigma'_1)^2} \\ \text{Lode's angle:} & \vartheta = tan^{-1}\left[\frac{1}{\sqrt{3}}\left(2\frac{(\sigma'_2 - \sigma'_3)}{(\sigma'_1 - \sigma'_3)} - 1\right)\right] \end{array} \tag{16}$$

Linear Elastic The theory of linear elasticity has a long history of application to geotechnical problems. Initially, emphasis was placed on the use of linear elasticity for a homogeneous isotropic medium. With the development of numerical solution techniques, the behavior of nonhomogeneous and anisotropic media has also been addressed. This applies to dry or drained conditions. In undrained conditions, this will be in terms of total stress. The basic assumption in elastic models is that the directions of principal incremental stress and incremental strain coincide. The general constitutive matrix relates increments of total stress to increments of strain:

$$\begin{pmatrix} \Delta\sigma_x \\ \Delta\sigma_y \\ \Delta\sigma_z \\ \Delta\tau_{xz} \\ \Delta\tau_{yz} \\ \Delta\tau_{xy} \end{pmatrix} = \begin{pmatrix} C_{1111} & C_{1122} & C_{1133} & C_{1112} & C_{1123} & C_{1113} \\ C_{2211} & C_{2222} & C_{2233} & C_{2212} & C_{2223} & C_{2213} \\ C_{3311} & C_{3322} & C_{3333} & C_{3312} & C_{3323} & C_{3313} \\ C_{1211} & C_{1222} & C_{1233} & C_{1212} & C_{1223} & C_{1213} \\ C_{2311} & C_{2322} & C_{2333} & C_{2312} & C_{2323} & C_{2313} \\ C_{1311} & C_{1322} & C_{1333} & C_{1312} & C_{1323} & C_{1313} \end{pmatrix} \begin{pmatrix} \Delta\varepsilon_x \\ \Delta\varepsilon_y \\ \Delta\varepsilon_z \\ \Delta\gamma_{xz} \\ \Delta\gamma_{yz} \\ \Delta\gamma_{xy} \end{pmatrix} \tag{17}$$

with the total stress constitutive matrix [**C**]. In linear isotropic elasticity, two basic constants - the effective Young's modulus E and the Poisson ratio ν are required.

$$C = \frac{E}{(1+\nu)(1-2\nu)} \begin{pmatrix} 1-\nu & \nu & \nu & 0 & 0 & 0 \\ & 1-\nu & \nu & 0 & 0 & 0 \\ & & 1-\nu & 0 & 0 & 0 \\ & & & \frac{1-2\nu}{2} & 0 & 0 \\ & & & & \frac{1-2\nu}{2} & 0 \\ \text{sym} & & & & & \frac{1-2\nu}{2} \end{pmatrix} \tag{18}$$

Alternatively, the effective bulk and shear moduli, G and K' may be used to define constitutive matrix [**C**] such as:

$$C = \begin{pmatrix} K+\frac{3}{4}G & K-\frac{2}{3}G & K-\frac{2}{3}G & 0 & 0 & 0 \\ & K+\frac{3}{4}G & K-\frac{2}{3}G & 0 & 0 & 0 \\ & & K+\frac{3}{4}G & 0 & 0 & 0 \\ & & & G & 0 & 0 \\ & & & & G & 0 \\ \text{sym} & & & & & G \end{pmatrix} \tag{19}$$

$$G = \frac{E}{2(1+\nu)} \qquad K = \frac{E}{3(1-2\nu)} \tag{20}$$

Mohr Coulomb and Ducker-Prager Failure Criterion, in terms of effective stresses (denoted by a prime) is written as follows

$$\tau_f = c' + \sigma'_n \tan(\phi') \tag{21}$$

where the c' is the effective cohesion, and τ_f and σ'_n are the shear stress and normal effective stress on the failure plane at failure, respectivley. This can be seen in figure 34. It should also be noted, that in this formulation, compressive stresses are assumed to be positive.

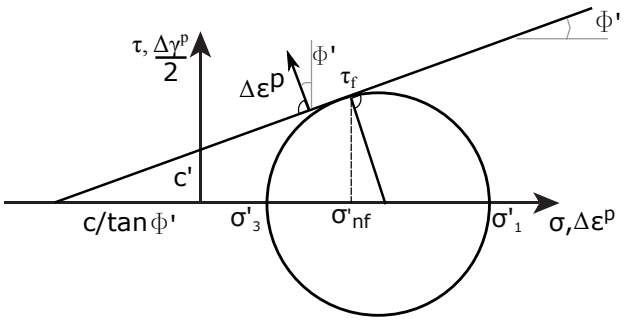


Figure 34: Mohr Coulomb failure envelope plotted with a mohr stress circle. NOTE: Compression is assumed to be positive.

If this failure criteria is re-written in terms of principal stresses is becomes

$$\sigma'_1 - \sigma'_3 = 2c' \cos(\phi') + (\sigma'_1 + \sigma'_3) \sin(\phi') \tag{22}$$

this can be expressed as a yield function as:

$$F(\boldsymbol{\sigma}, \zeta) = \sigma_1' - \sigma_3' + (\sigma_1' + \sigma_3') \sin(\phi) - 2c' \cos(\phi) \leq 0. \tag{23}$$

and, consequently, this yield function can be expressed in terms of stress invariants as

$$F(\boldsymbol{\sigma}) = J_2 - \left(\frac{c'}{\tan(\phi')} + p' \right) g(\theta) \tag{24}$$

with

$$g(\theta) = \frac{\sin(\phi')}{\cos(\theta) + \frac{\sin(\theta)\sin(\phi')}{\sqrt{3}}} \,. \tag{25}$$

In principal stress space, the yield envelope plots as a six sided cone oriented with it's central axis along the hydrostatic axis, as seen in figure 35a .

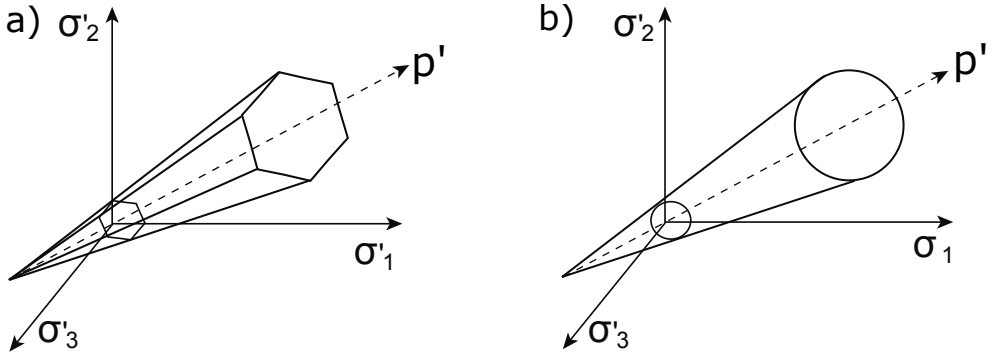


Figure 35: Mohr Coulomb and Drucker-Prager *yield surfaces. a)* Mohr-Coulomb *Modell, b)* Drucker-Prager *Model (pressure is here positive)*

Major flaw of the Mohr-Coulomb Yield Criterion is that it is effectively a multisurface model when plotted in principal stress space, as this produces numerical difficulties in the calculation of the derivative (cite). Thus the Mohr-Coulomb Yield surface can be generalized to a perfectly conical Yield Function with a circular cross section. If this is done, the resulting Yield criterion is referred to as the Drucker-Prager Yield function. The Drucker-Prager Yield Criterion, in terms of stress invariants, is:

$$F(\boldsymbol{\sigma}) = J_2 - \left(\frac{c'}{\tan(\phi')} + p' \right) M_{JP} \tag{26}$$

evaluated at a specific value of the Lode's angle. The Yield surface can be seen in figure 35b.

Clay and Sand Model – CASM The CAS-model Yu (1998) is a generalisation of the well known Cam-Clay model Roscoe and Burland (1968). It is characterised by the yield surface:

$$F = \left(\frac{\sqrt{3} J_2}{M_\theta p'} \right)^n + \frac{1}{\ln r} \ln \frac{p'}{p_0'} = 0 \,, \tag{27}$$

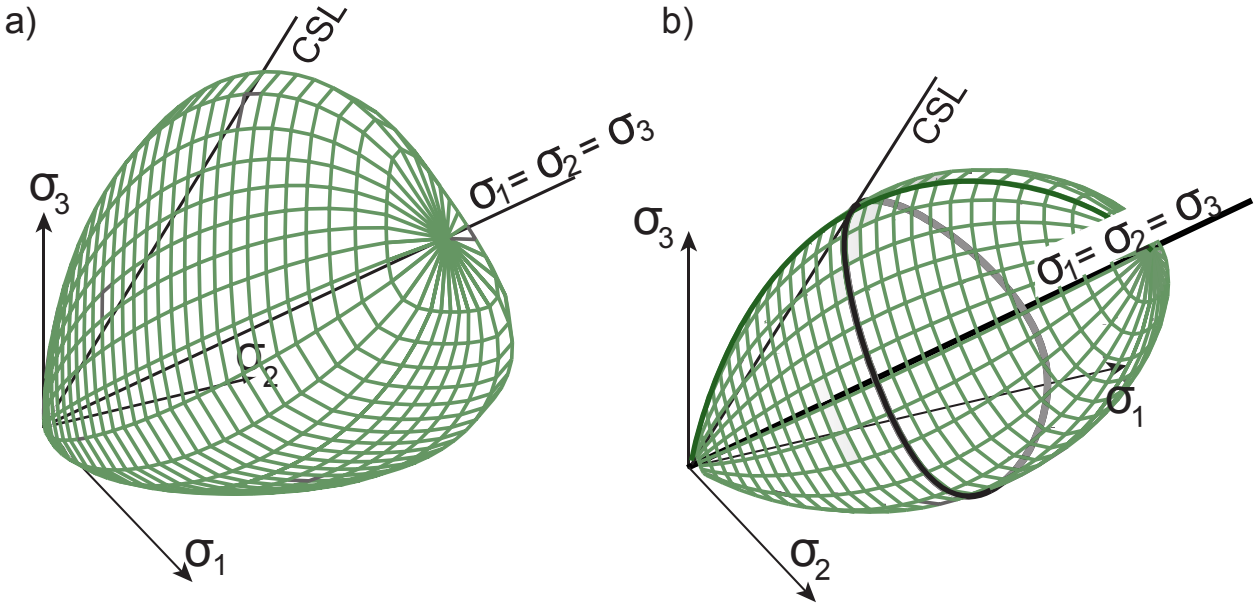


Figure 36: Yield surface of the CAS-model in principal stress state and in the p′-q plane Yu *(1998)*

In CAS-model non-associated flow rule is used for the plastic strain director similar to the one obtained from Rowe's dilatancy rate. Secondly, the dependence of the yield surface and the plastic potential on the Lode angle ϑ is considered by the following formulation of the slope M_ϑ of the Critical State Line (CSL) on ϑ:

$$M_\vartheta = M_{\vartheta=-30^\circ} \left(\frac{2\alpha^4}{1 + \alpha^4 + (1 - \alpha^4) \sin 3\vartheta} \right) , \tag{28}$$

with

$$\alpha = \frac{3 - \sin \Phi_{cs}}{3 + \sin \Phi_{cs}} . \tag{29}$$

The formulation of M_ϑ allows to recover the Mohr-Coulomb hexagon as a special case Sheng et al. (2000). Thirdly, two additional parameters, the so-called *spacing ratio* r and the *shape parameter* n, are used to control the shape of the yield surface to improve the description of sands and highly overconsolidated clays. These modifications, allow to control the pre-failure shear strength on the supercritical and subcritical side of the CSL, and the possibility to derive the model parameters using the state parameter according to Been and Jefferies (1985) instead of the OCR make it applicable to both clays and sands.

3.2.3 Lining Models

A numerical analysis of the influence of the joints of the segmental lining on the overall behaviour of the tunnel structure is typically performed without consideration of the complete tunnel construction analysis, but rather applying sophisticated models for lining and joints, and observing the behavior under design loads, while the resistance of the soil is modelled

by subgrade reaction springs Arnau and Molins (2012). On the other hand, most sophisticated 3D simulation models for mechanized tunnelling do not consider the segment-wise installation of tunnel lining and joints between the segments. Instead, lining is modelled using linear-elastic solid or shell elements, where the complete lining rings are installed stepwise Kasper and Meschke (2004); Nagel et al. (2010); Lambrughi et al. (2012). Recently, a 3D numerical model for the shield tunnelling process was developed using the FLAC3D software, where the influence of the joint pattern of the lining for both segment joints and ring joints is taken into consideration Do et al. (2014a). This study has shown a significant effect of the position and stiffness of the joints on the bending moment and normal forces in the lining, while the effect of joint pattern on the surface settlement is insignificant. In the SATBIM project, a different alternatives for modelling of tunnel lining are enabled following the multi-level approach.

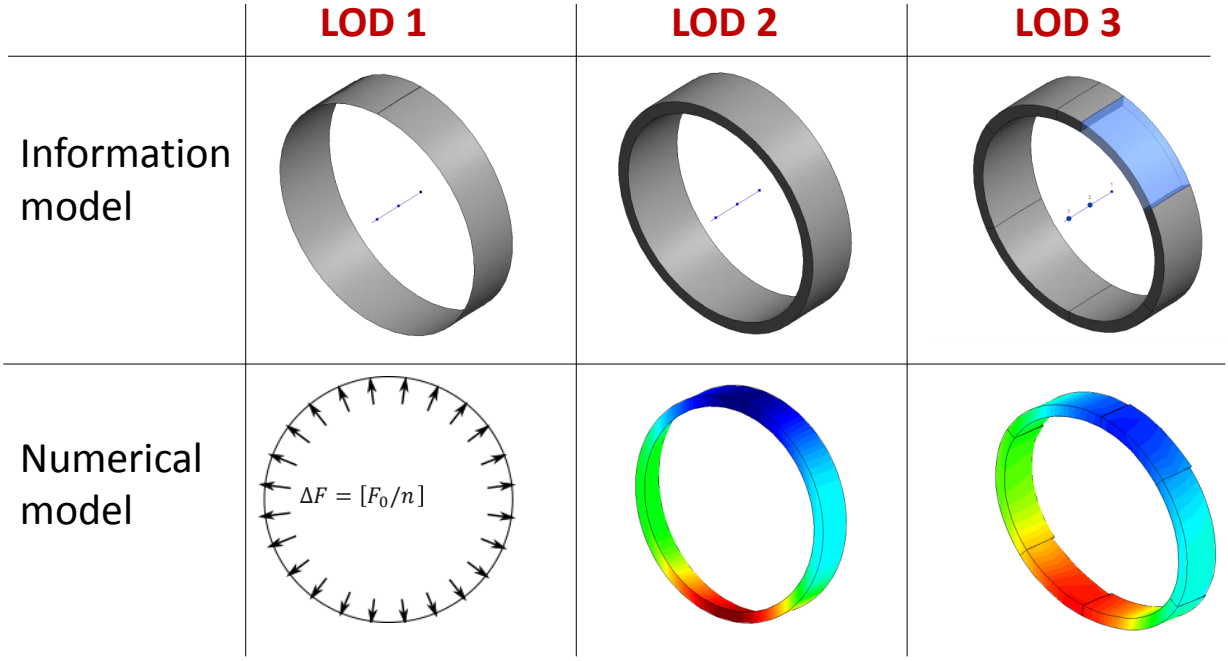


Figure 37: Lining information and numerical models on different level of details

Lining LoD1 When the tunnelling process is modelled with a 2D plane strain model, a simplified assumption must be made. This assumption allows the pre-displacement of the ground surrounding the tunnel boundary, prior to the structural element installation, to be taken into account. The available equivalent approaches that allow the deconfinement process due to tunnelling are: convergence-confinement method Panet and Guenot (1982), Gap method Rowe et al. (1983), Stiffness reduction method Swoboda (1979),Hypothetical modulus of elasticity Powell et al. (1997) and Volume loss method Dias et al. (2000). For the simple representation of shield tunnelling, i.e. confinement and and support by lining structure without explicit modelling of the lining structure, a Volume loss method is implemented. In this method the volume loss that will result from the completion of excavation will be prescribed, and the TBM passage, the injection process and the grout

consolidation phase were simulated considering the change in diameter of the excavation boundary. The method assumes that the support pressure at the tunnel boundary is reduced in increments, and the volume loss which is generated can be monitored. Equivalent nodal forces (σ_0) on the tunnel boundary, which represents the pressure exerted by soil to be excavated, are calculated and divided by the number of increments n, which simulates excavation stage. Equal but opposite forces ($-\Delta\sigma$) are then applied to the tunnel boundary over each n increments for the excavation stage. After installation of lining, ($-\Delta\sigma$) is still applied to the tunnel boundary for the remainder of the n increments. Fig. 38 shows the modelling process of the method.

It is noted that the 2D analyses only agree with the 3D analyses for certain choices of model parameters. The importance of the choice of parameters is emphasised byDias et al. (2000) who use a similar approach to modelling TBM tunnelling in 3D and find that on comparison with 2D predictions using the convergence confinement method, reasonable agreement was only reached once appropriate parameters were obtained by back analysis.

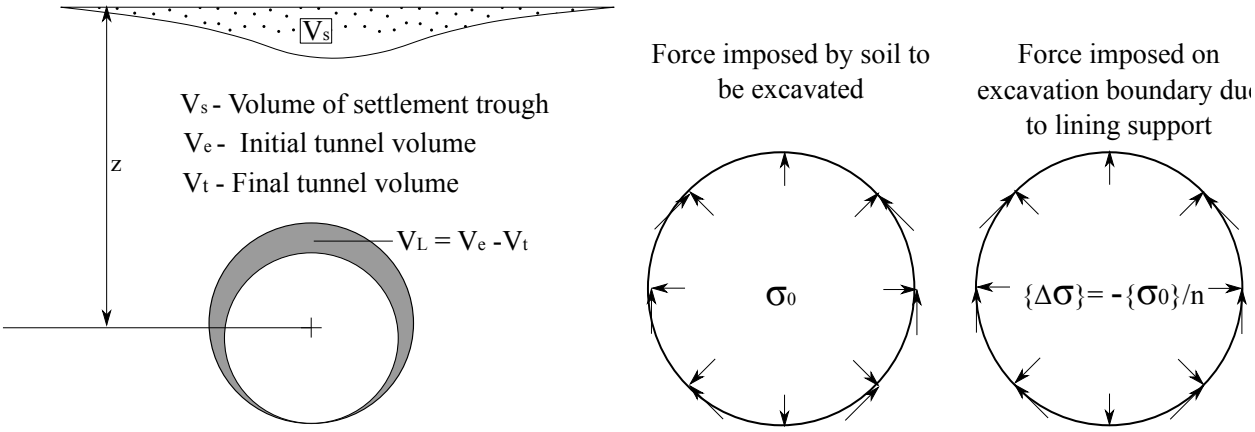


Figure 38: Volume loss method and modelling the excavation of the soil

There are two different approaches when modelling the volume loss. In the first approach the tunnel wall displacement was permitted, however the tunnel centre is assumed to be fixed, and therefore the shape of the tunnel wall is always circular during the deconfinement process. The tunnel wall displacement process is stopped when the volume loss reaches a specified value. These uniform displacements are in good agreement with the behaviour of a tunnel that is excavated through a with a ground mass the earth pressure coefficient at rest $K_0 = 0$ Hejazi et al. (2008).

Second technique is that the tunnel wall is allowed to move freely and is not controlled by the confinement forces or by pre specified displacement values. Instead after the deconfinement deformed area of the tunnel is continuously calculated at each computation cycle during the displacement of the tunnel boundary and deformations of the excavation boundaries are fixed when volume loss value of the tunnel boundary is reached Do et al. (2014b).

Lining LoD2 The segmented lining tube is modelled by means of volume elements that can be activated in course of the simulation advance. Each Lining ring is imported as a

single volume however, meshed with number of hexahedral kinematic linear elements. When simulating the tunnel advance each lining ring is activated in a stress free manner. This initialisation procedure is used to reset the reference configuration of the element. The new reference configuration of the reactivated element then matches the deformed state of the former structure.

Lining LoD3 In order to account for the reduced stiffness of the tunnel lining due to the presence of joints and for the segment-wise installation of the tunnel lining, a model for longitudinal and ring joints are proposed in the simulation model. Longitudinal and circumferential joints however have been modelled in a discrete manner. The reduced stiffens of segmental lining ring due to presence of joints is modelled by introducing bolts represented with beam elements and surface-to-surface normal contact condition between segments and transversal joins of the lining rings 37 LoD3. Bolts are embedded in the solid matrix representing lining segments, where tying conditions are imposed between the integration points of the beam elements and arbitrary point in the solid segment elements with the same global coordinates. Additional normal contact condition between the facing surfaces of the segments in longitudinal and transversal direction, prevents the penetration of one volume into another.

Grouting The tail void grouting has a considerably large effect on the change in the initial stress state of the soil around the tail, which finally causes surface settlements. In particular, the redistribution of the grouting mortar within the annular gap and the transition from liquid mortar in the beginning to solid state after its hydration plays a crucial role in maintaining the stress state of the surrounding soil and controlling the induced settlements. Therefore, in the simulation model, a constitutive model is applied that accounts for the time-dependent material behaviour of grouting mortar.

Within the simulation model, the pressurization of the grouting mortar is accounted for using a two-phase formulation similar to the soil as described in Section 3.2.1 LoD2/3. The hydration is described by time-dependent material properties for both the strength characteristics and the permeability. The formulation is based on the model for hydration of shotcrete proposed by Meschke et al. (1996). In this model the total strain $\boldsymbol{\varepsilon}$ is split into the initial strain $\boldsymbol{\varepsilon}^0$, the elastic strain $\boldsymbol{\varepsilon}^e$ and the irreversible strain due to time-dependent hydration $\boldsymbol{\varepsilon}^t$:

$$\boldsymbol{\varepsilon} = \boldsymbol{\varepsilon}^e + \boldsymbol{\varepsilon}^t + \boldsymbol{\varepsilon}^0 \tag{30}$$

The constitutive law is basically a linear elastic Hooke's law with time-dependent Young's modulus $E(t)$ that represents the hydration of the mortar and cause the irreversible strain $\boldsymbol{\varepsilon}^t$. Hence, the strain-stress behaviour is defined with the irreversible strain and material tensor after 28 days $\mathbb{C}^{(28)}$, assuming that the stress rate linearly depends on the strain rate in terms of the time-dependent material tensor $\mathbb{C}$ as:

$$\text{(a) } \dot{\boldsymbol{\sigma}}' = \mathbb{C} : \dot{\boldsymbol{\varepsilon}} \qquad \text{where} \qquad \text{(b) } \mathbb{C} = \mathbb{C}^{(28)} \frac{E(t)}{E(28)} \tag{31}$$

This leads to the development of the irreversible strains in time increment Δt:

$$\Delta\boldsymbol{\varepsilon}^t = (1 - \frac{\xi}{E^{(28)}\Delta t})\Delta\boldsymbol{\varepsilon} \tag{32}$$

At the end of a time increment, the stress is calculated in terms of the material tensor $\mathbb{C}^{(28)}$ and the total strain (Eqn. 30), updated by means of the irreversible strain (Eqn. 32). The hydration of the mortar is modelled by assuming the time-dependent permeability using an exponential relation between the initial permeability of the mortar $k^{(0)}$ and the permeability after 28 days $k^{(28)}$ with the transition coefficient β:

$$k(t) = (k^{(0)} - k^{(28)})e^{-\beta_{grout}t} + k^{(28)} \tag{33}$$

Using such a formulation, it is possible to account for the pressurization of the grout by means of stepwise installation and prescription of equivalent pressure boundary conditions on the grouting element.

3.2.4 Building Models

In some cases, for design purposes, the interest is not in the detailed distribution of stresses within the structural members, but in the distribution of averaged quantities such as bending moments, axial and shear forces, while clearly keeping the accuracy of the global structural response. If the focus is on tunnelling-induced settlements due to interaction with existing structures rather than the influence of settlements on the damage of the structures, the reduced models for structures can be used. Otherwise, if the target is to access the effect of tunnelling on structural behaviour, the detailed models of structures are required. Selection of the optimal LoD based on objective of the analysis proposed in SATBIM fulfils those requirements. Selected LoDs for representation of buildings are chosen such way that lowest LoD will not introduce any additional DOFs, however, represent the buildings by means of additional stresses due to building weight, while the higher LoD has detailed representation of the building structure and include all soil-structure interaction effects.

LoD1 Building is substituted with a dead load from building weight acting on the soil surface. In this model the effect of the soil-structure interaction and building stiffness are neglected.

LoD2 Buildings are considered in the tunnelling model by means of reduced models with a substitute elastic stiffness E, height H and weight ρ computed according to an approach proposed in Schindler and Mark (2013). In the presented FE formulation, isotropic volume hexahedra kinematic linear elements are adopted with respective structural properties, interacting with the soil through a mesh independent surface-to-surface contact algorithm, which prevents the penetration of the foundation of the building into the soil . It also takes into account different mechanisms of the soil-structure interaction corresponding to "sagging" and "hogging" modes. The geometry of the buildings is imported from the TIM as described in Section 2.7 and illustrated in Fig. 29.

LoD3 Buildings are modelled as a full structural frame models. The columns and floors are both modelled with isotropic volume hexahedra kinematic linear elements. In order to control number of DOFs, a custom structured mesh is generated. Since foundations

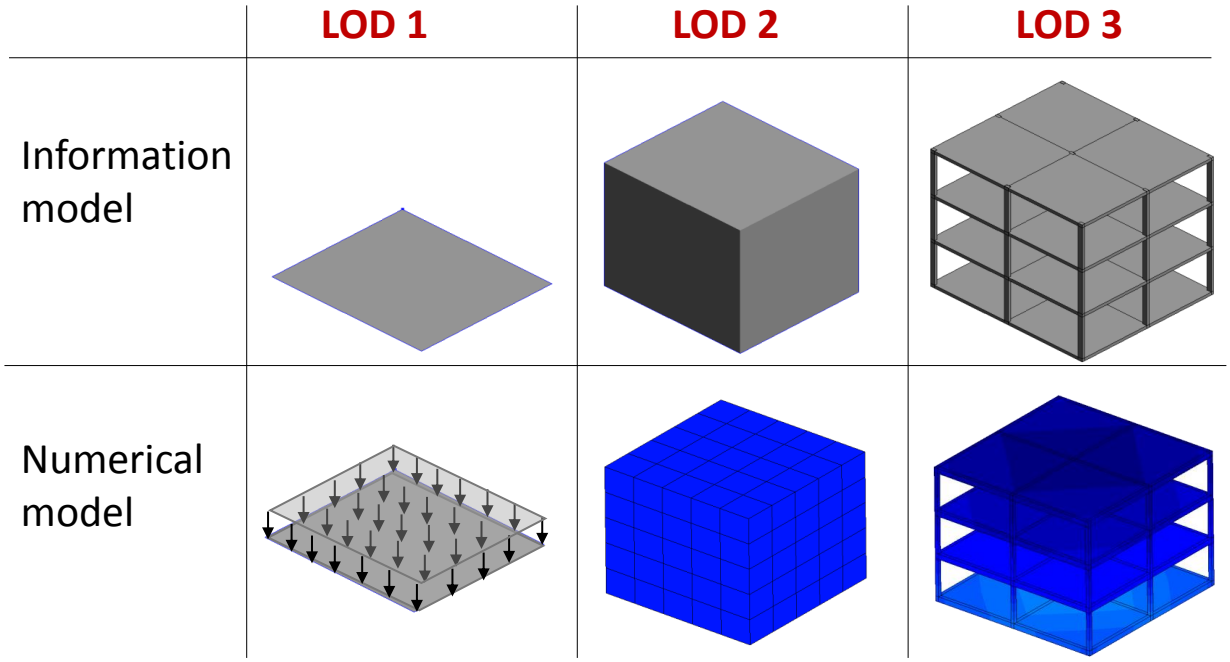


Figure 39: Building information and numerical models on different level of details

play a fundamental role in the transmission of the ground deformations to the building, to simulate such relative movements between the soil and foundation, surface-to-surface contact conditions are introduced.

3.2.5 TBM Models

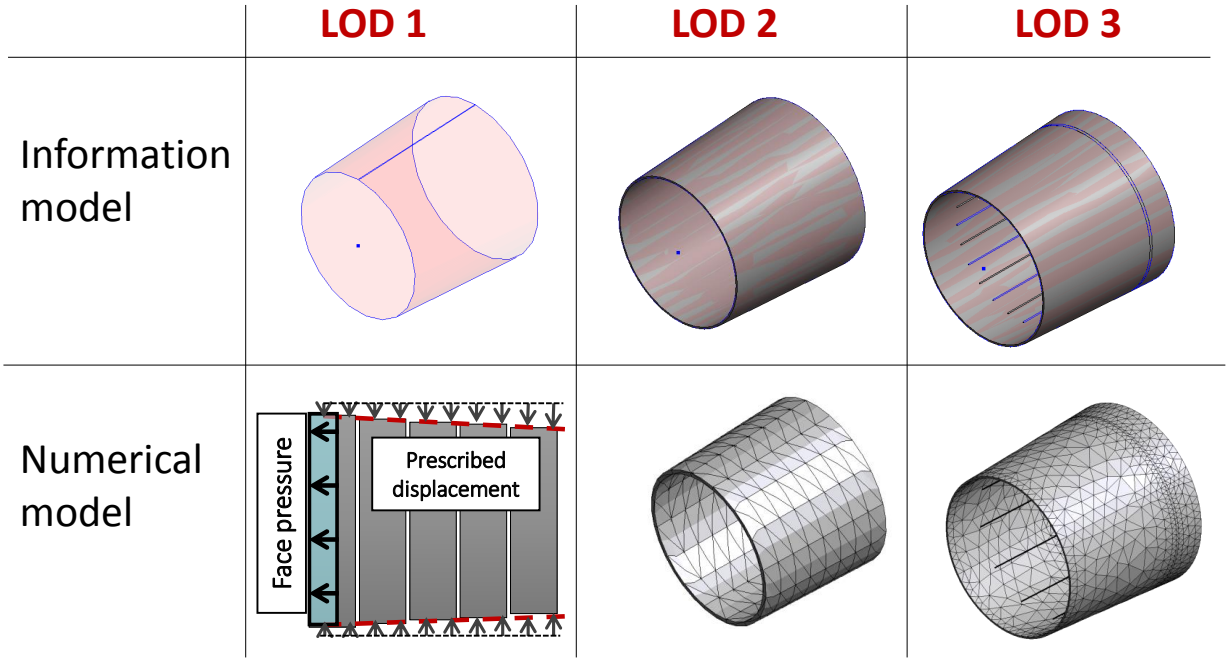


Figure 40: Tunnel Boring Machine (TBM) information and numerical models on different level of details

LoD1 Since the main function of the shield is preventing that the material around the excavation area converges towards excavation, some authors represented TBM exactingly as boundary condition limiting deformation of the soil Founta et al. (2013). In this approach shield is represented by a set of n ring length (L_rm) long segments with uniform radial displacement-defined boundaries that approximate the conical surface of the shield, where $n = L_{TBM}/L_r$. There are also equivalent approaches of modelling the TBM without physical FE model of the shield. In the second approach, after excavation process, the soil is let to deform freely around the TBM, and displacements at soil nodes surrounding TBM are measured. If the displacements in nodes exceed the value of the gap between TBM and soil, the displacement nodes are fixed to the value of the gap. Both approaches are implemented and available for SATBIM project.

LoD2 The TBM is modelled as a deformable body moving through the soil and interacting with the ground through surface-to-surface contact. By virtue of this modelling approach, the volume loss due to the excavation process naturally follows the real, tapered geometry and the over-cutting of the shield machine. The frictional contact between the shield skin with the surroundings plays an important role for the re-distribution of stresses and pore pressures in the soil, therefore is modelled by means of surface-to-surface contact formulation introduced by Laursen (2002).

Contact is accomplished by means of a surface-to-surface contact formulation introduced by Laursen (2002). The contact formulation imposes a geometric constraint between the contacting ("slave") body (the TBM) and the contacted ("master") body (the soil) which controls the interaction between the two bodies with independent deformations. The surface-to-surface contact algorithm is based on the fulfilment of the contact constraint at each quadrature point on the so-called slave contact surface. The physical requirement of impenetrability is stated in a discrete manner in terms of the gap g between the slave and contact surface.

Accounting for frictional slip in the contact formulation requires special consideration. Slip is considered as the relative movement between slave and master contact partners. The frictional response is introduced using the Coulomb friction law, stating that the tangential displacement $\mathbf{u}_T$ is proportional to the tangential stress $\boldsymbol{\sigma}_T$ if $\mathbf{u}_T \geq \mu\sigma_N$, otherwise equal to 0 (Laursen, 2002).

The displacements of the TBM are prescribed at the TBM end, and the direction of the advance is determined by the calculated tunnel alignment vector.

LoD3 In the highest LoD, the advance of the TBM is pushed forward by elongation of the hydraulic jacks, excavating the soil by a rotating cutting wheel. In order to realistically model the movement of the TBM and its interaction with the soil, to avoid drift off-course of the TBM and to simulate curved tunnel advances, an automatic steering algorithm to control the individual jack thrusts similar to the one proposed in (Kasper and Meschke, 2006) has been used to keep the TBM on the designed alignment path. Identical to LoD2, the interaction between the soil and TBM skin is modelled applying frictional-surface-to-surface contact interface.

3.3 Examples of multi-level numerical models for tunnelling

In the following subsection the setup and results of the numerical simulations including different LoDs of individual components and their combinations are shown.

3.3.1 Soil LoD1 and Lining and Building models on different LoDs

In this example soil is modelled using subgrade reaction method (LoD1), however, for lining and building structural models are used. Here, in both cases lining structure is loaded with in situ stress applied as external distributed surface loading, neglecting the soil-structure interaction and the weight of the excavated soil. In Alg. 5 is shown how Python interface for the simulation software KRATOS is used to apply the Dirichlet and Neumann boundary conditions on the outer lining surface for both, lining LoD1 and LoD2.

The applied conditions are dependent on the alignment of of the tunnel, since in subgrade reaction model, the elastic springs as well as applied loadings act perpendicular to the lining surface, and in KRATOS, all conditions are applied in global Cartesian coordinate system. Therefore, in Alg. 5 the transformation of the condition quantities, surface loads and elastic

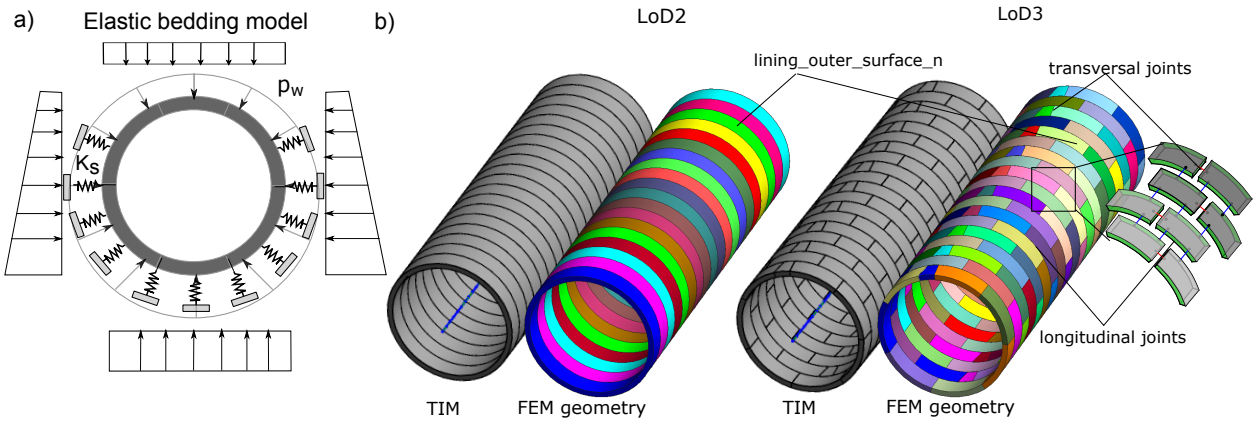


Figure 41: Subgrade reaction model for lining: a) Dirichlet *and* Neumann *boundary conditions; b) geometry of the models on LoD2 and LoD3 with main logic layers*

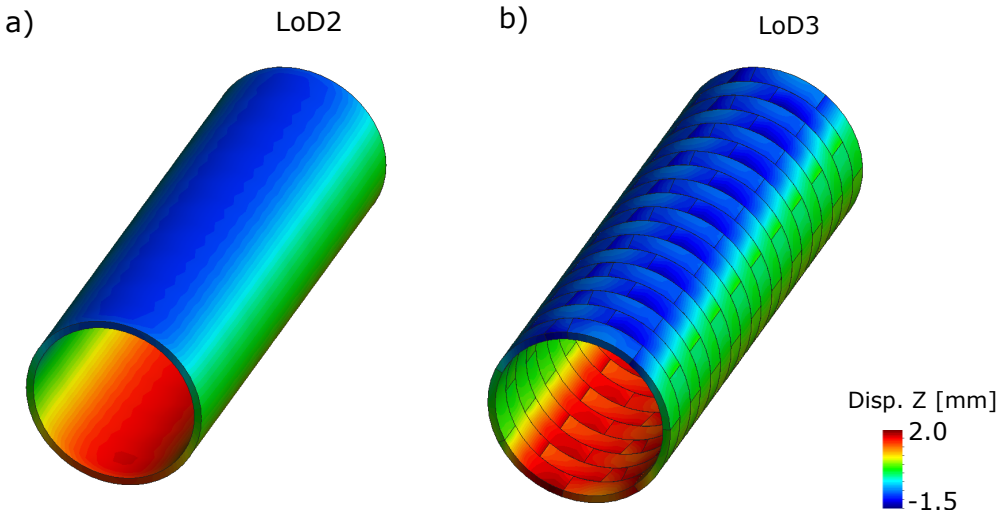


Figure 42: a) Vertical deformation of the lining LoD2 for subgrade reaction model; b) Vertical deformation of the lining LoD3 for subgrade reaction model

bedding stiffness, from the local coordinate system characteristic for the each point of the calculated tunnel alignment to global coordinate system is calculated and conditions are applied onto the lining outer surface for respective ring *lining_outer_surface_n*.

For lining representation on LoD2 the individual rings are kinematicaly compatible, i.e. neighbouring rings share nodes on their contact interface. On contrary, lining on LoD3 each segment is kinematicaly independent form it neighbouring segment. This is achieved by introducing surface-to-surface contact interface, which prevents the penetration of one segment volume into another. Moreover, to model the joint connections between the segments,the connection between the segments in longitudinal and transversal direction are enhanced with bolts, represented with the beam elements (see Fig. 41). Between each two segments connection, a beam is embedded within both segments, and beam nodes are tied with arbitrary point in segment domain with same global coordinates by means of penalty constraint. The contact conditions between individual segments by means of normal contact

Listing 5: Assigning subgrade reaction boundary conditions

```
for step in range (1, number_of_slices+1):
        delta_x=(float(alignment[step][0])-float(alignment[step-1][0]))
        delta_y=(float(alignment[step][1])-float(alignment[step-1][1]))
        alpha=math.atan(delta_y/delta_x)
        for node_id in model1.layer_nodes_sets['lining_outer_surface_'+str(step)]:
                node = model1.model_part.Nodes[node_id]
                point=[node.X0,node.Y0,node.Z0]
                for i in range (0,3):
                        radius[i]=float(alignment[step][i])-point[i]
                theta=math.atan2( radius[1], radius[2] )
                depth = node.Z
                pressure = -(overburden*9.81*ro- depth*9.81*ro)
                pressure_z=pressure *math.cos(-theta)
                pressure_y_1=pressure *math.sin(-theta)
                pressure_x=pressure_y_1*math.sin(alpha)
                pressure_y=pressure_y_1*math.cos(alpha)

#assign soil insitu pressures
                node.SetSolutionStepValue( FACE_LOAD_Z, -pressure_z )
                node.SetSolutionStepValue( FACE_LOAD_Y, K0*pressure_y )
                node.SetSolutionStepValue( FACE_LOAD_X, K0*pressure_x )

#assign elastic bedding stiffness - subgrade reaction coefficient
                eb_z=k_spring*math.cos(-theta)
                eb_y_1=k_spring*math.sin(-theta)
                eb_x=eb_y_1*math.sin(alpha)
                eb_y=eb_y_1*math.cos(alpha)
                node.SetSolutionStepValue(ELASTIC_BEDDING_STIFFNESS_X, abs(eb_x))
                node.SetSolutionStepValue(ELASTIC_BEDDING_STIFFNESS_Y, abs(eb_y))
                node.SetSolutionStepValue(ELASTIC_BEDDING_STIFFNESS_Z, abs(eb_z))
```

and bolt connections are applied calling following KRATOS methods using Python interface:

From deformations of the lining rings modelled on LoD2 and LoD3, shown in Fig. 42, it can be seen that for the same loading and bedding conditions the deformation of LoD2 and LoD3 lining model significantly differs. The joints introduced in LoD3 reduce the stiffness of the system and allows to certain extent the rotation of the segments. Moreover, if required, in LoD3 lining model, the concentrations of stresses on the contact surfaces between the segments can be observed. However, more detailed representation in LoD3 is paid with higher computational costs, which can scale exponentially with increase of number of DOFs when using linear solvers.

The similar approach as modelling the lining with subgrade reaction method can be applied to Buildings (see Fig. 43). Here the elastic bedding is applied to the foundation, i.e. bottom surface of the building.

Listing 6: Assigning contact conditions between lining segments

```
#assign tying conditions between bolts and segments
segment_elements=[]
for i in range (1,number_of_slices+1):
        for j in range (1,8):
                for element in model1.layer_sets['lining_'+str(i)+'_'+str(j)]:
                        segment_elements.append(element)

beams_node_list=[]
for i in range (1,number_of_slices+1):
        for j in range (1,8):
                for node in model1.layer_nodes_sets['bolts_'+str(i)+'_'+str(j)]:
                        beams_node_list.append(node)

Epltu = EmbeddedNodePenaltyTyingUtility()
links1 = Epltu.SetUpTyingLinks( model1.model_part, beams_node_list, segment_elements )
for cond in links1:
        cond.SetValue(INITIAL_PENALTY, 1.0e12)

#assign normal contact constraint between segment surfaces with given IDs
for i in range (1,number_of_slices+1):
        model1.solver.solver.contact_tying_indices[1000+i] =
        'contact_link_kinematic_linear_penalty'
        model1.solver.solver.mortar_contact_utility.SetValue(MAXIMAL_DETECTION_DISTANCE, 0.1)
        model1.solver.solver.mortar_contact_utility.SetValue(GAP_TOLERANCE, 1.0e-10)
        model1.solver.solver.Parameters['penalty'][1000+i] = 1.0e10
```

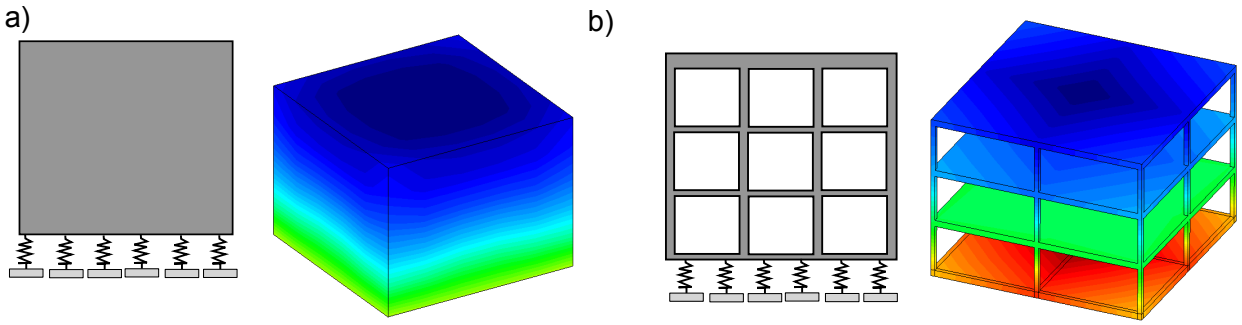


Figure 43: Geometry of the building model with vertical displacements for representation of soil with subgrade reaction model for: a) Building LoD2 and b) Building LoD3

3.3.2 Lining LoD1 - Volume Loss method

In this example soil is represented with LoD2 (or LoD3 would be equivalent in terms of FEM model, however, topology would be exact) while lining is represented with LoD1. As already explained in Section 3.2.3 Lining is not represented with a structural model, but instead, the TBM and lining support is described with equivalent volume loss which will be permitted due to excavation of the soil. The permitted deformation of the excavation boundary is described with parameter *volume_loss*. The simulation of the tunnel excavation consist of three main steps:

- calculation of the insitu stresses in soil model
- transfer of the insitu stresses onto main model
- step by step deactivation of the excavation volumes and application of volume loss boundary conditions and heading face pressure

Listing 7: Determination and transfer of the insitu stresses in the soil

```
#determination of the insitu stresses in virgin model
isu = InSituStressUtility()
isu.SetInSituStressFromCurrentStress( model_virgin.model_part,
 model_virgin.model_part.ProcessInfo )

#transferring of insitu stresses to the main simulation model
vtu = VariableTransferUtility(MKLPardisoSolver())
vtu.TransferPrestress( model_virgin.model_part, model1.model_part )

# scaling of the pre-stresses to 0, if they are transferred to other structural components
isu.ScalePrestress( model1.model_part, model1.layer_sets['foundation'],
 len(model1.layer_sets['foundation']), 0.0 )
```

In the first step of the simulation of tunnelling process a in-situ stress field in an undisturbed model of the ground is determination. The results of this initial simulation, namely the prestresses, are then used to define the initial stress state of the main simulation model. The Python script where the insitu stresses are calculated in $Insitu_model$, and transferred to original model is given in Alg. 7.

The volume loss boundary conditions can be applied with 2 different algorithms. In first radial displacements are applied on the excavation boundary, equivalent to prescribed volume loss. This method is used for the examples shown below. In second algorithm soil is deforming freely after removing the excavation boundary and displacements are measured and respective volume loss is calculated. When defined volume loss is reached the displacements are fixed. Both Algorithms are given in Appendix in Alg. 15 and Alg. 16 respectively.

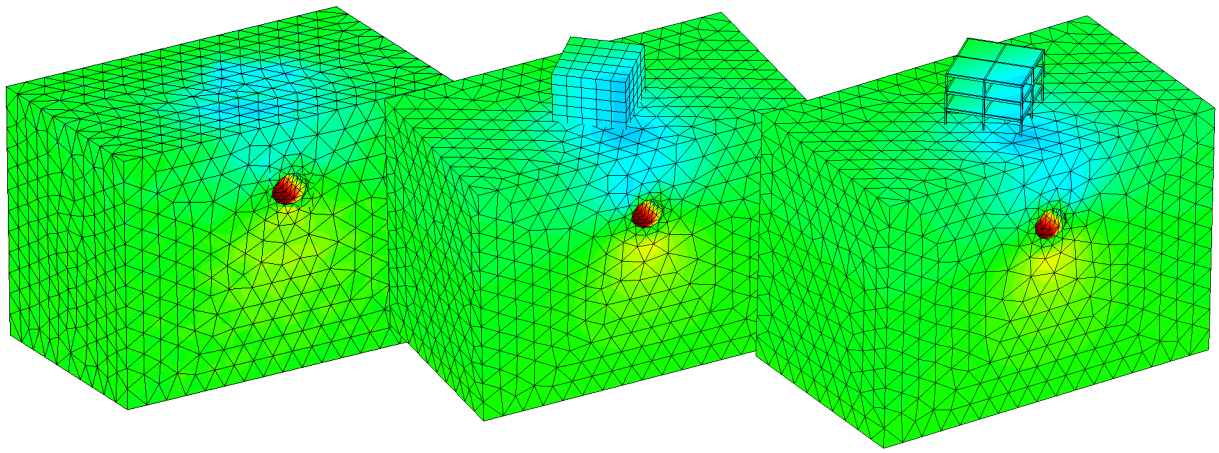


Figure 44: Vertical deformations of the soil induced by tunnelling for building representation with LoD1, LoD2 and LoD3. Lining support is represented by volume loss method.

In the examples in Figures 45 and 46 , again the volume loss method is combined with representation of buildings on different level of detail (LoD1, LoD2 and LoD3). The goal here is to show how the distance of building from the tunnel axis influences surface settlements.

The LoD of a building is varied from LoD 1 (dead load from building weight acting on the soil

surface) over LoD 2 (reduced models with a substitute elastic stiffness E, height H and weight ρ computed according to an approach proposed in Schindler and Mark (2013)) to LoD 3 (full structural frame model), cf. Fig. 45a. As a simple representation of shield tunnelling, i.e., confinement and support by lining without explicit modelling of the lining structure, the volume loss method is used. For this method, the confinement is described with the volume loss coefficient $V_l = (V_0 - V_{def})/V_0 \cdot 100\%$. In the implementation of the volume loss method, after the deconfinement, the deformed area of the tunnel is continuously calculated at each computation cycle, and deformations of the excavation boundaries are fixed when the volume loss value of the tunnel boundary is reached Do et al. (2014b).

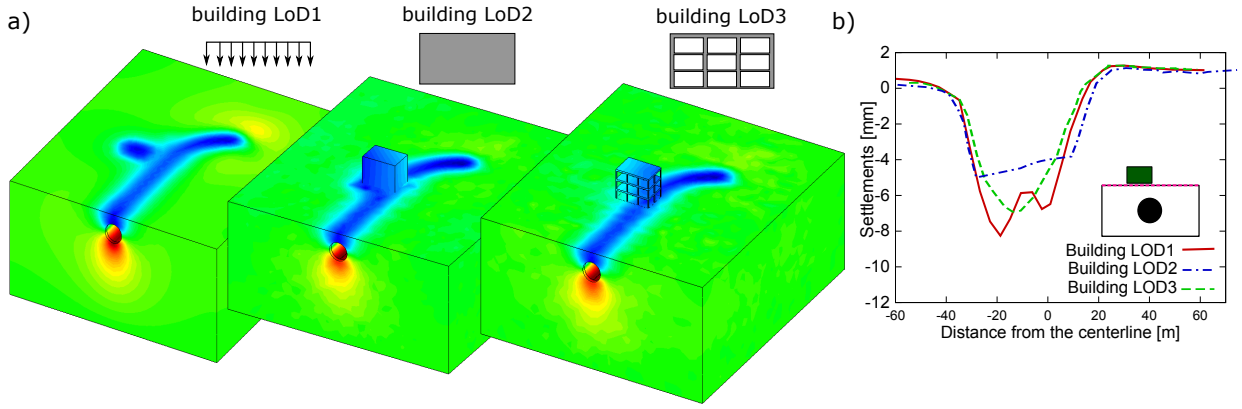


Figure 45: Effect of different LoDs for building on surface settlements using the volume loss method: a) geometry and vertical displacements; b) surface settlements.

Figure 45b shows the effect of the choice of the LoD for the building, where the maximum settlements are obtained for building LoD 1 due to the negligence of the building stiffness in soil-structure interaction. In contrast, for building LoD2 , this effect is overestimated. However, if the building is far from the influence of the tunnelling-induced settlements, the choice of the LoD becomes insignificant, i.e., the simplest representation is sufficient (see Figure 46). This model can be further used to evaluate the importance of LoD choice based on the building type and position w.r.t. the tunnel in detail in order to come up with suggestions for the optimal LoD selection.

Figure 46 shows that if building is located far from the tunnel alignment, i.e. far from the influence of the tunnelling-induced displacement, the model used for representation of the building plays no role. The quantification of the factors determining this adequate choice of the building LoD, as well as the LoD of other components, will be the subject of further research. The choice of the lover LoD can significantly reduce the computational costs.

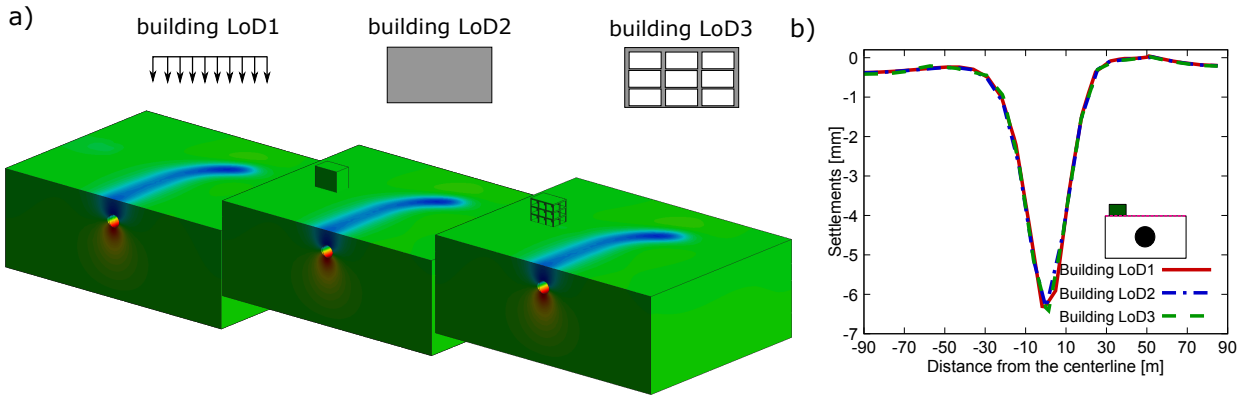


Figure 46: LoD of the building does not affect surface settlements if building is far from the tunnel: a) geometry and vertical displacements; b) surface settlements.

3.3.3 TBM LoD1/2/3 with excavation process

In this subsection, very briefly will be introduced how the TBM is represented within the simulation model for different LoDs, while the parts of actual implementation are given in Appendix 7.

TBM LoD1 TBM LoD1 is not represented with a structural model, but instead with set of boundary conditions representing the shield. In Fig. 47(a) is shown how the machine is substituted with radial displacement boundary condition along the soil excavation boundary surface. The soil excavation is performed in steps, and therefore the TBM is moved forward for each excavation step, and lining structure and grouting support are instilled. The TBM boundary condition is applied under L_{TBM}/L_{ring} excavation surfaces, and here, a mid pint rule is used when determining the gap between TBM and the soil for each ring. The Python

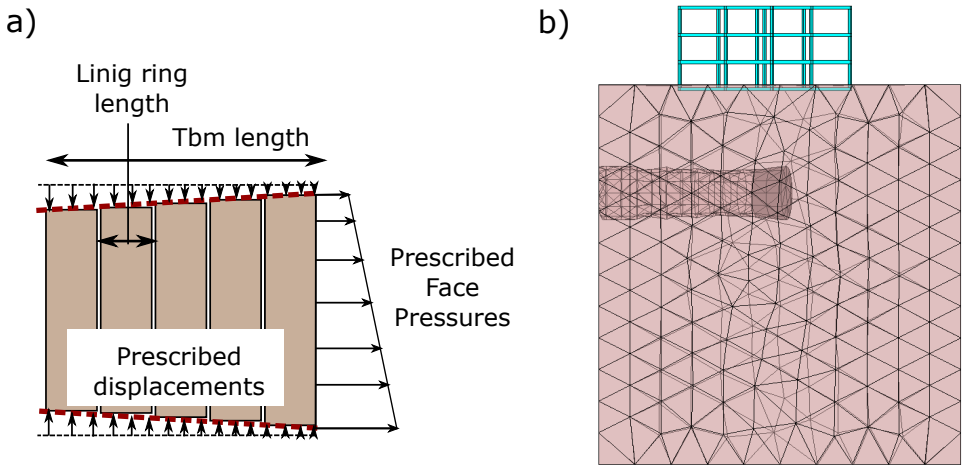


Figure 47: a) Sketch of the boundary conditions for representation of TBM LoD1 b) Deformation of the soil during shield tunnelling where the TBM is represented with LoD1

script for implementation of the TBM on LoD1 is given in Appendix 7 in Alg. 17.

Listing 8: Setup of hydraulic jacks and steeriing of TBM

```
#setting up steering utility
steering_utility = SteeringUtility()
with_diagonals= False
jacks = __import__( self.name+"_jacks" )
jacks_obj = jacks.HydraulicJacksPositions()
jack_locations= jacks_obj.coords

steering_utility.InitializeSteeringUtility(model1.model_part, jack_locations,
 len(jack_locations),with_diagonals)
steering_utility.ResetHydraulicJacks(model1.model_part,with_diagonals)
steering_utility.SetHydraulicJacks(model1.model_part,with_diagonals)

#Move TBM and solve afterwards
steering_utility.Move( advance, model1.model_part,with_diagonals)

#Reset hydraulic jacks and solve afterwards
steering_utility.ResetHydraulicJacks(model1.model_part, with_diagonals)
```

TBM LoD2 On the second LoD TBM is represented with a structural model interacting with the soil excavation boundary through the surface-to-surface algorithm. In order to simulate TBM advance, displacements boundary conditions are now prescribed to TBM. Here the displacement prescribed are following the calculated alignment path as shown in Fig. 48. Since prescribed contact interface permits penetration of the two body volumes, the soil deforms until it reaches the TBM outer surface, and there is contained. The Python

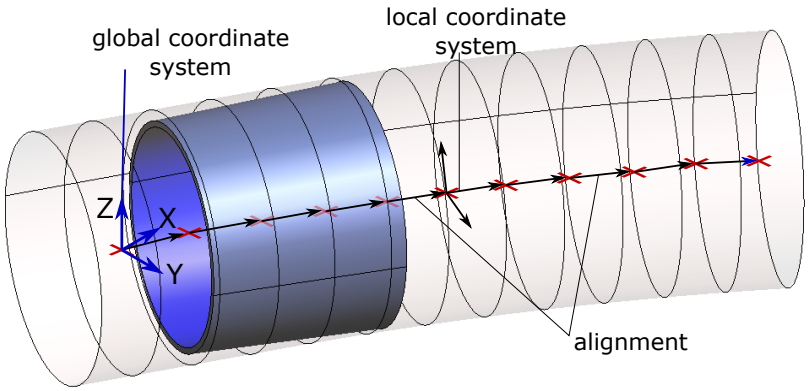


Figure 48: Structural model of TBM on LoD2 with illustration of the alignment path

script for modelling of the TBM on LoD2 is given in Appendix 7 in Alg. 18.

TBM LoD3 In this representation of TBM, shield is represented as a structural deformable body, however, the TBM advanced is controlled by elongation of hydraulic jacks. The setup of hydraulic jacks and steering algorithm is assessed via following Python script:

3.3.4 Lining LoD2 and Soil LoD2/3- Installation of lining and grouting

In this simulation model lining is represented with structural model as shown also in Example 3.3.2, however, since here the soil is also represented with structural model, we are able to simulate excavation process with stepwise excavation of the soil and installation of the lining and grouting, which is here also a structural models. The simulation is preformed in following steps:

- calculation of the insitu stresses in soil model
- transfer of the insitu stresses onto main model
- For each advance step:
  - Excavate the soil and move TBM in next position in n moving steps
  - Install Lining ring and grouting ring
  - Assign grouting pressure and heading face pressure
  - After tunnel section is completed, depending on soil type preform n calculation steps of consolidation process

In this simulation the calculation of if insitu stress is performed identical as in Example 3.3.2, while modelling of TBM can be accomplished by application any model shown in Example 3.3.3. The additional feature here is that after each soil excavation step the lining ring is installed (activated), and corresponding support measures are applied. The steering gap behind the TBM is filled with grouting and this is simulated by activation of the grouting elements and application of water pressure boundary conditions representing the pressurisation of the tail gap with high "injection" pressures. The face is supported with heading face support as schematically shown in Fig. 47(a).

Application of support measures The annular gap between the segmented lining tube and the excavation boundary is assumed to be refilled with cement-based grouting material, modelled as a fully saturated two-phase material with a hydrating matrix phase, considering the evolution of stiffness and permeability of the cementitious grout Kasper and Meschke (2006) (see Sec. 2.3.4). To provide the stability of the tunnel face due to distortions caused by the excavation process and to reduce ground loss behind the tapered shield, the face support pressure and the grouting pressure are applied at the tunnel face and in the steering gap, respectively. For EPB shields, the earth pressure is applied as a face load applied to the heading face following the deformations of the mesh such that the face pressure always acts perpendicular to the heading face.

In this simulation model, both support and grouting pressure have liner change over the height for prescribed *gradient* value.The pressure gradient is determined according to input parameters stating the design pressure and the weight of the earth compound used for the face support. Assignment of face support measures trough Python script is given in Appendix in Alg. 19.

Consolidation The deformation of the ground due to tunnelling not only depends on the ground loss, but in the case of clay soils also strongly on time. In a saturated soil, the deformations can be retarded by the time that it takes for the water to flow out of

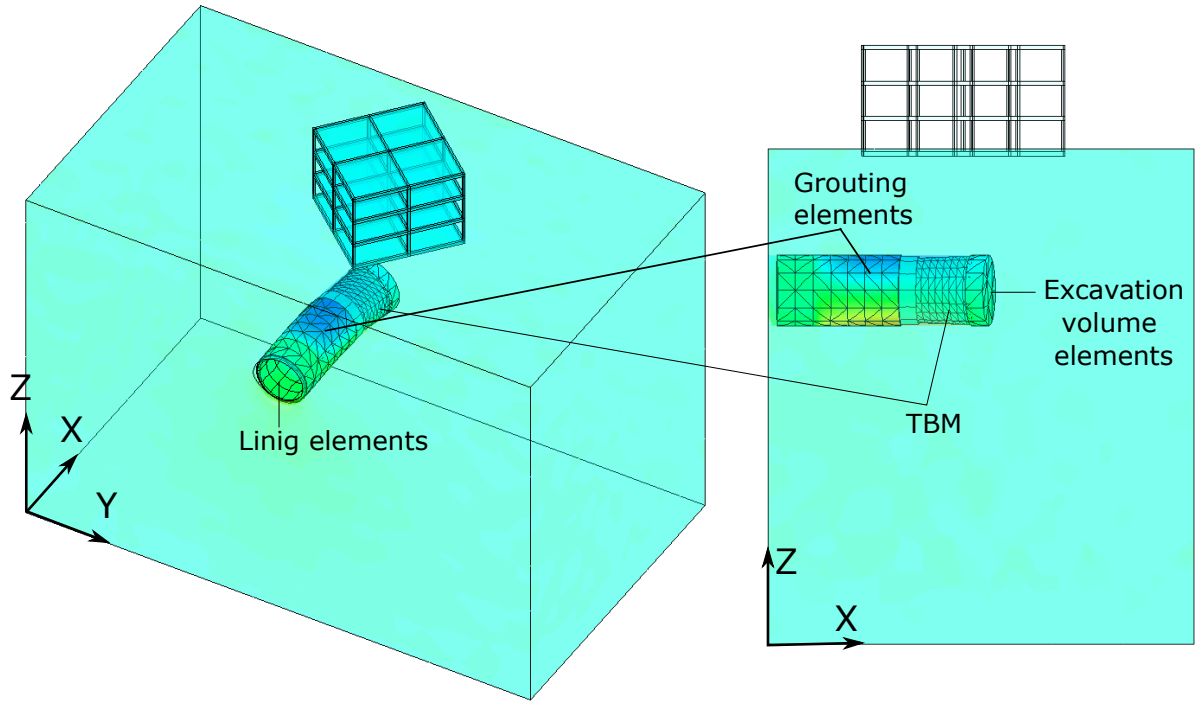


Figure 49: Vertical displacements due to tunnel construction process for the Soil LoD2, Lining LoD2, TBM LoD2 and Building LoD3

the soil. The high compression stresses in the soil cause closure of the soil pores and the water outflow, which may take a considerable time for the soil with low permeability. The process is called consolidation. Theoretically, consolidation is an infinite process. However, for most engineering applications, it can be considered that the consolidation is complete when 99% of the final deformation have been reached, which can be calculated wit following equation Verruijt (2012):

$$t_{99\%} = \frac{2h^2}{c_v} = \frac{2h^2(m_v + n\beta)\gamma_w}{k} \tag{34}$$

If characteristic consolidation time is larger than excavation steps for the tunnel section, to calculate final tunnelling induced displacements after the simulation of the construction of tunnel section is completed , further calculation step have to be performed, accounting for dissipation of pore water pressure in time.

The simulation model including all above mentioned features is created based on tunnel information model. The results of vertical displacements induced by stepwise tunnel construction are shown in Fig. 49.

3.3.5 Lining LoD3 within complete simulation model

In this example the very complex simulation with segment-wise installation of the lining is generated using the TIM. The lining structure is modelled on the highest LoD (3), and besides in simulation model the installation of the lining structure is preformed segment-by-segment, where the connection between the segments is established by means of steal

bolts modelled with the beam elements and surface-to-surface normal contact conditions in longitudinal and transferral direction. The simulation is performed on the similar manner as in the the previous example (see Example 3.3.4), with the difference that the lining ring is not installed as a single unit, but segment-by-segment, as shown in Figure 50. Therefore,

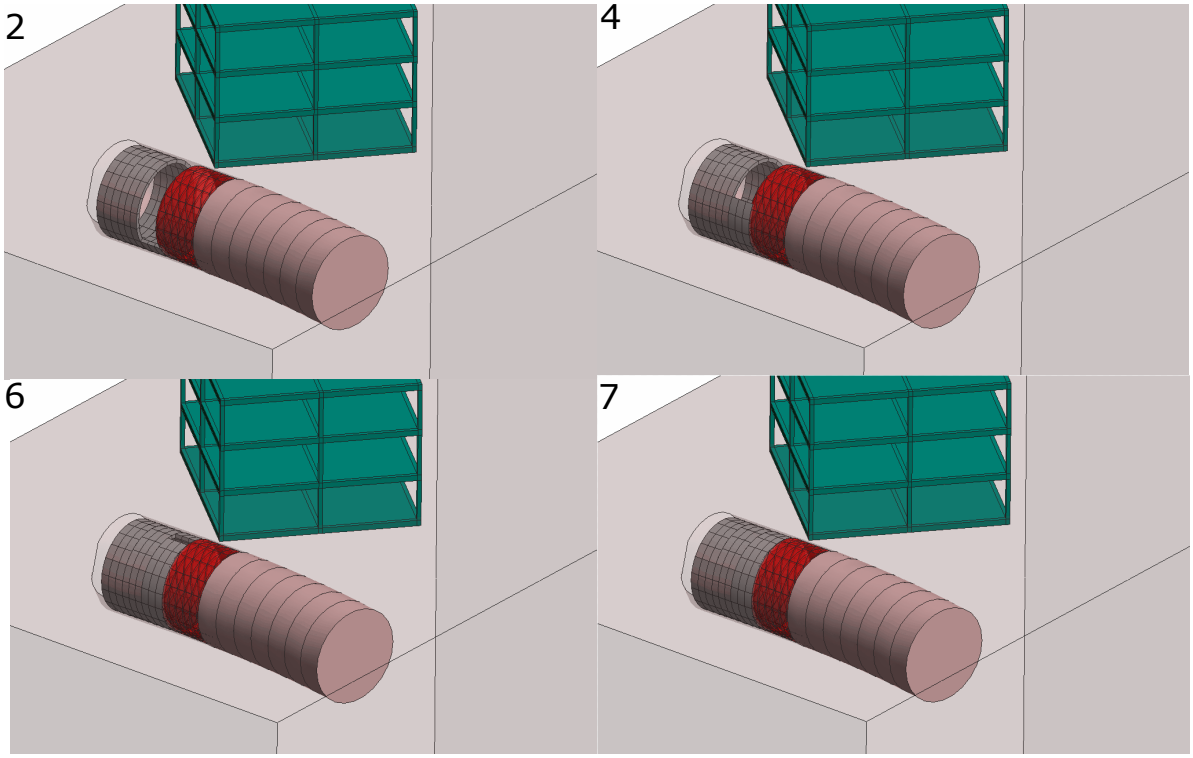


Figure 50: Geometry of full 3D model with the highest LoD representation of the building and lining structure with segment-wise installation of lining ring.

general steps in such simulations are now listed as:

- calculation of the insitu stresses in soil model
- For each advance step:
  - Excavate the soil and move TBM in next position in n moving steps
  - For each lining segment:
    - Install segment and bolts and sign contact interfaces
  - Install grouting ring
  - Assign grouting pressure and heading face pressure

Therefore, the heading face support pressure, the grouting elements and the grouting pressures are applied the same way as explained in previous example. The contact conditions between longitudinal and traversal joints are enforced the same way as in example for segmental lining given in Listing 6. The algorithm for the assignment of the conditions and activation of the lining segments into ring is given in Listing 9.

Listing 9: Method 1: Assignment of the conditions and activation of the lining segments into ring

```
# segment-wise installation
for i in range (1,segment_number+1):
        #cloning time step
        time = time + exc_delta_time/segment_number
        #set bolt index
        #inforcment of tying between bots and segments
        for node in model1.layer_nodes_sets['bolts_'+str(active_bolt_index)+'_'+str(i)]:
                beams_node_list.append(node)
        Epltu = EmbeddedNodePenaltyTyingUtility()
        links1 = Epltu.SetUpTyingLinks( model1.model_part, beams_node_list, segment_elements)
        for cond in links1:
                cond.SetValue(INITIAL_PENALTY, 1.0e12)
        #reactivate lining segment
        reactivation_index_stressfree = reactivation_index_stressfree + 2
        reactivation_index = reactivation_index + 2
        model1.deac.ReactivateStressFree( model1.model_part, initial_reactivation_index,
        reactivation_index )
        model1.deac.Deactivate( model1.model_part, 0, deactivation_index )
        print('done...')

        #solve and consolidate results
                time = time + exc_delta_time
                model1.model_part.CloneTimeStep( time )
                print('solving...')
                model1.solver.Solve()
                model1.WriteOutput( time )
        print('##############')
                print('step_'+str(step)+'_done.')
```

The simulated segment-wise installation of the lining ring is shown in Figure 51. Form the steps shown in this figure we can see how the deformation and consequently strains and stresses will change during the installation process. This is significant to consider if the target is to observe in detail the lining structure during the contraction process.

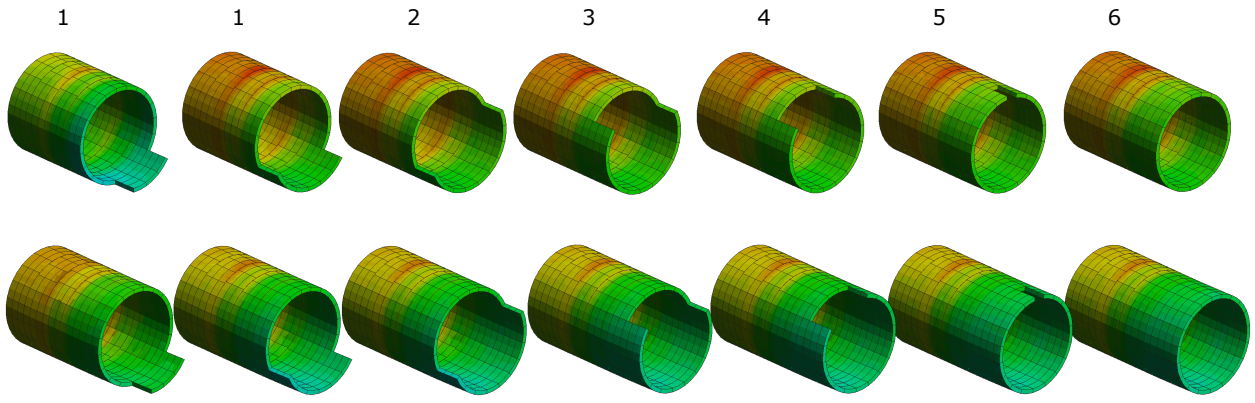


Figure 51: Simulation of segment-wise installation of tunnel structure.

Although the accuracy of representation of the lining structure will have significant influence on it structural response during tunnel construction Figure 52 shows that the LoD of the lining has insignificant influence on the tunnelling-induced settlements. This is due to fact that the gap between the lining and the soil is refilled with the pressurised grouting and that

the soil deformation in the vicinity of the tunnel, as well as displacements which propagate to the soil surface, will be mainly controlled by this factor.

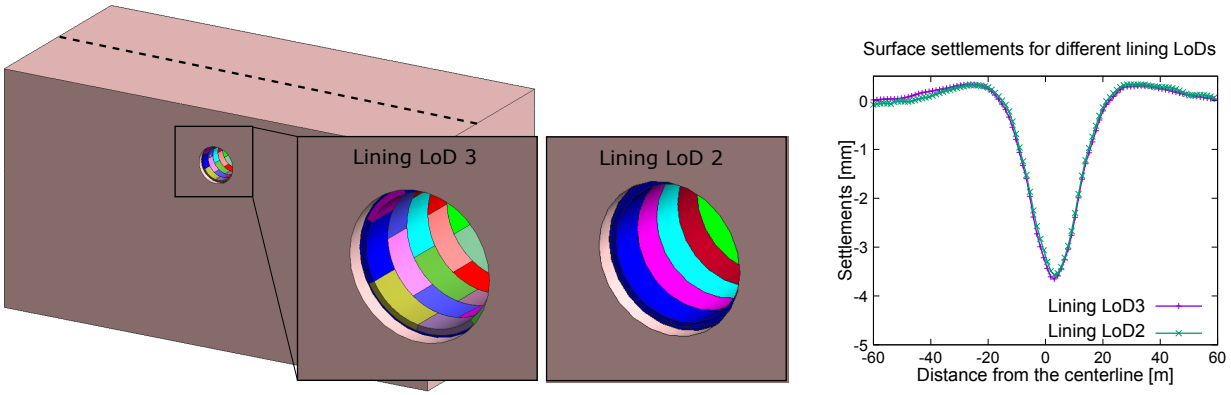


Figure 52: The choice of the LoD of the lining structure does not influence significantly the tunnelling induced settlements.

Using the SATBIM approach, we can analyse also much larger tunnel sections with high accuracy. This is shown in the example in Figure 53. In this example over 200 m long tunnel section is analysed using the highest levels of details for representation of the building and lining structure. That means, that we are able now to predict global response of the large systems in therms of building-structure interaction and induced surface settlements (see Figure 53a), as well as the very fine scale segment interactions as shown in see Figure 53b. Simulation like this are than naturally characterized with large number of DOFs, and high computational costs. However, solution for this challenge lays in high performance computing, and example for this is shown in the next Section 3.4.

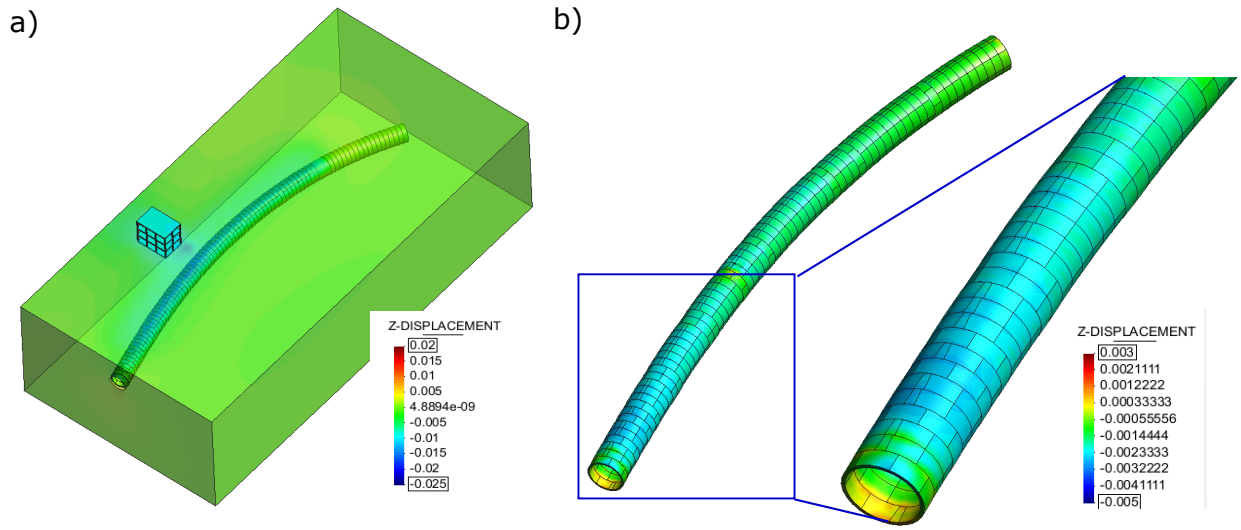


Figure 53: Large scale tunnel simulation model with highest LoD for building and lining. a) Vertical deformations of the compete model; b) Deformation of the segmental lining.

3.3.6 Testing different LoDs of the soil material models

In SATBIM project different possibilities are offered for the modelling of soil behaviour as explained in Section 3.2.2, including:

- Linear elastic (LE) – LoD1
- Drucker-Prager (DP) and Mohr Coulomb (MC) – LoD2
- CASM – LoD3

In this section TIM (see Section 2) is used to generate simulations of the tunnelling process, considering different complexity of lining structure and its installation (LoD 1, LoD 2 and LoD3), and testing surface settlement response w.r.t. different material models (LE, DP, MC and CASM).

Figure 54a shows the Revit model of the tunnel section used for testing diffrent soil material models. The soil is modelled on the medium level of detail (LoD 2), and the lining structure is varied form low to high complexity (LoD 1, LoD 2 and LoD 3). For medium and high representation of the lining structure, TBM is modelled on the LoD 2, while for the volume loss representation of the lining structure support, TBM is omitted as explained in component dependencies (see Table 31). Here the in this tunnelling case, 20 lining rings of 2.5 m lenght and 10. m excavation radius are excavated under 15 m overburden. Figure 54b shows the GiD geometry for the respective Revit information model.

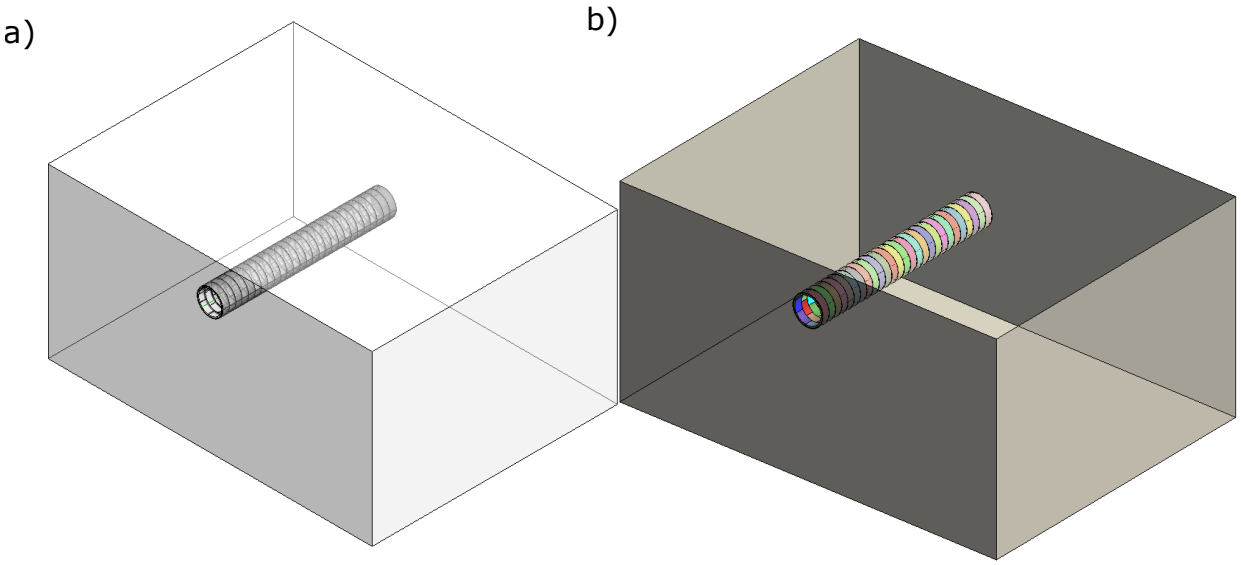


Figure 54: Tunnel information model and GiD geometry for the example of thesing soil material models: LE, DP, MC and CASM.

Based on this Revit model, a simulation model including FE mesh with predefined layers for belonging BCs (see Figure 55a) and simulation script defining the flow of simulation is generated using `SatBimModeller`. The FE simulation model is calculated using the FE

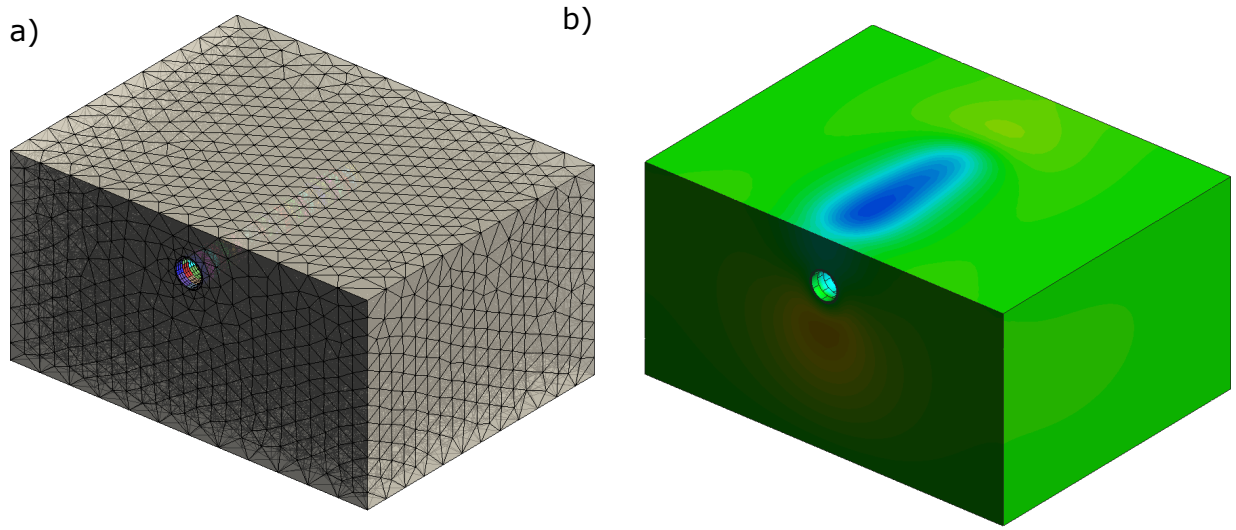


Figure 55: TFE mesh and results of the tunnelling induced surface settlements for the example of testing soil material models: LE, DP, MC and CASM.

framework for the high performance commuting KRATOS, and the typical tunnelling induced displacements are shown in Figure 55b.

Soil material parameters for soil material representation on LoD 1 and LoD 2 are shown in Table 1. Clearly, for the definition of the linear elastic representation of the soil only the portion of those parameters is needed (Young modulus, Poisson ratio, Density, Porosity and Permeability), while for the simple elasto-plastic constitutive models MC and DP the rest of parameters are used. On contrary, large number of parameters is needed for definition

Parameter	Value
Constitutive law	LE/MC/DP
Young modulus	20 MPa
Poisson ratio	0.25
Density	1732 kg/m^3
Porosity	0.4
Cohesion	200 kPa
Hardening modulus	5.83 MPa
Friction angle	30
Dilatancy angle	30
Permeability	0.001 m/s

Table 1: Material parameters for example 1 & 2

of the CASM . Those parameters are shown in the Table 2. It has to be noted that the the parameters for the soil models are not calibrated based on lab testing, however, for this example we used the parameters based on some project data, and recommendations. For the material models LE, MC and DP it is obvious that the crucial matrail parameters are

Parameters	Index	Value	Unit
Constitutive soil behavior — CASM			
Density	ρ	1732	[kg/m^3]
Poisson's ratio	ν	0.25	[-]
Slope of the unload/reload curve in the (v — ln p') space	κ	0.0025	[-]
Slope of the normal compression curve in the (v — ln p') space	λ	0.01	[-]
Spacing ratio	r	0.2	[-]
Shape parameter of the yield surface	n	2.0	[-]
Slope of the critical state line under triaxial compression	M	1.08	[-]
Initial preconsolidation mean stress for saturated soil	P_0	10^{12}	[kN/m^2]
Porosity	n	0.2	[-]
Permeability water	k_w	4.4 10^{-3}	[m/s]

Table 2: Material parameters used for the CASM

shared among all constitutive modles, however, for the CASM description, new parameters are introduced (e.g. Slope of the unload/reload curve κ and Slope of the normal compression curve λ) which are not compatible with other constitutive models. In real world application, those parameters would have to be calibrated with lab material testing.

As already meditation, the in the first test example, the lining ism modelled using simplified representation - volume loss method (see Section 3.2.1) The volume loss that exhibits after the soil deconfinment due to soil excavation and instalation of the linig structure is addopted to be $vl = 0.5\%$. The effect of the choice of the soil constitutive behaviour description on the tunnelling-induced settlements trough is shown in Figure 56. As expected, the LE will show the smallest induced settlements compared to LE and DP, and especially CASM where the settlements are largest due to plastification of the soil.

In the second example, a higher level of definition (LoD 2) for the representation of the tunnel lining structure and the TBM is selected. This model accounts for the shield as a deformable body moving through the soil and interacting with the ground through surface-to-surface contact. The tunnel advance is modeled by means of deactivation of soil elements and installation of the tunnel lining and grouting elements. The annular gap between the segmented lining tube and the excavation boundary is refilled with grout, modelled as a fully saturated two-phase material, considering the time-dependent evolution of stiffness and permeability. Tunnelling-induced deformations are controlled by applying the face support pressure and the grouting pressure at the tunnel face and in the steering gap, respectively. The tunnelling-induced settlements trough are highly dependent of the choice of the constitutive modelling of the soil as shown in Figure 57. Of course, the soil response we are obtaining in this study is influenced by the more accurate representation of the soil support with the face and support grouting pressures and the support by means of the lining structure, and therefore this response slightly differs from the one shown in Figure 56.

Finally, if the highest accuracy for the representation of the ling structure installation is adopted in the simulation model, and lining rings are represented as group of segments

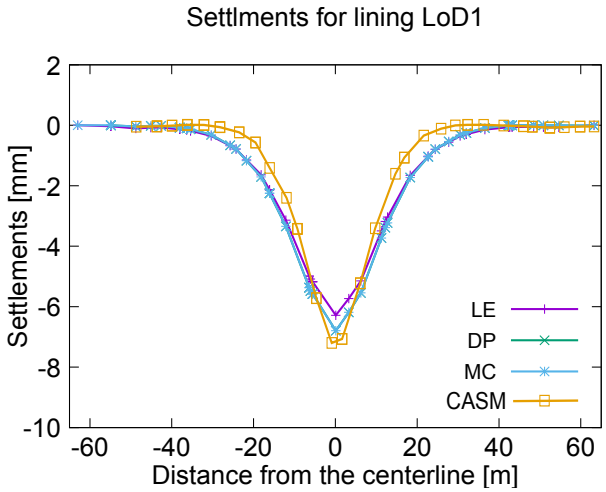


Figure 56: Tunnelling induced surface settlements trough for soil material models: LE, DP, MC and CASM. Lining is modelled using volume loss method.

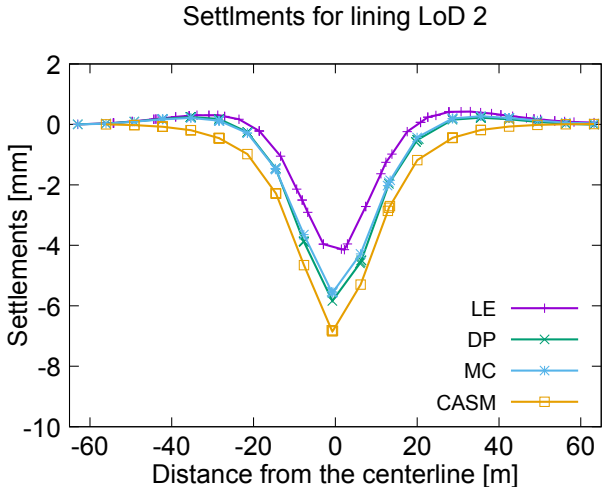


Figure 57: Tunnelling induced surface settlements trough for soil material models: LE, DP, MC and CASM. Lining is model as a solid structure with the ring-wise construction.

interacting trough normal contact interface and being connected with steel bolts, the dependency of the tunnelling induced displacements on the soil-matrial behaviour is shown in Figure58.

The sensitivity of the tunnelling-induced settlements w.r.t. CASM parameter the slope of the unload/reload curve in the κ is shown in Figure 59. As it can be noted, the tunnelling induced displacements are highly dependent on this parameter, and therefore is recommended to be very careful when adopting this parameter for the numerical simulation because the soil response will be mainly driven by it.

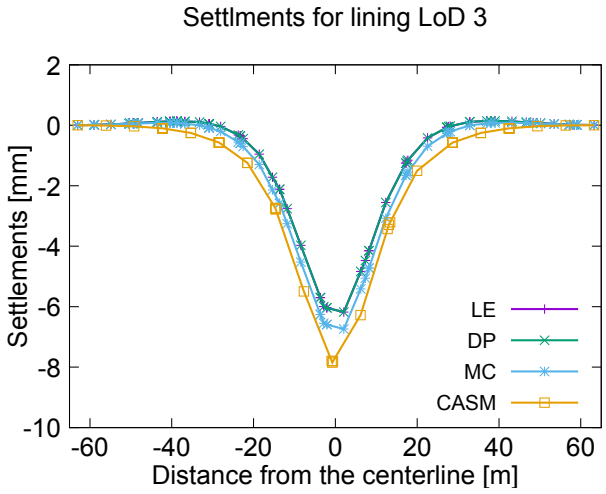


Figure 58: Tunnelling induced surface settlements trough for soil material models: LE, DP, MC and CASM. Lining is model as a segmented structure with interfaces between segments with the segment-wise installation.

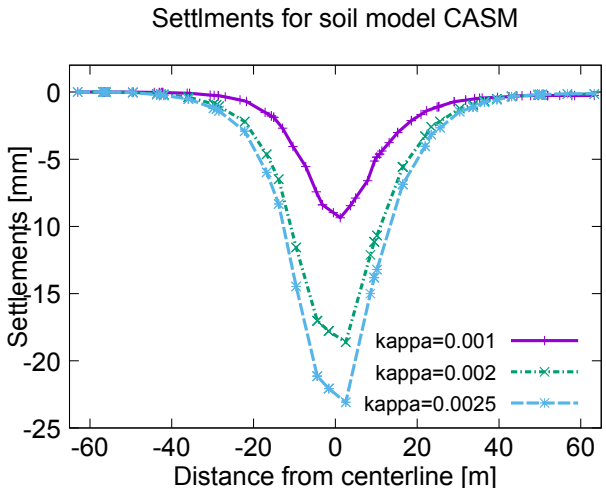


Figure 59: Tunnelling induced surface settlements trough different parameters κ of the CASM soil model using lining representation on LoD 2

3.4 Parallel strategies for large scale tunnel simulation

In this work, a parallelization strategy is applied to perform large scale simulations of the advancement process in mechanized tunneling. To support the generation of realistic simulation models, the Building Information Modeling (BIM) concept is employed. In this approach, different simulation components in terms of geometrical representation, material modeling and process modeling are stored in the database, and can be selected on demand, depending on the objective of the analysis. In the highest modeling level, all components

involved in the TBM advancement in urban environments, namely the soil, the tail-void grouting, the segmented lining, and the existing buildings are considered.

3.4.1 Governing equations and finite element discretization

In this section, the governing equations describing the underlying physics of the particular problem at hand is briefly presented. The emphasis is given to additional terms representing the interaction between the buildings, the TBM and the soils. The strong form of the governing equation for soil is well described in Nagel and Meschke (2010), hence only the weak, discretized form of the governing equations is discussed. Following that, the variation of the total energy of the system is characterized by

$$\delta W = \delta W^s + \delta W^w + \delta W^c + \delta W^t \tag{35}$$

The reader is referred to Kasper (2005); Nagel (2009); Nagel and Meschke (2010) for the details description of the spatial discretization of system equilibrium δW^s, the fluid-mass balance equation δW^w and Bui and Meschke (2017) for a short overview of the discretization for the virtual work δW^c of the contact force. In analyses at higher level of detail, contact conditions are considered along the interface between the moving TBM and the surrounding soil. The interaction between the buildings and the soil is represented by mechanical tying of nonconforming domains. Using the penalty method, the soil-building interaction is considered by the contribution δW^t to the total virtual work of the system

$$\delta W^t = \int_{\Gamma^t} \rho_t (\delta \mathbf{u}^{(1)} - \delta \mathbf{u}^{(2)}) \cdot (\mathbf{u}^{(1)} - \mathbf{u}^{(2)}) dV \tag{36}$$

After spatial discretization by means of finite elements, the expression (35) constitutes the nonlinear weak form in terms of two sets of primal nodal variables, the displacement degree of freedom (d.o.f) and water pressure d.o.f. After linearization, one obtains straightforwardly an equation system in the format

$$\begin{bmatrix} \Delta u^s \\ \Delta p^w \end{bmatrix}_{n+1} = \begin{bmatrix} \mathbf{A}_{ss} & \mathbf{A}_{sw} \\ \mathbf{A}_{ws} & \mathbf{A}_{ww} \end{bmatrix}^{-1} \begin{bmatrix} \mathbf{R}^s_{ext} - \mathbf{R}^s_{int} \\ \mathbf{R}^w_{ext} - \mathbf{R}^w_{int} \end{bmatrix}, \tag{37}$$

where R^s_{ext} - R^s_{int} (R^w_{ext} - R^w_{int}) denote the residuals between the external and internal force vectors related to the mechanical equilibrium (index s) and the mass balance of the groundwater (index w). It is noted that, to make the discretization in Equation (35) LBB-compatible, the water pressure degree of freedom (d.o.f) is approximated using a functional space of lower order than of displacement d.o.f. In the numerical example presented in 3.4.3, tetrahedral finite elements with 10 nodes for displacements and 4 nodes for the pressure is employed.

3.4.2 Computation and parallelisation strategy

Application of a domain decomposition strategy for tunnel advancement simulations must account for the fact, that the contact surface, i.e. the interface between the TBM and the soil, is continuously changing to adapt with the new position of the TBM. The contact surfaces

are identified from a contact search algorithm based on bounding volume hierarchy (BVH). In typical scenarios, when the TBM passes through the mesh boundary between computation domains, the contact search algorithm may fail since some of the master segments do not belong to the current working process. To cope with this issue, the master segments are distributed to all domains as ghost segments (see Figure 60 for an illustration of the ghost master segments distributed to the domain containing the TBM). It imposes additional memory overhead to store those ghost segments in the memory. However, this is negligible compared to the number of elements and boundary conditions of the complete system. On the rest of the domain, the system is decomposed by using the multilevel k-way partitioning algorithm from Metis package Karypis and Kumar (1998).

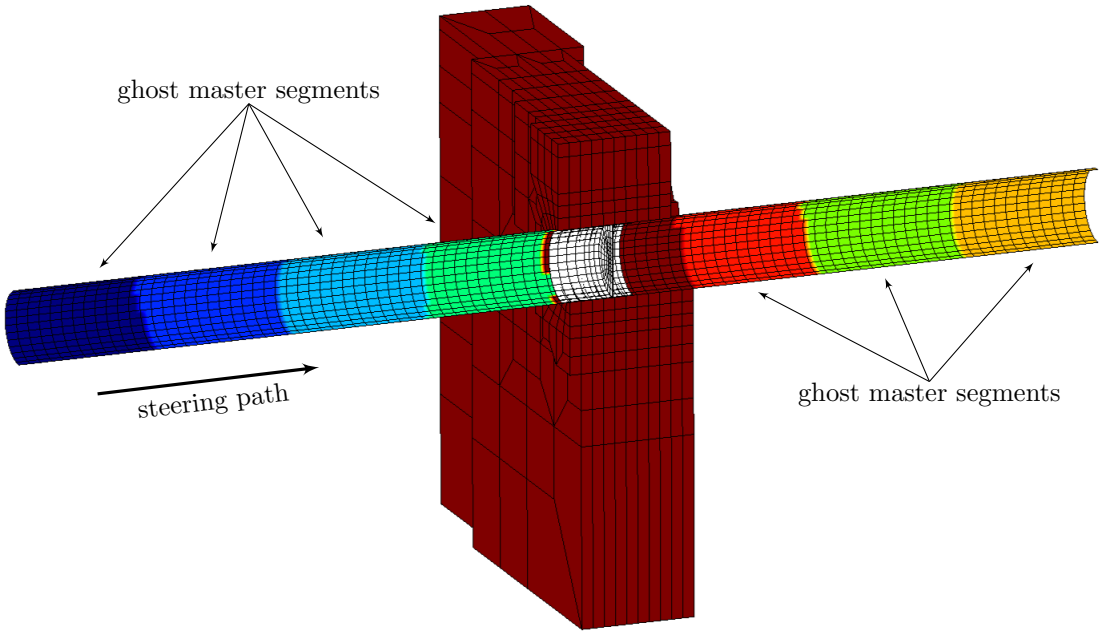


Figure 60: Ghost master segments in TBM partition

To solve the block system Equation (37) effectively using an iterative solver, the block preconditioner of the form

$$P^{-1} = \begin{bmatrix} I & 0 \\ 0 & -S^{-1} \end{bmatrix} \begin{bmatrix} I & 0 \\ -A_{ws} & I \end{bmatrix} \begin{bmatrix} A_{ss}^{-1} & 0 \\ 0 & I \end{bmatrix} \tag{38}$$

is employed. The Schur preconditioner S is approximated by $S = A_{ww} - A_{ws} diag(A_{ss}^{-1}) A_{sw}$ to retain the sparsity. It is computed more efficiently as compared to the dense counterpart. The quality of this block preconditioner depends substantially on the quality of the sub-preconditioners, which are used to invert the sub-block A^{-1} and S^{-1}. In the numerical example, the multigrid preconditioner Falgout and Yang (2002) is chosen as sub-preconditioners. The setting for the multigrid preconditioner is chosen by manual calibrating and comparing the number of required iterations per each solution step.

The discretization of multiphase system involving intricate interactions between various components, i.e. the ground, the tail-void grouting, the lining and the hydraulic jacks leads to a

challenging linear algebra problem. The employed linear solver is required to solve the resulting linear system in a reasonable time, and with finite amount of computer memory. To solve a big system efficiently, the iterative solver is chosen, with appropriate block preconditioner to speed-up the convergence and to improve numerical accuracy. The large tunnel model is decomposed into sub-domains using domain decomposition techniques. The sub-domains are stored and computed locally on separate computing node. This approach virtually enables the simulation of tunnel model of arbitrary size.

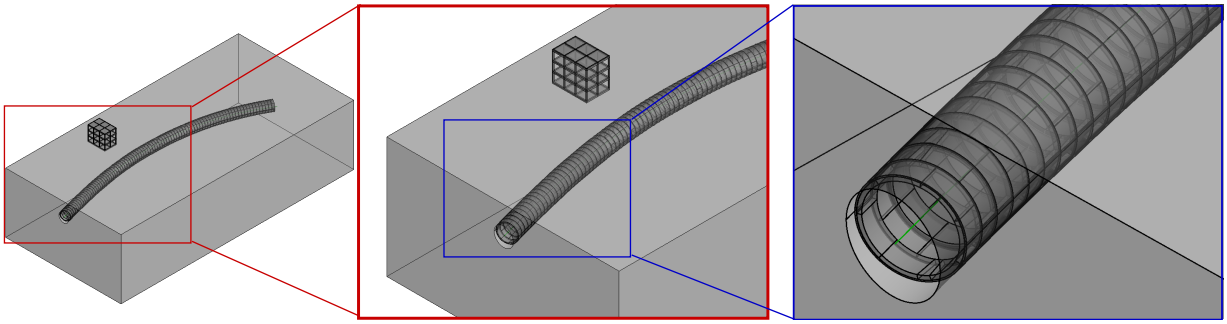


Figure 61: Tunnel information model for more than 200m long tunnel section used for generation of large scale tunnel simulation

3.4.3 Numerical example for panellisation of the simulation software

TIM described in Section 2 is used for generation on large tunnel sections combining components on different LoDs, as shown in Figure 61. Based on this Revit model, a simulation model is generated using `SatBim Modeller`. This simulation model (see Figure 62a) is characterized with large number of DOFs (over $1,000,000$). Such large scale simulation is very difficult to solve using standard non-iterative solvers like PARDISO. Therefore, a panellisation strategies developed by Bui et al. (2013) are applied to minimise the commutation time. The decomposition of the simulation model into sub-domains using domain decomposition techniques is shown in Figure 62b.

Figure 63 shows the calculation time for the large scale simulation model shown in Figure 62a, using different solvers and cluster set-ups. Here The non-iterative solver PARDISO is compared with iterative solver GMRES, and is shown that for the same cluster setup (8 threads) the calculation time is reduced for more than eight times, while increasing number of threads the calculation time can be reduced for more than 12 times. The Blue line shows how the iterative solver speeds-up when increasing number of threads. From this it can be seen that increasing the number of threads over 32 wil lead to slow-down.

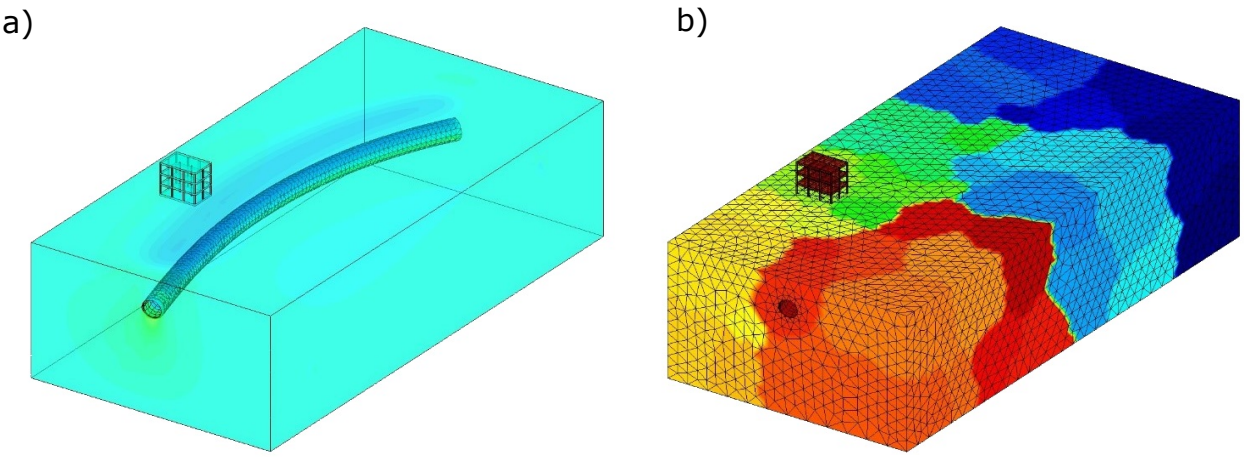


Figure 62: a) Simulation of tunnel advancement with building on top; b) sub-domains decomposition using in the simulation.

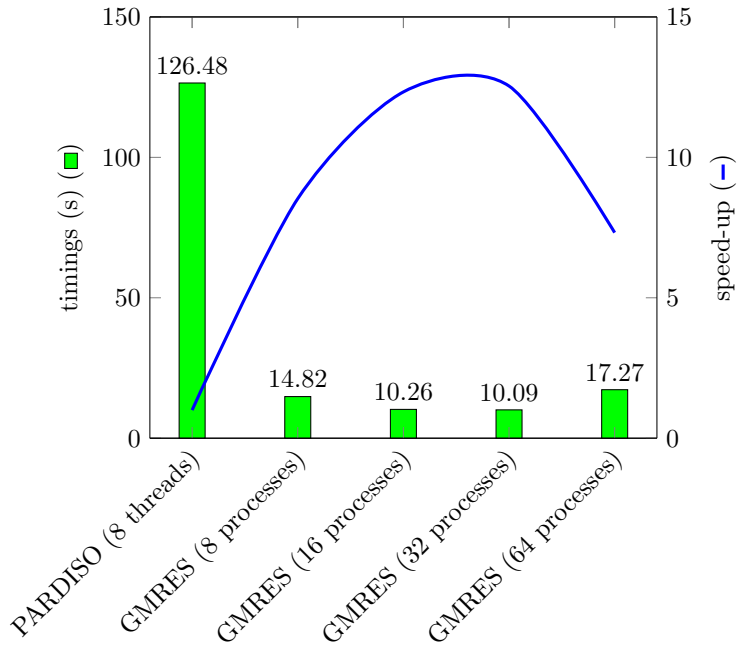


Figure 63: Calculation time and sped up using different solvers and cluster setups

4 Visualisation of Numerical analysis results

The visualisation of "raw" numerical analysis output (e.g., simulation results stored in each node or Gauss point of the FE model) might not be efficient in order to draw conclusions about phenomena occurring during tunnel construction.

Therefore, these results are processed by a user-defined evaluation algorithm. As an example, an algorithm can be applied that searches and processes the stresses in buildings and returns risk of damage of structures due to tunnelling in a simplified way. To enable this, a new visualisation method was developed within the SATBIM project. With this approach, the results of the numerical simulations (e.g., settlements) and their impact on the existing environment (e.g., structural forces, risk of damage) can be visualised within the multi-level TIM to enable comprehensive, intuitive, and quick understanding of effects of design actions on the stability and safety of the existing environment or the tunnel itself.

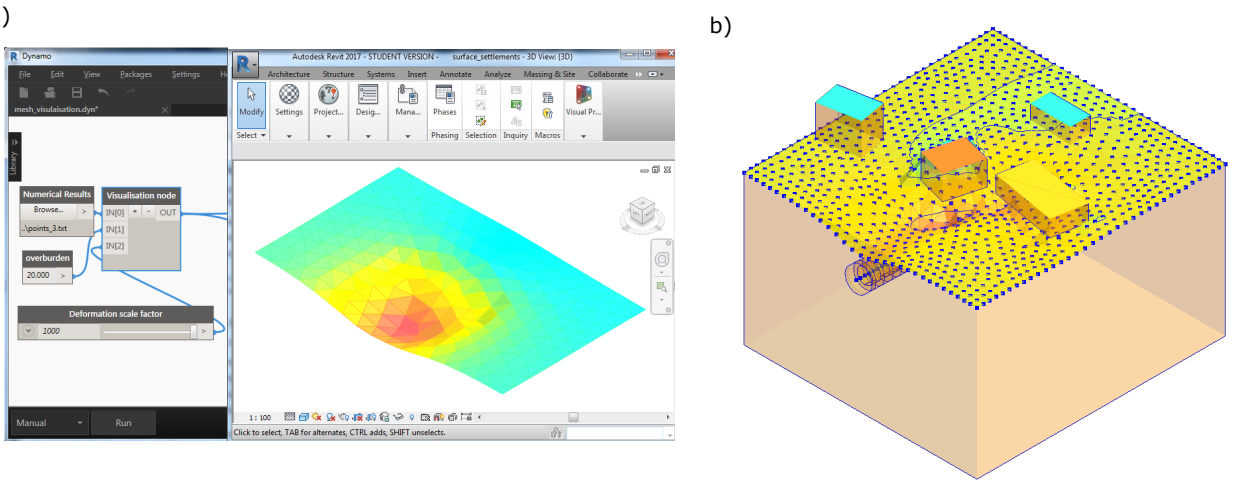


Figure 64: Visualisation of numerical results in the design software Revit, i.e., tunnelling induced effects: a) surface settlements; b) effect on buildings.

To enable visualisation of simulation results as described before, a visualisation node has been implemented in Dynamo. This node reads customised simulation results and displays them in Revit (see Fig. 64a). Figure 64a shows the visualisation of the surface settlements induced by tunnelling in Revit, highlighting both the contour-fill coloured surface as well as the deformation of the surface (which was made more visible using a deformation scale factor). This simple and intuitive representation is crucial for improving the understanding of tunnelling-induced effects by non-experts, who may be involved in the decision process of the project development. Besides surface settlements, the risk on existing infrastructure is also visualised as shown in Figure 64b, where the buildings are sorted in relative scale from green (safe) to red (in risk) based on the tunnel excavation influence.

Alternatively, if the user needs to access the detailed results of the analysis, this data is stored in result files and can be visualised in the post-processing module of GiD. Then, results such as displacements, stresses, or internal variables can be easily illustrated. Also, the construction sequence (TBM moving forward) can be animated.

Listing 10: Writing the surface settlements results to an output file

```
ifile = open('settlments.dat', 'a' )
for node_id in surface_nodes_print:
        if node_id!=0:
                node=model1.model_part.Nodes[node_id]
                x = node.GetSolutionStepValue(DISPLACEMENT_X)
                y = node.GetSolutionStepValue(DISPLACEMENT_Y)
                z = node.GetSolutionStepValue(DISPLACEMENT_Z)
                ifile.write(str(node_id)+' '+str(node.X)+' '+str(node.Y)+' '+str(node.Z)
                +' '+str(x)+' '+str(y)+' '+str(z)+'\n')
ifile.close()
```

4.1 Custom output of simulation results

Results of numerical simulations are exported as a "raw" numerical analysis output (e.g., simulation results stored in each node or Gauss point of the FE model) by defining the set of nodes and giving it as an argument to *Output Utility* . The output file contains selected information about the model in the observation points and is exported in *.tex* files. An example of writing the custom simulation results, which are later on imported by the Visualisation node is given in the following algorithm:

4.2 Implementation in Dynamo

A visualisation node has been implemented in Dynamo to enable visualisation of simulation results as described before. As shown in Figure 65, in the first part the custom simulation results are imported, and based on those results the deformed surface is generated. Deformation of the surface is emphasized by using the a deformation scale factor. In the second part the surface is coloured by the e contour-fill node.

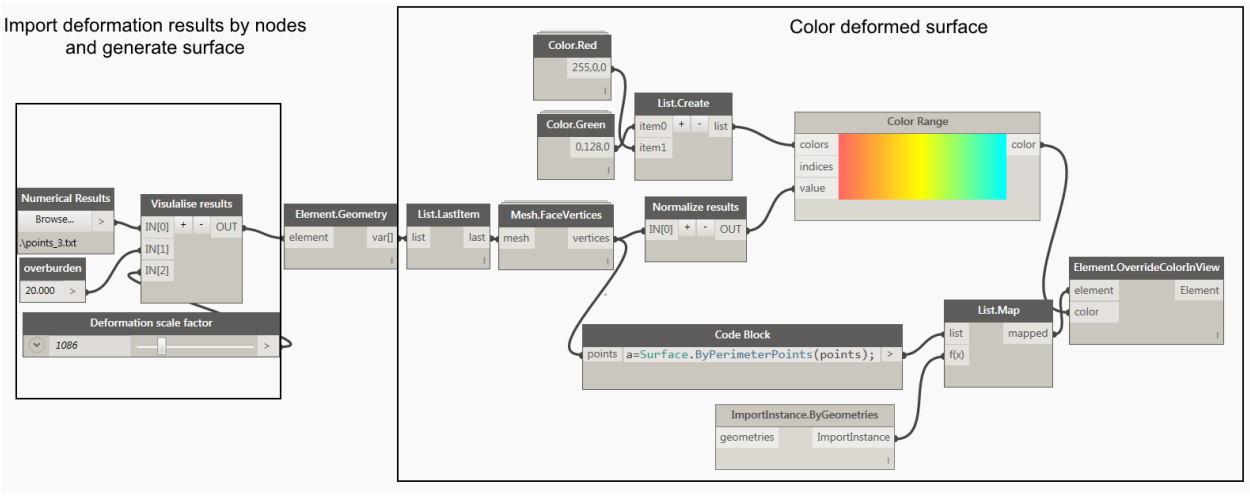


Figure 65: Visualisation node in Dynamo.

A simple python script for generation of the deformed surface based on simulation results of the displacements of surface nodes is given in Listing 11.

Listing 11: Generation of deformed coloured surface

```
from math import *
params = {}
file_name= IN[0]
overburd=IN[1]
scale=IN[2]
file = open( file_name, 'r' )

lines = file.readlines()
file_len = len(lines)
print(file_len)
points=[]
for i in range (0,file_len):
        line= lines[a]
        columns = line.split()
        x=float(columns[1])*100.0 #scale from m to cm
        y=float(columns[2])*100.0 #scale from m to cm
        z=float(float(columns[3])-overburd)*100.0*scale+overburd*100 #scale from m to cm & scale factor
        point=Point.ByCoordinates(x,y,z)
        points.append(point)
surface=Revit.Elements.Topography.ByPoints(points)
OUT=surface
```

4.3 Multi-user multi-touch video wall

The design and optimization can be performed on Multi-user multi-touch video wall to support design, collaboration and decision making in engineering (see Figure 66). Large-scale video walls and interactive multi-touch displays (e.g. PQ Labs, TUIO) are important for establishing the link to scientific and interactive visualisation of numerical simulation results.

Figure 66: Investigation of design alternatives on multi-user multi-touch video wall .

5 Meta model for real time prediction

5.1 The concept of using meta models for real-time assessment of design

In order to apply numerical simulations as a tool for real-time design optimisation, we can substitute numerical simulations with computationally cheap meta models. To enable real-time predictions of tunnelling-induced surface settlements, meta models are trained a priori from the process-oriented simulation model based on the multi-level TIM. For the generation of meta models, i.e., learning of numerical results, Artificial Neural Networks (ANN) (Ninić et al., 2011; Ninić and Meschke, 2015) are applied here.

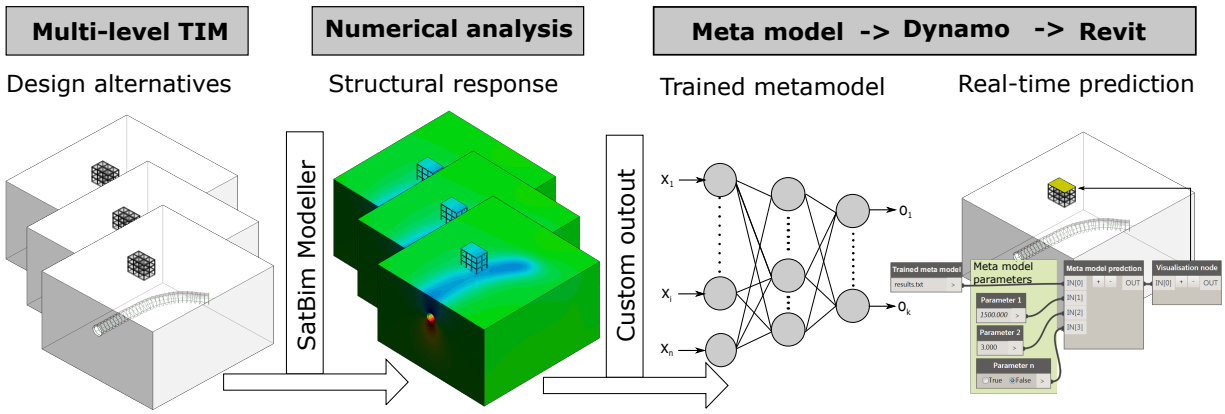


Figure 67: Scheme of the real-time prediction for design optimisation within the TIM.

Figure 67 illustrates a way of using the simulation-based meta models for real-time optimisation of the design. Based on different design alternatives of the tunnel project, characterised with particular design parameters, simulation models are generated and executed using `SatBim Modeller`. The simulation results are stored in a format suitable for meta model training.

Combining the already implemented visualisation nodes (see Section 4) with the meta model prediction, the real-time assessment of design alternatives is enabled. It has to be noted that the meta models are able to predict structural response not only for the trained set of parameters, but also interpolate between the defined ranges, and therefore allow for fine-grained assessment of the design.

5.2 Meta models for robust data training

For the purpose of real-time predictions of tunnelling-induced effects such surface settlements, risk on building damage, etc., a meta model is employed to substitute computationally demanding 3D numerical simulations. In a SATBIM project an algorithm is developed to select an optimal meta model based on different methods for data training:

- Linear Regression (LR),

- Polynomial Regression (PR),
- Support Vector Regression with RBF kernel (SVR-RBF),
- Support Vector Regression Polynomial kernel (SVR-Poly),
- Artificial Neural Networks (ANNs).

In all training methods the simulation model parameters i are taken as input variables for the meta model training while the tunnel-induced effects (settlements, risk of damage, etc.) are target values. Meta models are than trained to predict the output for given input values minimising the error between the target and output values. Some of the mentioned machine learning methods are characterized with parameters which influence the tarring performance. In order to have the robust meta modes, those parameters are optimized with Particle Swarm optimization method Kennedy and Eberhart (1995).

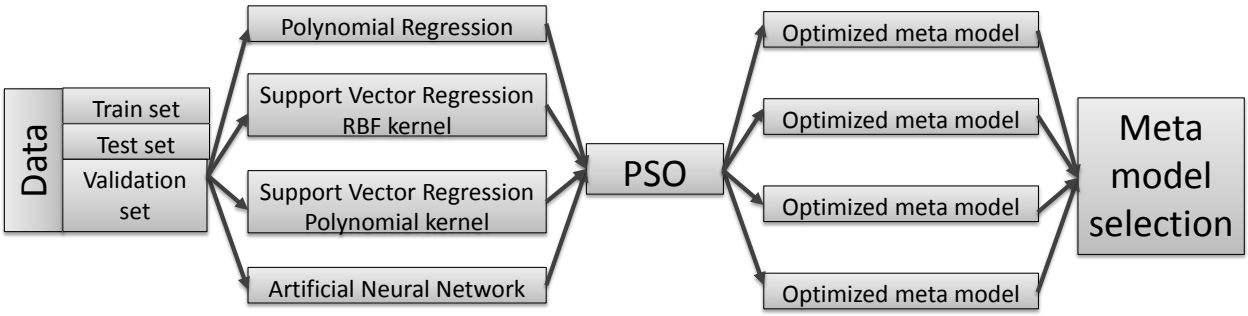


Figure 68: Selection of robust meta models by using Particle swarm optimization for selecting the optimal training parameters

Considering the different dimensions of different parameters, all input-output pairs are mapped to the interval $(0.1, 0.9)$ using a *Data Normalization* algorithm. For a parameter V, the normalized value V_{norm} is obtained from

$$V_{\text{norm}} = \frac{V - V_{min}}{V_{max} - V_{min}}(\bar{V}_{\text{max}} - \bar{V}_{\text{min}}) + \bar{V}_{\text{min}} \tag{39}$$

V_{max} and V_{min} are the maximal and minimal value of the variable V, and $\bar{V}_{\text{max}}$ and $\bar{V}_{\text{min}}$ are the maximal and minimal values of the variable V after normalization, defined as 0.1 and 0.9.

After the complete data set has been processed, the data is split into data for training, testing and validation of the meta model . The data is sent for training, testing and validation to the robust machine learning algorithm for optimized meta models, and after the training is completed the optimal meta model is selected (see Figure 68).

The results of the performance of trained, tested and validated meta model are shown in Figure 69. In following subsection the description of machine learning methods and optimization for the robust meta model training is presented.

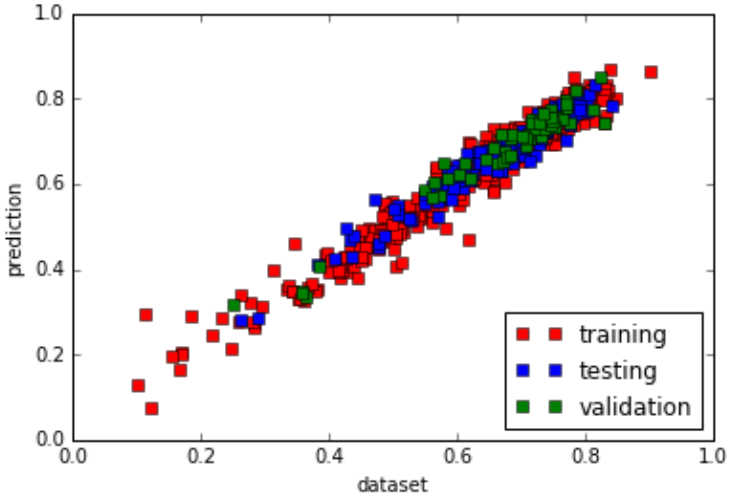


Figure 69: Training and testing performance of the optimal meta model for given data set.

5.2.1 Linear Regression and Polynomial Regression

Linear regression is an approach for modelling the relationship between a scalar dependent variable y (output or target variable) and one or more independent variables x (in our case input vector).

For given a data set $\{y_i, x_{i1}, \ldots, x_{ip}\}_{i=1}^{n}$ of "n" patterns, a linear regression model assumes that the relationship between the dependent variable y_i and the "**p**" vector of input variables x_i is linear function. This relationship is modelled through an "error variable" ε_i an unobserved random variable that adds noise to the linear relationship between the dependent variable and inputs. Thus the model takes the form:

$$y_i = \beta_1 x_{i1} + \cdots + \beta_p x_{ip} + \varepsilon_i = \mathbf{x}_i^{\mathrm{T}} \boldsymbol{\beta} + \varepsilon_i, \qquad i = 1, \ldots, n, \tag{40}$$

Often these "n" equations are stacked together and written in vector form as

$$\mathbf{y} = \mathbf{X}\boldsymbol{\beta} + \boldsymbol{\varepsilon}, \tag{41}$$

where

$$\mathbf{y} = \begin{pmatrix} y_1 & y_2 & \cdots & y_n \end{pmatrix}^T, \tag{42}$$

$$\mathbf{X} = \begin{pmatrix} \mathbf{x}_1^{\mathrm{T}} \\ \mathbf{x}_2^{\mathrm{T}} \\ \vdots \\ \mathbf{x}_n^{\mathrm{T}} \end{pmatrix} = \begin{pmatrix} x_{11} & \cdots & x_{1p} \\ x_{21} & \cdots & x_{2p} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{np} \end{pmatrix}, \qquad \boldsymbol{\beta} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{pmatrix}, \quad \boldsymbol{\varepsilon} = \begin{pmatrix} \varepsilon_1 \\ \varepsilon_2 \\ \vdots \\ \varepsilon_n \end{pmatrix}. \tag{43}$$

Linear Regression fits a linear model i.e. or regression coefficients $\boldsymbol{\beta}$ to minimize the residual sum of squares between the observed responses in the dataset, and the responses predicted

by the linear approximation. Mathematically it solves a problem of the form:

$$min\frac{1}{2}\left\|\mathbf{X}^T\boldsymbol{\beta}+\boldsymbol{\varepsilon}-\mathbf{y}\right\|^2 \tag{44}$$

Ordinary least squares (OLS) is the simplest and thus most common estimation method. It is conceptually simple and computationally straightforward. OLS estimates are commonly used to analyse both experimental and observational data.

The OLS method minimizes the sum of squared Errors and residuals in statistics, and leads to a closed-form expression for the estimated value of the unknown parameter β:

$$\hat{\boldsymbol{\beta}}=(\mathbf{X}^{\mathrm{T}}\mathbf{X})^{-1}\mathbf{X}^{\mathrm{T}}\mathbf{y}=\left(\sum\mathbf{x}_i\mathbf{x}_i^{\mathrm{T}}\right)^{-1}\left(\sum\mathbf{x}_iy_i\right). \tag{45}$$

Polynomial regression is a form of linear regression in which the relationship between the independent variable $\mathbf{x}$ and the dependent variable y is modelled as an nth degree polynomial in x.

In general form the he polynomial regression model can be formulates as:

$$y_i\ =\ \beta_0+\beta_1x_i+\beta_2x_i^2+\cdots+\beta_mx_i^m+\varepsilon_i\ (i=1,2,\ldots,n) \tag{46}$$

It also can be expressed in matrix form in terms of a design matrix $\mathbf{X}$, a response vector$\vec{y}$, a parameter vector $\vec{\beta}$, and a vector $\vec{\varepsilon}$ of random errors. The i-th row of $\mathbf{X}$ and $\vec{y}$ will contain the x and y value for the i-th data sample. Then the model can be written as a system of linear equations:

$$\begin{bmatrix}y_1\\y_2\\y_3\\\vdots\\y_n\end{bmatrix}=\begin{bmatrix}1&x_1&x_1^2&\ldots&x_1^m\\1&x_2&x_2^2&\ldots&x_2^m\\1&x_3&x_3^2&\ldots&x_3^m\\\vdots&\vdots&\vdots&&\vdots\\1&x_n&x_n^2&\ldots&x_n^m\end{bmatrix}\begin{bmatrix}\beta_0\\\beta_1\\\beta_2\\\vdots\\\beta_m\end{bmatrix}+\begin{bmatrix}\varepsilon_1\\\varepsilon_2\\\varepsilon_3\\\vdots\\\varepsilon_n\end{bmatrix}, \tag{47}$$

which when using pure matrix notation is written as:

$$\vec{y}=\mathbf{X}\vec{\beta}+\vec{\varepsilon}. \tag{48}$$

In the applications in the implementations we are using second-order polynomials, so that the model looks like this:

$$y_i(\beta,x)\ =\ \beta_0+\sum_{i=1}^{n}\beta_ix_i+\sum_{i=1}^{n}\beta_{ii}x_i^2+\sum_{i=1}^{n}\sum_{j>i}^{n}\beta_{ij}x_ix_j=\mathbf{X}^T\boldsymbol{\beta} \tag{49}$$

$$\mathbf{X}^T=\left(1,x_1,x_2,\ldots,x_n,x_1^2,x_2^2,\ldots,x_n^2,x_1x_2,x_1x_3,\ldots x_{n-1}x_n\right)^T \tag{50}$$

$$\mathbf{X}^T=\left(1,\beta_1,\beta_2,\ldots,\beta_s,\beta_{11},\beta_{22},\ldots,\beta_{nn},\beta_{12},\beta_{13},\ldots\beta_{n-1,n}\right)^T \tag{51}$$

The vector of estimated polynomial regression coefficients (using OLS estimation) is

$$\widehat{\vec{\beta}} = (\mathbf{X}^T\mathbf{X})^{-1}\,\mathbf{X}^T\vec{y}. \tag{52}$$

This is the unique least-squares solution as long as $\mathbf{X}$ has linearly independent columns. Since $\mathbf{X}$ is a VANDERMONDE MATRIX, this is guaranteed to hold provided that at least $m+1$ of the x_i are distinct (for which $m < n$ is a necessary condition).

5.2.2 Support Vector Regression

The support vector machine theory is described in VAPNIK (2000). In SVR method, the support vector determines the approximation function. The SVR technique search the multivariate regression function $f(x)$ based on the input data set, i.e., the training set x to predict the output data.

$$f(x) = \sum_{i=1}^{n}(\alpha_i - \alpha_i^*)K(x, x_i) + b \tag{53}$$

where K is a kernel, n is the number of training data, b is an offset parameter of the model and α, α^* are Lagrange multipliers of the primal-dual formulation of the problem. The vector of the training data set is written as:

$$\mathbf{X} = \{(x_{ij}, y_i, \qquad i = 1, \ldots, n)\} \tag{54}$$

Where x_{ij} is the jth input vector of the ith pattern (input sample), while y_i is the target (output) value. The fitness function or the approximation model is considered good if the output of SVR regression is quite similar to the required output vector y_i . The kernel K represents an inner product of the kernel function ϕ . Polynomials, splines, radial basis are examples of kernel functions. The kernel is, in general, a non-linear mapping from an input space onto a characteristic space formulated as:

$$K(x, x_i) = \langle \phi(x) \cdot \phi(x_i) \rangle \tag{55}$$

The kernel functions ϕ used in the application below can be any of the following:

- linear: $\langle x, x' \rangle$.
- polynomial:$(\gamma\langle x, x' \rangle + r)^d$. d is specified by keyword degree, r by coef0.
- rbf: $\exp(-\gamma|x - x'|^2)$. γ is specified by keyword gamma, must be greater than 0.
- sigmoid $(\tanh(\gamma\langle x, x' \rangle + r))$, where r is specified by coef0.

Given training vectors $x_i \in \mathbb{R}^p$, i=1,..., n, and a vector $y \in \mathbb{R}^n \varepsilon$-SVR solves the following primal problem:

$$\min_{w,b,\zeta,\zeta^*} \frac{1}{2}w^T w + C\sum_{i=1}^{n}(\zeta_i + \zeta_i^*) \tag{56}$$

$$\begin{aligned} \text{subject to} \quad & y_i - w^T\phi(x_i) - b \leq \varepsilon + \zeta_i, \\ & w^T\phi(x_i) + b - y_i \leq \varepsilon + \zeta_i^*, \\ & \zeta_i, \zeta_i^* \geq 0, i = 1, ..., n \end{aligned} \tag{57}$$

Its dual is

$$\min_{\alpha,\alpha^*} \frac{1}{2}(\alpha - \alpha^*)^T Q(\alpha - \alpha^*) + \varepsilon e^T(\alpha + \alpha^*) - y^T(\alpha - \alpha^*) \tag{58}$$

$$\begin{aligned} \text{subject to} \quad & e^T(\alpha - \alpha^*) = 0 \\ & 0 \leq \alpha_i, \alpha_i^* \leq C, i = 1, ..., n \end{aligned} \tag{59}$$

where e is the vector of all ones, $C > 0$is the upper bound, Q is an n by n positive semi-definite matrix, $Q_{ij} \equiv K(x_i, x_j) = \phi(x_i)^T\phi(x_j)$ is the kernel. α_i and α_i^* are the weights that are found minimizing the function. Here training vectors are implicitly mapped into a higher (maybe infinite) dimensional space by the function ϕ. The decision function is stated in 53.

The transformed regression function of SVR, based on support vector, can be reformulated as:

$$f(x) = \sum_{x_i \in SV} (\alpha_i - \alpha_i^*)K(x, x_i) + b \tag{60}$$

where SV is the support vector set. The transformed regression problem may be solved, for example, by quadratic programming and only the input data corresponding to the non-zeros α_i and α_i^* contribute to the final regression model. The corresponding inputs are called support vectors. Graphical interpretation of the prediction algorithm using SVR with identified support vectors is shown in Fig. 70 .

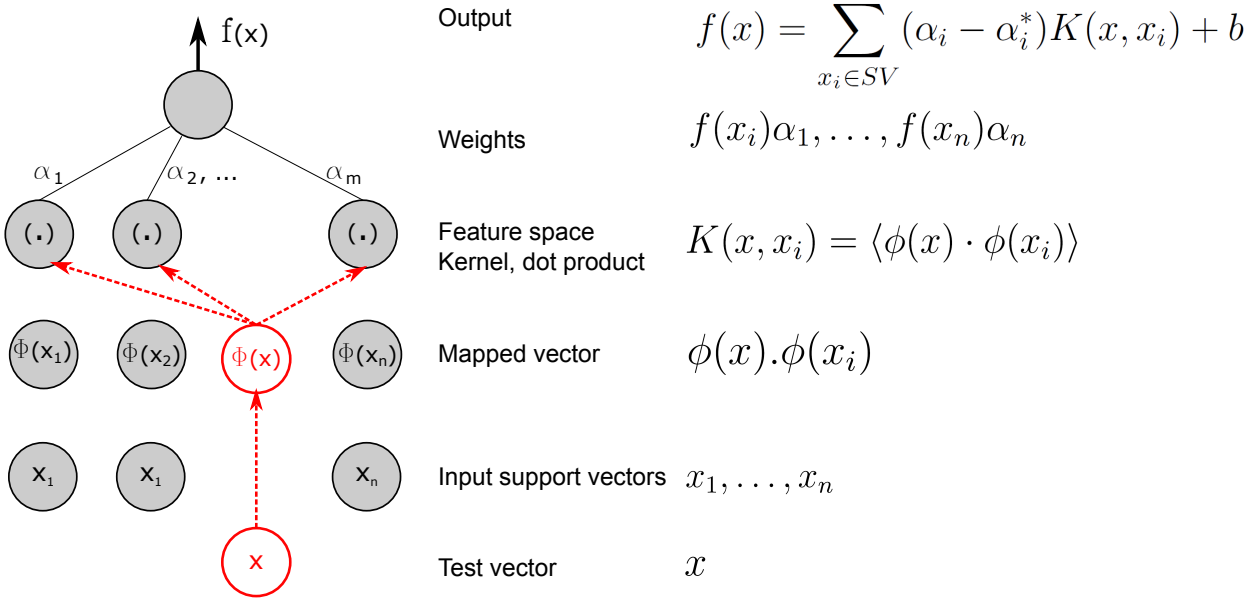


Figure 70: Graphical interpretation of the support vector regression model

5.2.3 Artificial Neural Networks

Artificial Neural Networks (ANNs), initially introduced by McCulloch and Pitts, are a form of artificial intelligence that attempts to mimic the behavior of the human brain and neural system, being capable of learning the pattern associated with a large body of data McCulloch and Pitts (1943). Following the initial proposal, numerous researchers gave a significant contribution to the development of this method and introduced new learning techniques Widrow and Hoff (1960); Rosenblatt (1962). The most popular ANN method, the Back Propagation Neural Network, with a typical architecture illustrated in Figure 5.2.3, was first proposed by Rumelhart, Hinton, and Williams Rumelhart et al. (1986). This method can be used to learn the weight values and yield values. During the learning process, the gradient descent method is used to change the weight values and yield values as quickly as possible to reduce the error.

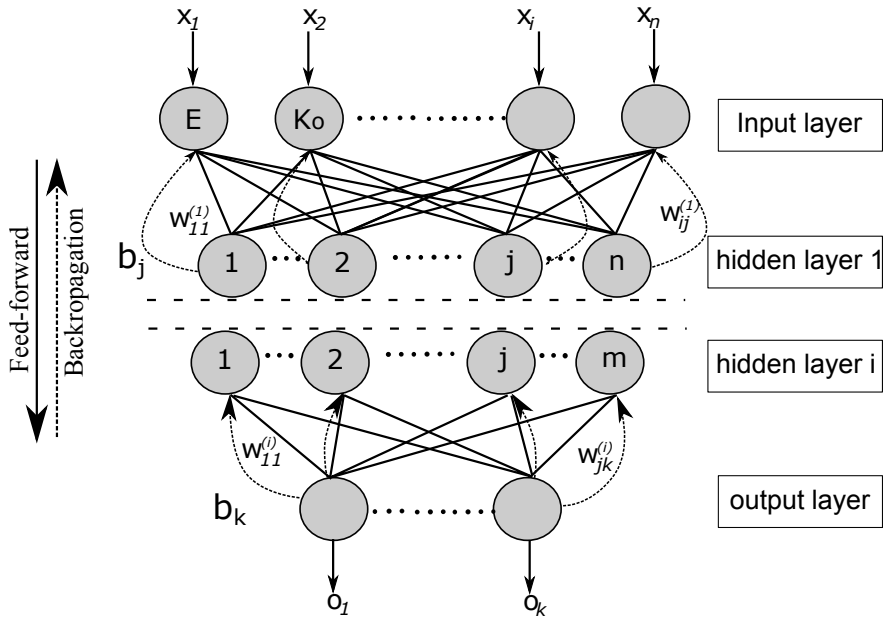


Figure 71: Basic structure of an BPNN

Considering all input nodes x_i $(i = 1, ..., n)$ and output nodes $o_k (k = 1, .., m)$ for a given number of patterns p, a basic back propagation algorithm consists of two steps:

- Feed-forward step
- Back propagation step

The first step is calculated according to Equation (61), where w_{ij} are connecting weights from the input to the hidden layer, w_{ij} are connecting weights from the hidden to the output layer, b_j and b_k are bias, and $f()$ is the activation function used to transform the incoming values and transfer them to the next layer:

$$o_k(x, w) = f\left(\sum_{j=1}^{m} w_{jk} + b_k f\left(\sum_{i=1}^{n} w_{ij} x_i + b_j\right)\right) \tag{61}$$

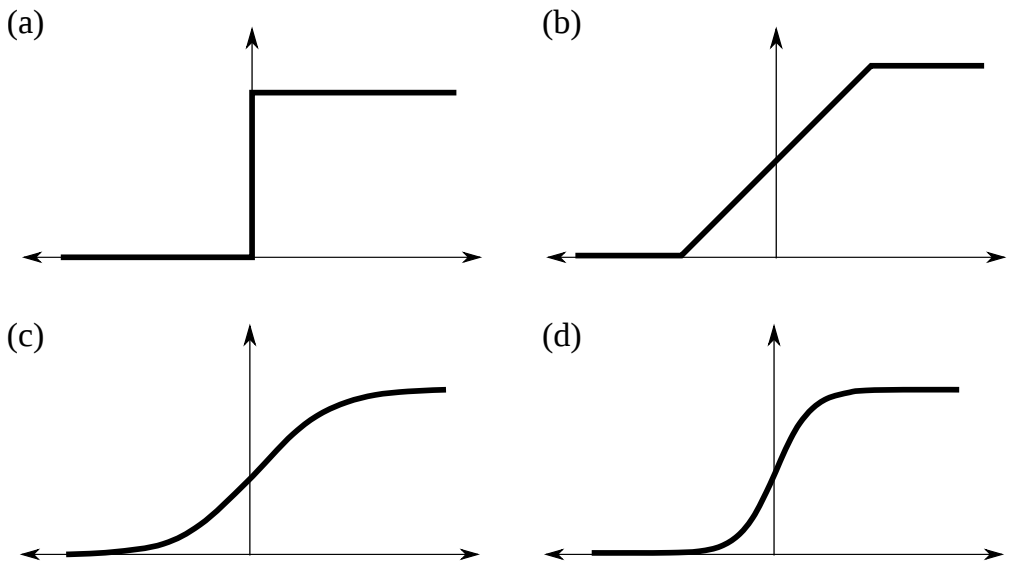


Figure 72: Common activation functions for ANN

An activation function performs a mathematical operation on the signal output. Different types of activation functions can be used. The most commonly used activation functions are:

- **Threshold function** as in Eqn. 62a. It depends on the threshold value. If the summed and weighted input is less than the threshold value, this function give a 0 value, otherwise, it returns 1 (Fig. 72a).
- **Piecewise linear function** as in Eqn. 62b. Like the threshold function, this function can take a value between 0 and 1 value. However, it can also take a value between these two values due to the linear interpolation (Fig. 72b).
- **Sigmoid function** as in Eqn. 62c. The Sigmoid is the most commonly used non-linear function for ANN and has an output range between 0 and 1. It is differentiable, mathematically simple and increases continuously (Fig. 72c).
- **Tangent hyperbolic function** as in Eqn. 62d. The tangent hyperbolic function is also a non-linear function. It is contentious and differentiable. It takes values in the range between 0 and 1 (Fig. 72d).

$$
\begin{aligned}
&(a)\quad f(x) = \begin{cases} 0 & \text{for} \quad x < 0 \\ 1 & \text{for} \quad x > 0 \end{cases} \qquad (b)\quad f(x) = \begin{cases} 0 & \text{for} \quad x < -0.5 \\ ax & \text{for} \quad 0.5 > x > -0.5 \\ 1 & \text{for} \quad x > 0.5 \end{cases} \\
&(c)\quad f(x) = \frac{1}{1 + e^{-ax}} \qquad\qquad (d)\quad f(x) = \frac{e^{ax} - e^{-ax}}{e^{ax} + e^{-ax}}
\end{aligned}
\tag{62}
$$

All output values are compared with the respective target values t_k. A measure of the error between the output and target values is given as:

$$E = \sum_{k=1}^{m} (o_k - t_k)^2 \tag{63}$$

The objective of the second step – the learning process – is to adjust the free parameters (the synaptic weights of the network) to minimize the error (cf. Equation 63). This problem can be solved with a gradient descent method, where the gradient of E with respect to the input quantities is calculated and weights are adjusted incrementally:

$$\Delta w_{ij} = -\gamma \frac{\partial E}{\partial w_{ij}} \quad \text{and} \quad \Delta w_{jk} = -\gamma \frac{\partial E}{\partial w_{jk}} \tag{64}$$

where γ is the learning rate, for which we explain how to determine this parameter below. Note that the learning process can also be accomplished using PSO as a learning method, where the synaptic weights of the network are unknown particle positions, and the fitness is a function of the learning error E.

5.2.4 Particle Swarm Optimization

A parameter identification problem involves the solution of a constrained non-linear optimization problem:

$$\min f(p) \quad \textit{subject to} \quad \mathbf{K}(\mathbf{p})\mathbf{u} = \mathbf{F} \quad \textit{with boundary conditions} \quad \mathbf{p} \in D_p \tag{65}$$

Where f is the objective function, $\mathbf{p}$ the parameter vector, D_p the admissible set of solutions or search space, and u the displacement vector in the FE equation system $\mathbf{K}(p)\mathbf{u} = \mathbf{F}$. In general, to solve this optimization problem, one can either use local or global optimization techniques.

PSO is an evolutionary computational algorithm suggested by Kennedy and Eberhart (1995). The method was developed to simulate social behavior associated with the movement of organisms in a bird flock or fish school. The system is initialized with a population of random solutions and searches for optima by updating subsequent generations. The potential solutions, called particles, "fly" through the problem space by following the current optimum particles. Each particle belongs to a swarm and has two properties: velocity (v_{ij}) and position (x_{ij}). The particle keeps track of its coordinates in the problem space, which are associated with the best solution (fitness), and achieves the particle-best value p_{best} ($x_{ij}^{p_{\text{best}}}$). If a particle takes the complete population as its topological neighbours, the best value is a global best g_{best} ($x_{ij}^{g_{\text{best}}}$). Kennedy and Spears (1998) improved PSO by introducing the inertia weight (w) into the basic update rule to reduce the particle speed when particle swarms are searching a large area. The new velocity and position of the particles is updated in each iteration using the following equations:

$$\begin{aligned} v_{i,j+1} &= w_{ij} + \phi_1 r_1 (x_{ij}^{p_{\text{best}}} - x_{ij}) + \phi_2 r_2 (x_{ij}^{g_{\text{best}}} - x_{ij}) \\ x_{i,j+1} &= x_{ij} + v_{i,j+1}\,. \end{aligned} \tag{66}$$

In Equation 66, r_1 and r_2 represent random numbers uniformly distributed over $[0, 1]$ and ϕ_1 and ϕ_2 are so-called cognition and social learning factors.

(a): Initialize the problem
Define the search space: x_{min}, x_{max}
Define the limit velocities: v_{min}, v_{max}
Define the parameters: w_{ij}, r_1, r_2, ϕ_1. ϕ_2
(b): Initialize the swarms
Create particle list for each swarm i
for each particle
  Set initial particle position x_{ij} and velocity v_{ij}
(c): Start iterative optimization
for the defined number of iterations
  (d) Calculation of objective function values
  For each particle
    Start forward calculation and calculate $Fitness$
    if the stop criteria is met
      break
    if current $Fitness$ is larger than own best in past:
      Save position as own best: p_{best}
    if current $Fitness$ is larger than global best in past:
      Save position as global best: g_{best}
  (e): Update particle position and velocity
  for each particle
    Calculate and set new velocity based on the corresponding equation $v_{i,j+1}$
      if $v_{i,j+1} > v_{max}$; $v_{i,j+1} = v_{max}$
      if $v_{i,j+1} < v_{min}$; $v_{i,j+1} = v_{min}$
    Calculate and set new position based on the corresponding equation $x_{i,j+1}$
      if $x_{i,j+1} > x_{max}$; $x_{i,j+1} = x_{max}$
      if $x_{i,j+1} < x_{min}$; $x_{i,j+1} = x_{min}$
  (f): Go to next iteration
(g): Selection of the optimal solution
$x_{i,j}^{best} = g_{\text{best}}$

Table 3: Algorithm for particle swarm optimization

The *Fitness* relates to the learning error E (Equation 63). It is defined as:

$$Fitness = \frac{1}{\sum_{0}^{p_n} E + TOL} \tag{67}$$

where TOL is a tolerance to avoid a singularity in Eqn. (67). Particles are moving towards the positions leading to the maximum possible values of the *Fitness* function. In Algorithm 3, the basic steps of PSO algorithm are described. In addition to that, the algorithm may contain the update of weights w, or update of maximal velocity of the particle v_{max} as proposed by Shi et al. (1998).

5.2.5 Robust Meta model

The algorithm for the robust meta model training is implemented in Python following the main idea described in Fig. 69 and applying the methods of machine learning described above. For the implementation an exiting python library for supervised learning is used Pedregosa et al. (2011). This tool-kit contains implementation of Regression models, ANNs and SVR, however, in order have a robust learning, a method $PSO()$ is implemented to optimize training parameters of different models machine Learning models. Moreover, following methods are implemented:

- method for reading the data set $ReadDatasetFile()$ with arguments $file$, $training\_portion$, $test\_portion$, $validation\_portion$
- $ComputeError()$ with arguments $train\_method$, $training\_set$, $test\_set$
- $ForwardPass()$ with arguments $weights$, $data\_set$, $method\_parameters$
- $view\_image()$ with arguments $training\_set$, $test\_set$, $validation\_set$

The main function for the robust meta-model training is given in following listing:

Listing 12: Method 1: Optimized meta model training using sci-kit tool

```
from skl\begin{lstlisting}[ basicstyle=\ttfamily, basicstyle=\scriptsize, language=python,frame=single,
from sklearn import linear_model
from sklearn import svm
from sklearn.model_selection import GridSearchCV
from sklearn.neural_network import MLPRegressor
from sklearn.preprocessing import PolynomialFeatures

def main():
    """Training, testing and validation of metamodels."""
    data_file='test_1.txt'
    data= ReadDatasetFile(data_file, 80,15,5)
    bx=data[0][0] #train input
    by=data[0][1] #train output
    bxt=data[1][0] #test input
    byt=data[1][1] #test output
    bxv=data[2][0] #validation input
    byv=data[2][1]      #validation output
    param_nn=[20,10]
    param_svr1=[10,1]
    param_svr2=[10,3]

    meth=[linear_model.LinearRegression(),
        svm.SVR(kernel='rbf', C=5, gamma=0.4),
        svm.SVR(kernel='poly', C=1e3, degree=3),
        MLPRegressor(solver='lbfgs', alpha=1e-4, random_state=1, max_iter=5000,activation='relu'),
        linear_model.SGDRegressor()]
    tot=[]
    i=0
    for clf in meth:
        if i ==1:
            param_svr_new=PsoOptimize(clf, param_svr1, bx, by, bxt, byt, 20, 20 )
            clf.set_params(C=param_svr_new[0])
            clf.set_params(gamma=param_svr_new[1])
            clf.fit(bx, by)
            rrmse=ComputeError(clf, bxt, byt )
            tot.append(rrmse)
        if i ==2:
```

```
            param_svr_new=PsoOptimize(clf, param_svr2, bx, by, bxt, byt, 20, 20 )
            clf.set_params(C=param_svr_new[0])
            clf.set_params(degree=param_svr_new[1])
            clf.fit(bx, by)
            rrmse=ComputeError(clf, bxt, byt )
            tot.append(rrmse)
        if i ==3:
            param_nn_new=PsoOptimize(clf, param_nn, bx, by, bxt, byt, 20, 20 )
            clf.set_params(hidden_layer_sizes=(param_nn_new[0],param_nn_new[1]))
            clf.fit(bx, by)
            rrmse=ComputeError(clf, bxt, byt )
            tot.append(rrmse)
        else:
            clf.fit(bx, by)
            rrmse=ComputeError(clf, bxt, byt )
            tot.append(rrmse)
        i=i+1

    clf=meth[BestModel(tot)]
    rrmse_test=ComputeError(clf, bxt, byt )
    rrmse_train=ComputeError(clf, bx, by )
    tr=clf.predict(bx) #training forward pass
    p=clf.predict(bxt) #test forward pass
    v=clf.predict(bxv) #validation forward pass

    view_image(by, tr, byt, p, byv, v) #plot image
```

5.3 Implementation in Dynamo

Having established robust meta model now we can use it for fast prediction of the effect of design actions in Revit. The In order to do that, a Dynamo node is developed to calculate Forward pass of the meta model based on model parameters set which are set in Dynamo. The trained meta model is imported as a text file, while parameters are set using value boxes or sliders as shown in Figure 73. A visualisation node described in Section 4 is used for graphical interpretation of the meta model prediction.

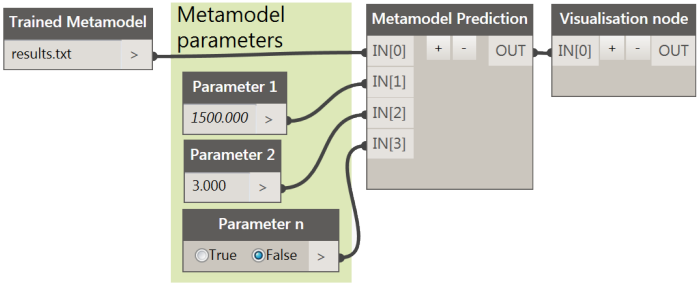


Figure 73: Dyanmo node for real-time prediction using simulation-based meta model

In prediction node we have defined two functions:

- $ReadMetamodel(filename)$ (see in Appendix listing 20)
- $ForwardPass(weights, input, hidden\_array_size, output\_array_size, active\_func)$ (see in Appendix listing 21)

Method ReadMetamodel() is reading the file where the information about trained meta model is stored. Those are the information needed for the forward pass of the meta model. In listing 20 in Appendix, the algorithm for reading the information about trained ANN model is given.

Method ForwardPass() is actually the method which is used to predict the effects of the design actions in real time. The listing 21 in Appendix the source code for prediction based on input parameters, interactively chosen in Dynamo, and trained meta model imported by ReadMetamodel() is given.

5.4 Sensitivity analysts

The tunnelling process is characterized with the large complexity in terms of the geological conditions, construction process and existing infrastructure interacting with the tunnelling process. This complexity of the system directly transfers to the numerical model. The simulation model for mechanized tunnelling is composed of distinct models for the tunnelling components (soil, TBM, grouting, lining, buildings), where each component contains parameters related to the element type and the associated material law. On the other hand, the simulation process contains parameters related to the construction process, for example, changes in the topology of the soil, advance rate, support and grouting pressures, steering of TBM, oscillations of ground water table in time, environmental impact etc. Real in situ condition are characterized by complex topologies with a number of soil layers and changes in the process conditions. Additional variables in contributing to the complexity of those numerical analysis is the LoDs used to include different complexity for the representation of tunnel components. Therefore, for real tunnelling cases, the model is usually described with more than 100 parameters.

The sensitivity analysis are vital tool in SATBIM process for performing following tasks:

- determination of the sensitivity/importance of the component LoD for defined analysis scenario and w.r.t. design parameters,
- general study of sensitivity/importance of design parameters (geometrical, material and process) for predefined analysis scenarios,
- generation of reliable meta models based on important parameters determined by sensitivity analysis.

The first targets is clearly determination of the optimal complexity of the model. As described in SATBIM concept, the LoDs are one of crucial variables of the design and numerical models for determination of complexity and consequently computational efficiency of the analysis. For certain scenarios such determination of stresses in lining, most likely the highest level of detail for the representation of the overlaying infrastructure is unnecessary. However,tool for quantifying our engineering intuition such is "*most likely unnecessary*" is exactly the sensitivity analysis which can be applied to determine in which cases the LoD of the building is insensitive (not important) for prediction of the stresses lining structure. Therefore, the sensitivity analysis are vital tool for determination of the optimal complexity of the numerical model and therefore minimization of the computational costs.

Sensitivity analysis also can be used as an important tool for determination of overall sensitivity and importance of design parameters in mechanized tunnelling. In a simple situation, such for instance unsupported excavation of the soil in homogeneous soil layer, engineering experience and intuition leads us to a conclusion that the soil stiffens is sensitive and important parameter for excavation induced surface settlements of the soil. However, in a complex processes such mechanised tunnelling, where beside the excavation of the soil, the continuous support is provided by means of TBM, lining structure and support measures, and where the tunnelling induced displacements depend on mutual interactions of all structural components (soil, TBM, lining, buildings) and time-dependent effects such consolidation, solidification of the grouting mortar, etc. it is difficult to conclude which parameter is the most sensitive/important for tunnelling induced surface settlements. This for instance is highly dependent on the ranges of parameters for certain tunnel case. For example, the sensitivity of the soil stiffens on tunnelling induced settlements will considerably differ for cases stiffness range from 5-200 MPa (large variation from low to medium stiffness) and 500-600 MPa (narrow range of high stiffness). Those ranges are usually suggested in geotechnical reports and can vary significantly. Also, process parameters such support and grouting pressure and advance rate are playing a crucial role in control of tunnelling induced displacements. The design ranges of an their sensitivity and effect on tunnelling induced displacement plays a crucial role in overall stability of the system and safety during the construction. Therefore, having design and analysis tools like SATBIM which enables fast and flexible generation of numerical models on one hand, and sensitivity analysis on the other hand, offers an excellent opportunity to investigate the importance of design parameters in such complex construction process. This can result in tables and recommendations for design engineers or, since the SATBIM will be established as an open source platform, even in a tool which can be used for specific projects for design analysis and optimisation.

Finally, if predictions in real time are required, and the simulation based meta models are used as a tool, it is necessary to insure the robustness and reliability of those meta models. As explained in the first paragraph of this subsection, the design and consequently numerical models are characterized with large number of parameters. The generation of simulation-based meta models requires a data set which is generated in so called " numerical experiments" and is strongly dependent on the number of investigated parameters. The accuracy of an approximation obtained from a meta model usually depends, besides the number of the training points, on the location of the training samples inside the input parameter space (the design domain). For this reason, the methods of design of experiments (sampling techniques such: Hammersley Sequence sampling (HS), Latin Hypercube sampling (LH), Pseudo-Random sampling (RD)) are commonly used to select the training points. Increasing the number of parameters to be considered for generation of the meta model increases the number of training samples (numerical simulations) exponentially and therefore the computational time. Secondly, if the insensitive parameters are considered for generation of the meta models, this will a) affect the robustness of the meta model and b) increases the computational time. Therefore, for generation of the robust meta models with optimal computational costs the sensitivity analysis has to be conducted to determine a set of important parameters sensitive to the output of the model.

5.4.1 One At a Time design

Screening methods have shown good performance for studying sensitivities for problems containing several tens of input variables. This method is based on the discretization of the inputs in levels, allowing a fast exploration of the model behaviour and identification of the influential inputs with a small number of model invocations. The most widely used screening method is One At a Time (OAT) design, where the input space for each variable is discretised and the output of the model is observed for the variation of each of this input parameter while fixing the others. Here, we are focusing on the Morris method (Morris, 1991), which, using OAT design, allows to classify the inputs in three groups: inputs having negligible effects, inputs having large linear effects without interactions and inputs having large nonlinear and/or interaction effects. In the Morris method, OAT tests are randomly selected from the discretized input space of X_j for each variable j ($j = (1, .., k)$). The elementary effect EE_j^i of the j-th parameter X_j in the i-th repetition is calculated using Equation 68, where $Y(X_j)$ is an output and Δ is value in $\{1/(p-1), ..., 1-1/(p-1)\}$, where p is the number of intervals of X_j.

$$EE_j^i = \frac{Y(X_1, X_2, ..., X_j + \Delta, .., X_k) - Y(X_1, X_2, ..., X_j, .., X_k)}{\Delta}, \tag{68}$$

The indicators of sensitivity of an input to an output are μ (mean), σ (standard deviation) and μ^* (mean of the absolute value), given in Equation 69a, b and c, respectively. μ estimates the overall influence of the input parameter on the system response. The standard deviation σ detects the interaction effects with the other parameters as well as the nonlinear relation between the corresponding input/output. If σ_j is small, elementary effects have low variations, suggesting a linear relationship between the studied input and the output. On the other hand, if σ_j is large, this implies nonlinear effects of the input on the output or an interaction of the input with other variables. μ^* is the mean of the distribution of the elementary effect's absolute value $\|EE_j^i\|$ for n OAT tests.

$$(a)\ \mu_j = \frac{1}{n}\sum_{i=1}^{n} EE_j^i, \qquad (b)\ \sigma_j = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(EE_j^i - \mu_j)} \qquad (c)\ \mu_j^* = \frac{1}{n}\sum_{i=1}^{n} \|EE_j^i\|, \tag{69}$$

Saltelli et al. (2008) suggested that for understanding and reliably estimating the sensitivity of an input parameter, the effects of μ and σ have to be combined, while calculating μ^* does not require any additional evaluations. The application of global sensitivity analysis to the evaluation of key soil parameters for numerical simulations in mechanized tunnelling is described in (Miro et al., 2014).

The sensitivity of inputs strongly depends on the range of chosen parameters and the target of evaluation. In the case of a mechanized tunnelling project, in geotechnical reports a range of admissible parameters describing material properties, water-soil conditions, topology etc. is often given and can be used to conduct sensitivity analyses. The result of the sensitivity analysis is then used to create a meta model based on important parameters.

5.4.2 Implementation

Calculation of the sensitivity measures is implemented in Python, as shown in listing 13. For the forward calculation of the effect of parameter variation onto the output the meta models trained by means of FE simulations are applied. This offers significant advantage since calculation of the elementary effect requires large number of forward calculations to insure the uniqueness of the solution. As given in the previous section, the meta models are excellent tools for interpolation and extrapolation of the trained data set, and therefore reasonable solution for the forward calculations like this. In the listing 13 the following steps are performed:

- define the set of parameter ranges to be investigated (bxt),
- use the meta model to predict the model output ($sens\_test$) based on given input (bxt) and trained meta model synaptic weights ($weights$) using $ForwardPass$ function described in 21,
- calculate the μ^* (mean of the absolute value of EE_i) using function $ComputeVariance$ given in Appendix in listing 22,
- calculate the σ (standard deviation of EE_i) using function $ComputeDeviation$ given in Appendix in listing 23.

Listing 13: Method 1: Algorithm for sensitivity analysts based on meta models

```
time_steps=25
input_size=4
n_delta=10
weights="D:/git-satbim_internal/Examples/sensitivity/weights/weights_build_settlement.dat"
arch_ann=[38, 27]

def main():
    w=ReadWeights(weights)
    sens=[]
    devs=[]
    bxt=[[0 for x in xrange( input_size)] for x in xrange(n_delta*time_steps)]
    for k in range (0, 3):
        for i in range (0, n_delta):
            for j in range (0, time_steps):
                bxt[i*time_steps+j][0]=0.1+0.4*k
                bxt[i*time_steps+j][1]=0.1+0.8/(n_delta-1)*i
                bxt[i*time_steps+j][2]=0.5
                bxt[i*time_steps+j][3]=0.1+0.8/(time_steps-1)*j

        sens_test=ForwardPass(w, bxt, arch_ann, 1, 'relu')
        variance=ComputeVariance(sens_test, n_delta, time_steps)
        dev=ComputeDeviation(sens_test, n_delta, time_steps)
        sens.append(variance)
        devs.append(dev)
```

5.5 Numerical examples

5.5.1 Instant prediction using simulation-based meta models

In this example, SATBIM is applied for a (synthetic) example of design assessment of the tunnel alignment alternatives in the vicinity of existing infrastructure. The tunnel is connecting points A and B as illustrated in Fig. 74b. The design question is the optimal tunnel alignment and tunnel depth w.r.t. risk induced on the existing building. To this end, 25 TIM models for the given problem were generated considering five different alignment alternatives and five overburdens (see Fig. 74b). The representation adopted for this model is: soil on LoD 2, lining on LoD 1, TBM on LoD 0 (see dependency), and building on LoD 3. A possible resulting model is illustrated in Fig. 74a.

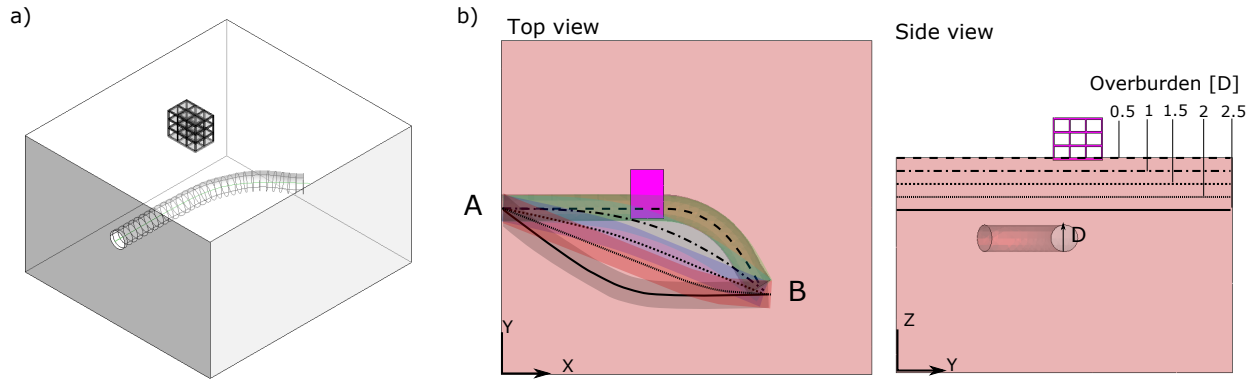


Figure 74: a) TIM for investigation of alignment design alternatives; b) design alternatives in terms of different alignment from point A to point B and depth of tunnel w.r.t. foundation of the existing building.

The 25 information and numerical models combining two parameters (alignment alternative and overburden) are generated with minimum efforts, since, as explained previously, the multi-level TIM is fully parametrized and the simulation model generation is fully automated. The simulation models are then executed in parallel on a computing node with 32 processors (two simulations at a time, each using 16 threads), taking advantage of a shared memory implementation based on openMP. The results of FE simulations for differnt tunnel alignment and overburden are shown in Fig. 75. The total calculation time was approximately 11 h. The simulation results (surface settlements and deformation of the building top) are output in a form suitable for subsequent training of the ANN. The meta models were trained with an overall relative Root Mean Square (rRMSE) accuracy of 4.2 %. The trained ANN (adjusted synaptic weights) is imported into Revit/Dynamo, or the design environment, and the forward model is executed by the respective Dynamo node as explained in Section 5.3. Figure 76 shows the surface settlements and deformation of the building based on meta model predictions using the visualisation algorithms implemented in Dynamo. When looking at the top of the building, it is clear how the influence of the tunnel excavation changes from negative (red) to significantly improved (green) by setting the alignment to pass around the building (alternative 5) and by increasing the tunnel depth. It is possible to visualise this in real-time by setting the parameters (using a slider user interface)

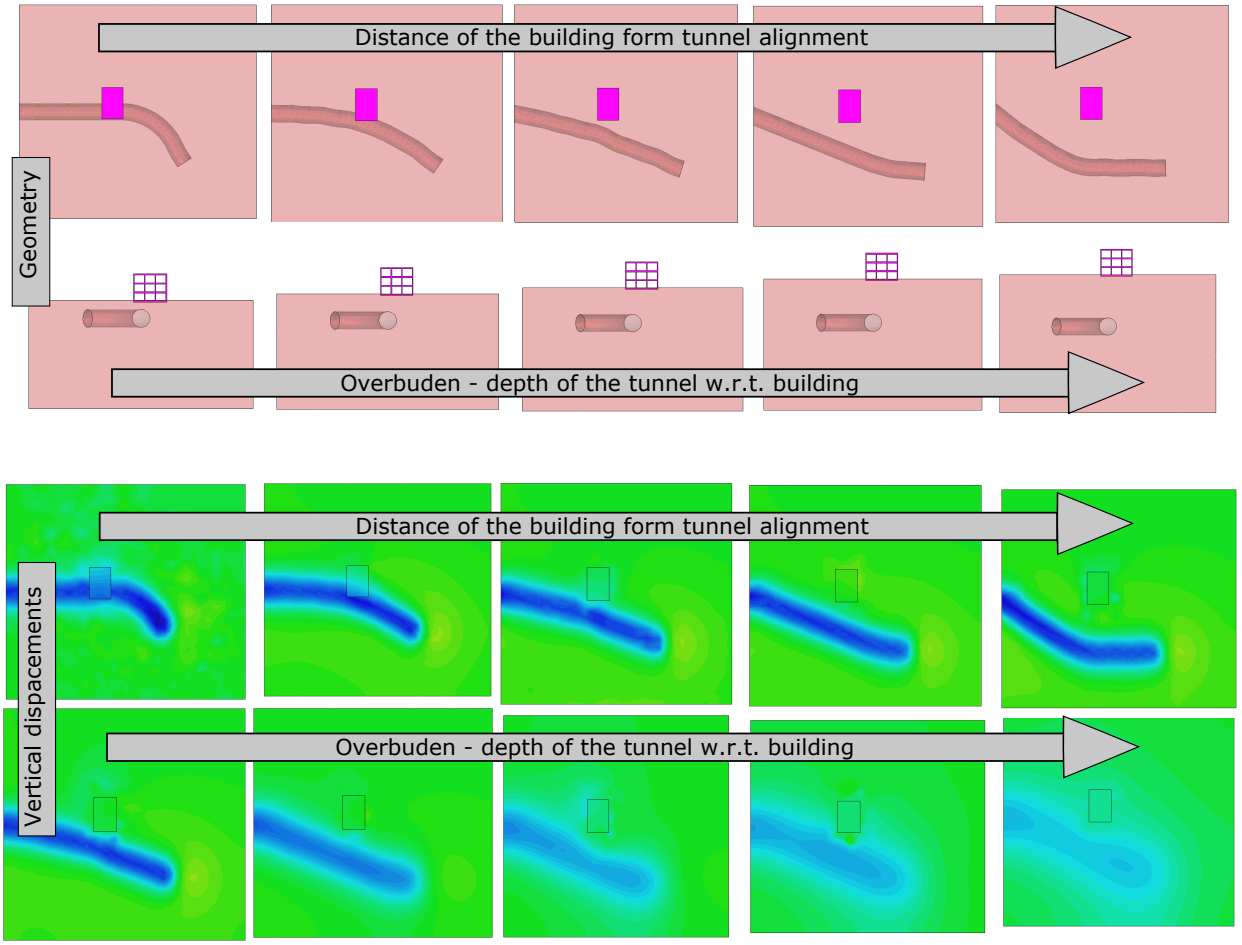


Figure 75: Meta model geometry and results of FE simulation for different simulation samples

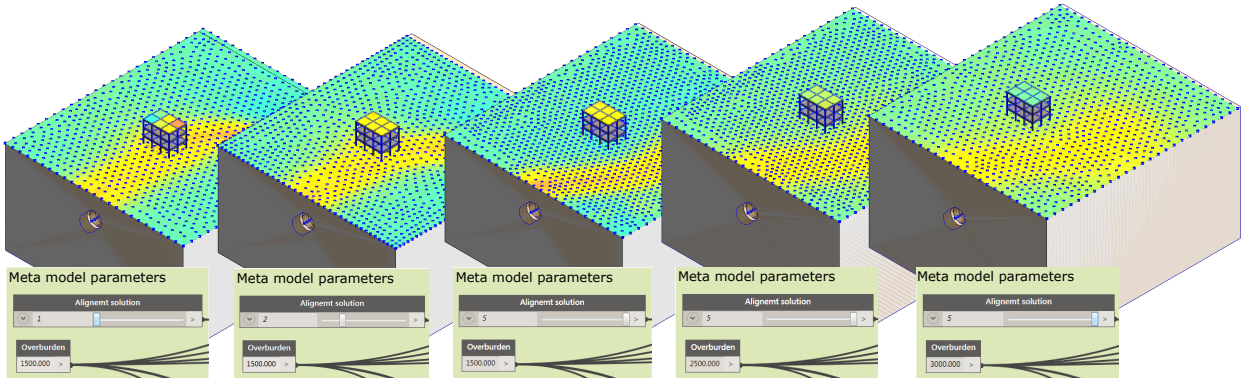


Figure 76: Real-time assessment of the soil-structure interaction w.r.t. tunnel alignment solution in TIM.

describing the geometry of the alignment in the Dynamo window shown in Fig. 76.

Utilizing the latest technology like a multi-touch multi-user video wall (Fig. 77), the design

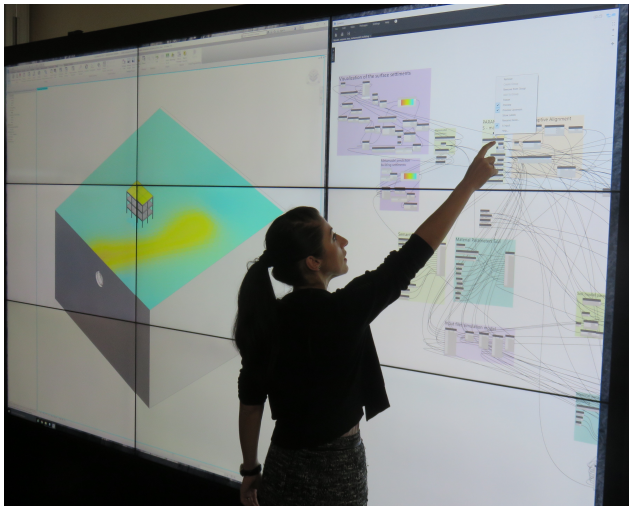


Figure 77: Real-time assessment of the design alternatives in terms of soil-structure interaction effects on multi-user multi-touch video wall .

alternatives can be evaluated directly to support design, collaboration and decision making in important meetings, for example with investors.

5.5.2 Example: investigation of building LOD sensitivity using meta model

SATBIM is applied for investigation of the sensitivity of choice of the building LoD, building position and tunnel depth to building deformations caused by tunnelling. The tunnel is passing in the vicinity of the existing building as illustrated in Fig. 78. Design and modelling parameters investigated in this study are building LoD (1, 2 and 3), building distance from the tunnel in Y direction $(0, 10, 20, 30, 40, 50, 60\,\mathrm{m})$ and depth of tunnel crown w.r.t. foundation of the existing building $(5, 10, 15, 20, 25\,\mathrm{m})$ as shown in Figure 79. For the remaining components the representation adopted for this model is: soil on LoD 2, lining on LoD 1 and TBM on LoD 0 (see dependency). The tunnel diameter D in this example is 10 m and the construction of 27 rings (steps) of 2.5 m length is modelled. In this example the Mohr-Coulomb model is used as the constitutive relation for effective stress and strain in the soil. This constitutive model is often used to model the plastic flow of cohesive soil. In this work, the constitutive model is assumed associative and hardened linearly beyond the yield limit. TIM described in Section 2 is used for generation of the 105 tunnelling information models combining the those three parameters, as shown in Figure 78 top. Based on this Revit model, a simulation model is generated using `SatBim Modeller` and the results for the equivalent information models are shown in Figure 78 bottom.

In this numerical experiment containing 105 simulations for the construction of 25 tunnel rings with combination of three parameters (building LoD, overburden and distance) selected monitoring quantities are: a) the vertical displacements of the building top and b) relative vertical displacement of the building in traversal direction δy. This resulted in two data sets of **2554** monitoring samples. These two data sets are used for generation of two meta modes: one of prediction of building settlement and second for prediction of the building inclination

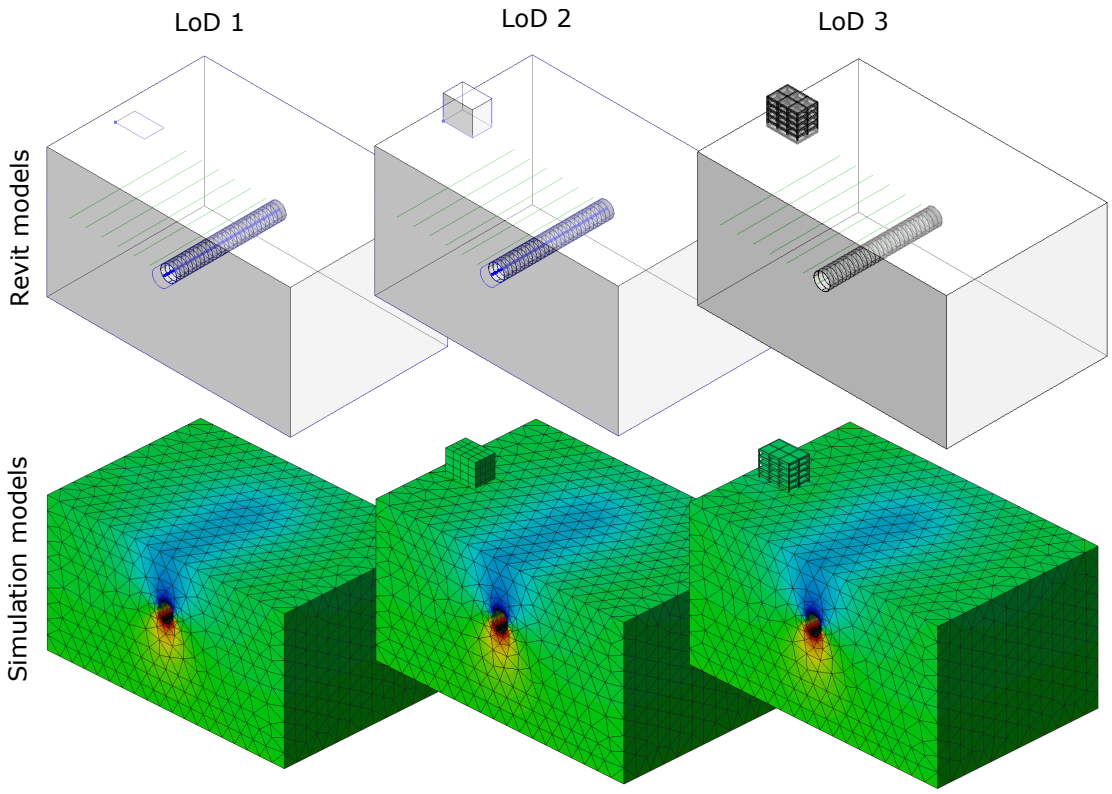


Figure 78: Top: TIM for investigation of building LoD sensitivity ; Bottom: Simulation models for the respective TIM models

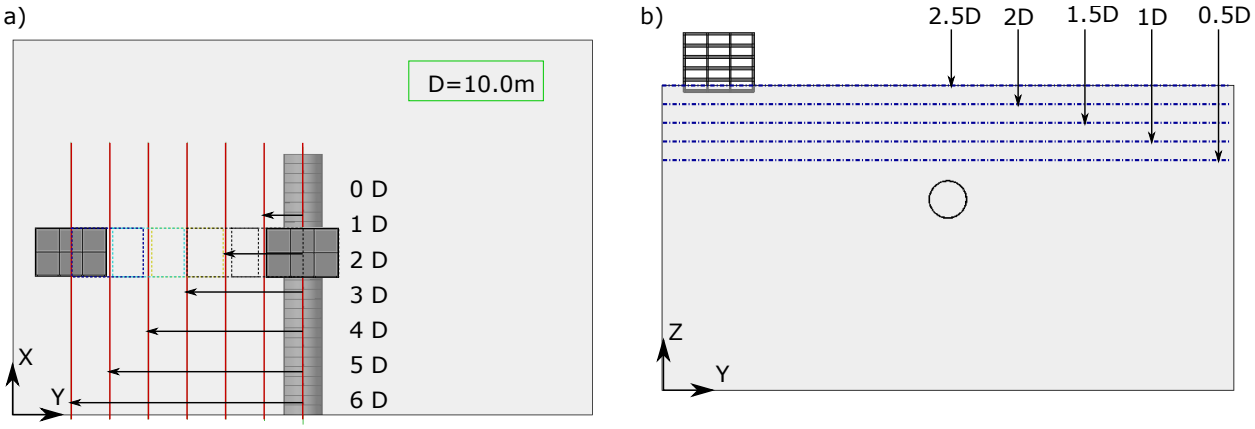


Figure 79: Parameters for investigation of the building LoD sensitivity. a) Design alternatives in terms of building distance from the tunnel alignment; b) depth of tunnel w.r.t. foundation of the existing building.

w.r.t building LoD, overburden and distance. In the next step, those meta models are used for the forward calculations in sensitivity analysis.

Meta model training for the prediction of building settlement The procedure and the algorithm described in Section 5.2 is applied for the meta model training based on data set of **2554** samples. Here, the data set is divided to a portions of 80, 15 and 5 % samples for training, testing and validation, respectively. Figure 80a shows the the relative Root Main Square Error (rRMSE) (see Eqn. 70) for different machine learning techniques applied for the data training. From the Figure 80a is clear that the optimized ANN shows the best performance for data set training. Figure 80b shows the convergence of the optimization of the ANN architecture leading to a minimized error for the test sample. PSO is used as optimization algorithm to determine the number of neurons in hidden layers, and from the Figure 80b it can be seen that the optimal solution is reached within just couple of iterations.

$$r\text{RMSE} = \sqrt{\frac{1}{p_n}\sum_{0}^{p_n}\left(\frac{o_k - t_k}{t_k}\right)^2}, \tag{70}$$

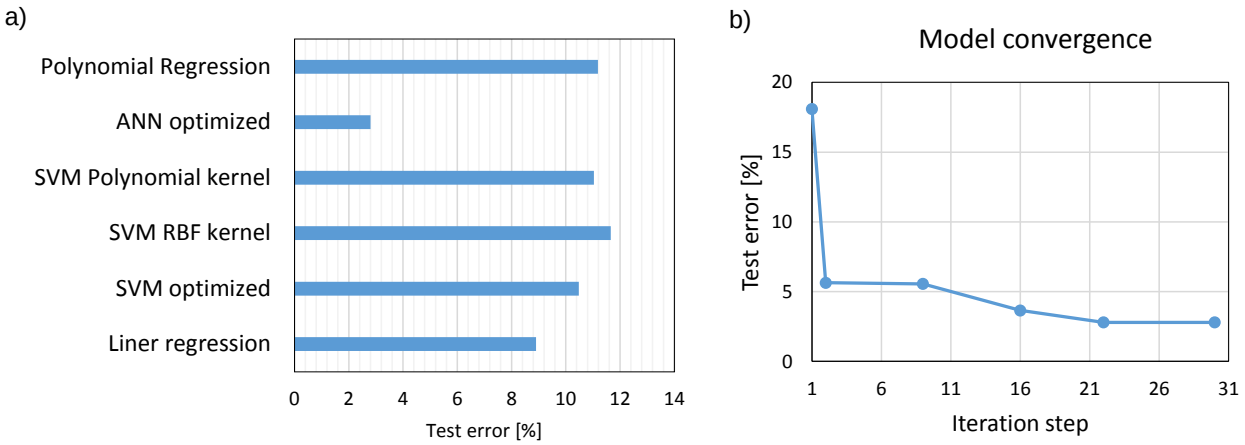


Figure 80: a) Training performance of the meta model using different machine learning techniques presented in Section 5.2. b) Convergence of the optimization process of ANN architecture.

Figure 81 shows the comparison between the data set and meta model prediction for the training, testing and validation set for the best meta model. From this figure it can be concluded that optimised ANNs meta model has excellent prediction capabilities with error for test and validation set less than 3%.

Sensitivity of the building LoD w.r.t. overburden and distance from the tunnel A variance-based global sensitivity analysis has been conducted in order to measure the sensitivities of the model output (settlements at the building top) to the aforementioned input parameters. In this methodology the importance of the input parameter is quantified through two sensitivity measures:

- Absolute mean $\mu*$ of the elementary effect EE_i, representing the total sensitivity index and a measure of the overall effect of a factor on the output, and

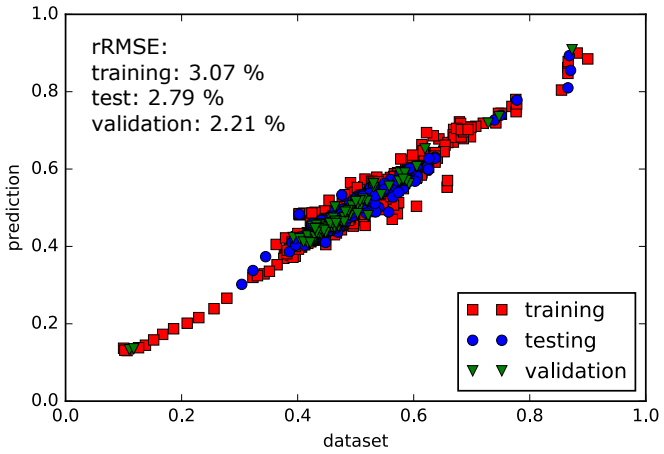


Figure 81: Comparison between vertical displacements of the building obtained from the FE simulation and predictions of the trained surrogate model for the training, test and validation sets for the meta model with best performance.

- standard deviation σ of the the elementary effect EE_i which detects the interaction effects with the other parameters as well as the nonlinear relation between the corresponding input and the system output.

In Figures 82 and 83 the sensitivity measures for the selected LoD of the building to vertical displacements are shown. From from the both plots we can conclude that global sensitivity indicated by $\mu*$ as well as interaction effect indicated by the σ of the LoD drops with the increase of the distance of the building from and overburden of the tunnel. It is for instance obvious that when the distance of the building form tunnel is approximately 4D, the selected building LoD becomes irrelevant, meaning that we can choose the lowest LoD and reduce computational costs.

Looking more closely at the results of the sensitivity measures $\mu*$ and σ of the building LoD for different tunnel overburden (Figure 83) we can see that the both global sensitivity and interaction effect drop with the increase of the overburden, however, are not insignificant even for the overburden of 25 m, especially σ which detects nonlinear relation between the input and output. This is due to two reasons: first, because the chosen limit of the overburden of 25 m (2.5 D) from tunnel crown is still zone of large influence, and second due to building distance from the tunnel in Y direction of 0 m, where the interaction effect is the strongest (see Figure 82).

Sensitivity of the building position to the tunnelling induced building deformation The elementary effect EE_i of certain parameter, is relative influence of the change of this parameter (input) onto output, while all other parameters are fixed. This clearly means that the sensitivity of that parameter is dependent on the value of other fixed parameters too. Therefore, when investigating the sensitivity of the building distance from the constructed tunnel, we show the global sensitivity indicator $\mu*$ (Fig.84) and the indicator of non-linearity σ (Fig.85) for two different overburdens: 10 and 25 m. From these plots is clear

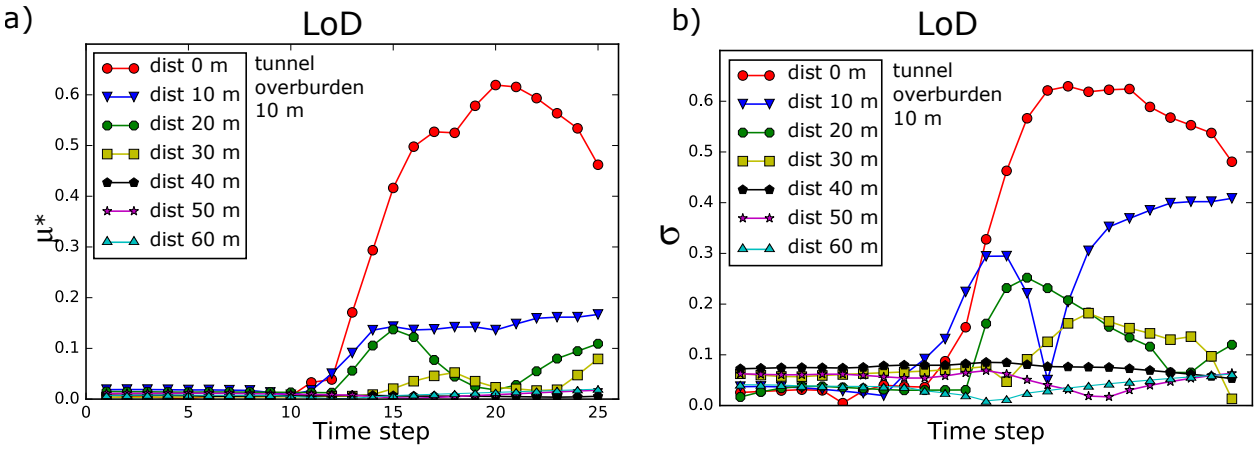


Figure 82: Sensitivity of the building LoD for different building distance from the tunnel alignment for tunnel overburden of 0 m: a) Absolute mean of EE_i; b) Standard deviation of EE_i.

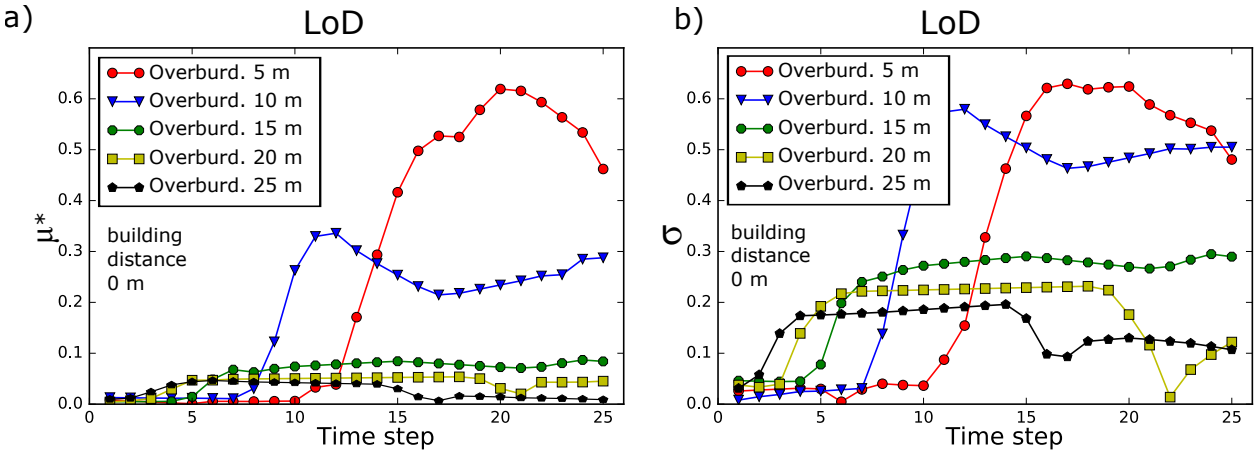


Figure 83: Sensitivity of the building LoD for different tunnel overburden for building distance of 0 m: a) Absolute mean of EE_i; b) Standard deviation of EE_i.

that the global sensitivity and nonlinear effect is much larger for the smaller overburden, which is due to the fact that in that case the settlement trough is more narrow and deeper, than in the second case, which leads to larger settlements and therefore deformations of the building.

Second effect which can be observed form those plots is difference in sensitivity measures for different LoDs of the building. For very low overburden the highest global sensitivity is observed for building LoD 2, probably due to very stiff response to steep settlement trough. The non-linear interaction effect indicated by σ is dominant for both building LoD 2 and 3 which is expected since in both cases structural model are used compared to LoD 1 where the building is not explicitly modelled. However, for larger overburden sensitivity measures are quite similar for each building LoD, which again brings us to a conclusion that with increase of tunnel depth and wider and shallower settlement trough the influence of the selected LoD becomes less significant.

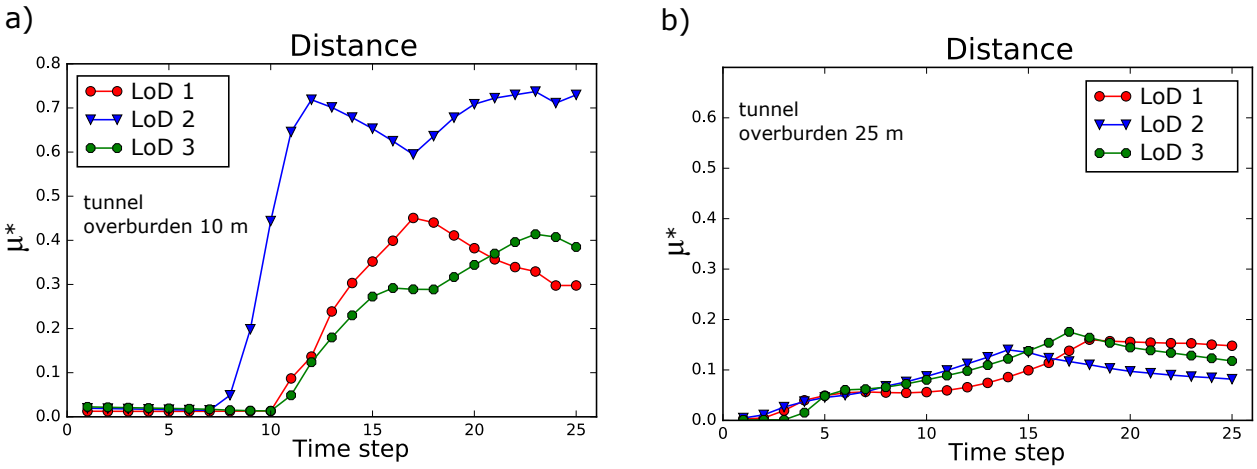


Figure 84: Absolute mean of EE_i of the building distance from the tunnel for different building LoDs for: a) Tunnel overburden of 10 m; b) Tunnel overburden of 25 m

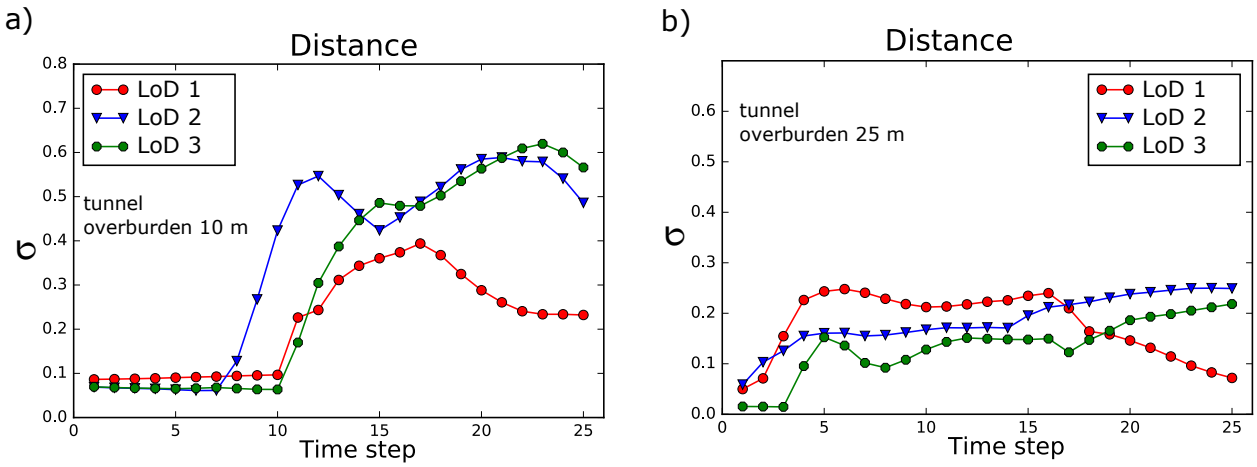


Figure 85: Standard deviation of EE_i of the building distance from the tunnel for different building LoDs for: a) Tunnel overburden of 15 m; b) Tunnel overburden of 30 m

Observing the sensitivity of tunnel overburden to the tunnelling induced building deformation (Figures 86 and 87)similar trend can be seen as in previous case. Firstly, if the building is located closer to constructed tunnel (15 m) the sensitivity is high, and is more dominant for building LoD 2 and LoD 3. Also, the effect of non-linearity and interaction σ is highest for the LoD 3, than the LoD 2 and lowest for LoD 1 which is due to the fact that the interaction is strongest for the detailed structural model of the building, and decreases with the level of complexity. Secondly, if the building is located far away from the tunnel (60 m) the settlements are insensitive to change of overburden, i.e. tunnel depth which is also shown in Figure 83.

Meta model training for the prediction of building inclination The relative building deformation in traversal and longitudinal direction is an important measure of building safety. This is due to fact that for the building safety often is more serious relative dis-

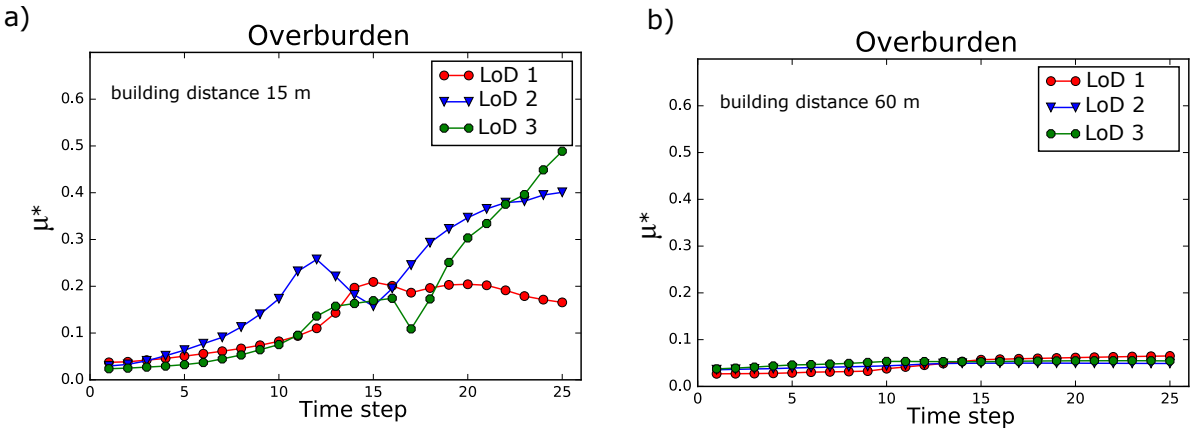


Figure 86: Absolute mean of EE_i of the tunnel overburden for different building LoDs for: a) Building distance of 15 m; b) Building distance of 60 m.

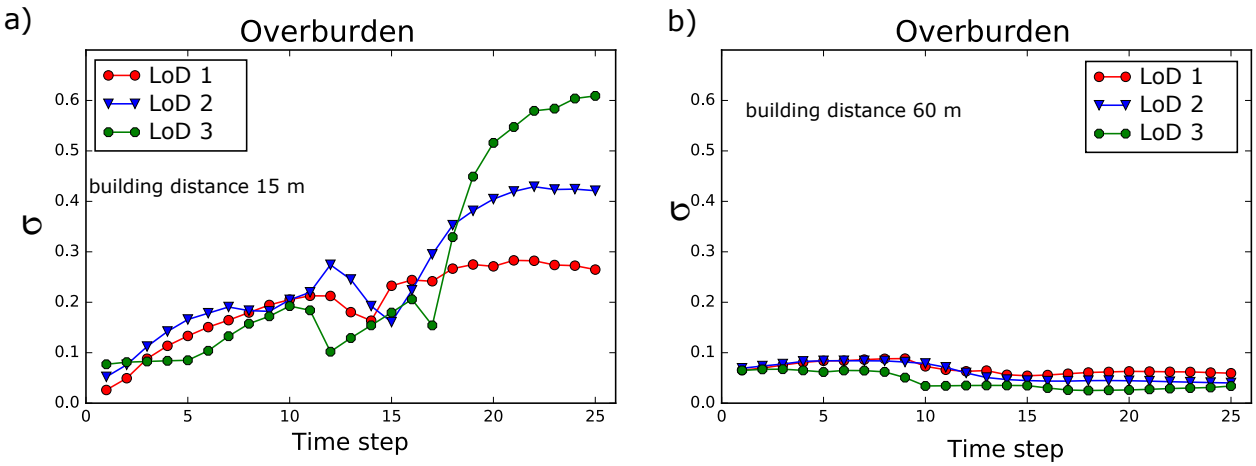


Figure 87: Standard deviation of EE_i of the tunnel overburden for different building LoDs for: a) Building distance of 15 m; b) Building distance of 60 m.

placement (inclination) form the absolute maximum settlement of the building, since this settlement might be uniform and therefore similar to rigid body motion which does not induce stresses in building and consequently no damage. On the other hand, the relative displacement is often connected with the bending of the building, which in the case of sensitive buildings can cause the damage or even collapse of the structure. To investigate this, the meta model generation procedure described in Section 5.2 is again applied for training of the data set of **2554** samples showing building inclination during the tunnel construction. Same as in previous case, the data set is divided to a portions of 80, 15 and 5 % samples for training, testing and validation, respectively. Figure 88 shows the comparison between the data set and meta model prediction for the training, testing and validation set for the best meta model. In this example as well optimised ANN was the meta model with the best performance compared to others with the $rRMSE$ for the training of 2.98%, for the test data set of 3.39% and for validation data set of 3.41%.

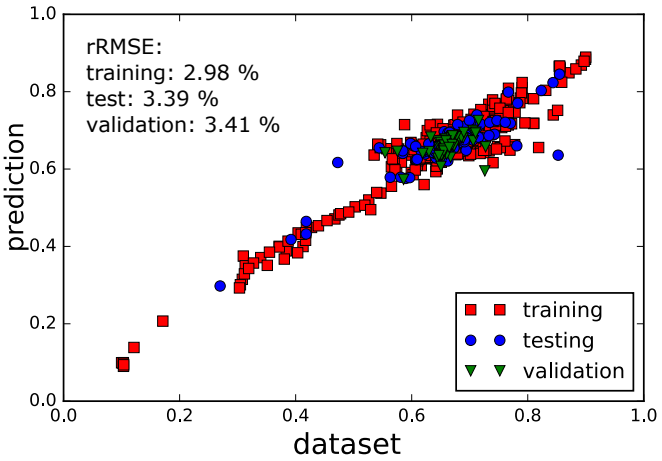


Figure 88: Comparison between building inclination obtained from the FE simulation and predictions of the trained surrogate model for the training, test and validation sets for the meta model with the best performance.

Sensitivity of the building position and tunnel overburden to the tunnelling induced inclination of building Figures 89 and 90 depict the sensitivity indicators $\mu*$ and σ for distance of the building from the constructed tunnel and tunnel overburden w.r.t. induced inclination of the building in traversal direction. For both parameters building model at LoD 1 shows little or no sensitivity and interaction compared to building models at LoD 2 and LoD 3. It is interesting to notice that building distance is mostly sensitive for building model LoD 2 while overburden is more sensitive with building model LoD 3, which is trend similar to sensitivity of the same parameters to absolute settlement. However, the evolution of sensitivity for different construction steps shows completely different trend for inclination compared to total settlements, and either is relatively constant (for distance) or decreases (for overburden) with the advance of tunnel construction.

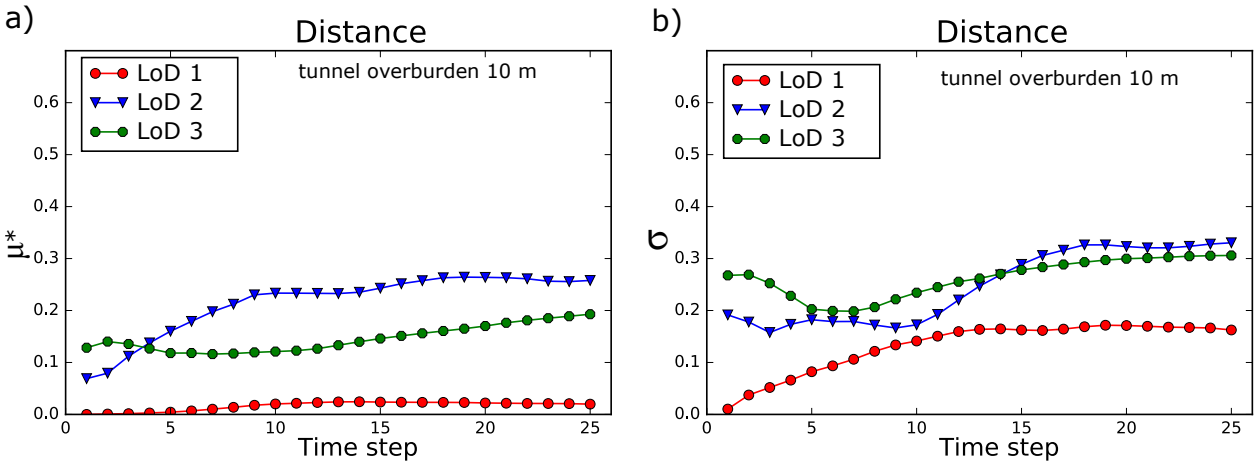


Figure 89: Sensitivity of the building distance from the tunnel to building inclination for different building LoDs for tunnel overburden of 10 m: a) Absolute mean of EE_i; b) Standard deviation of EE_i.

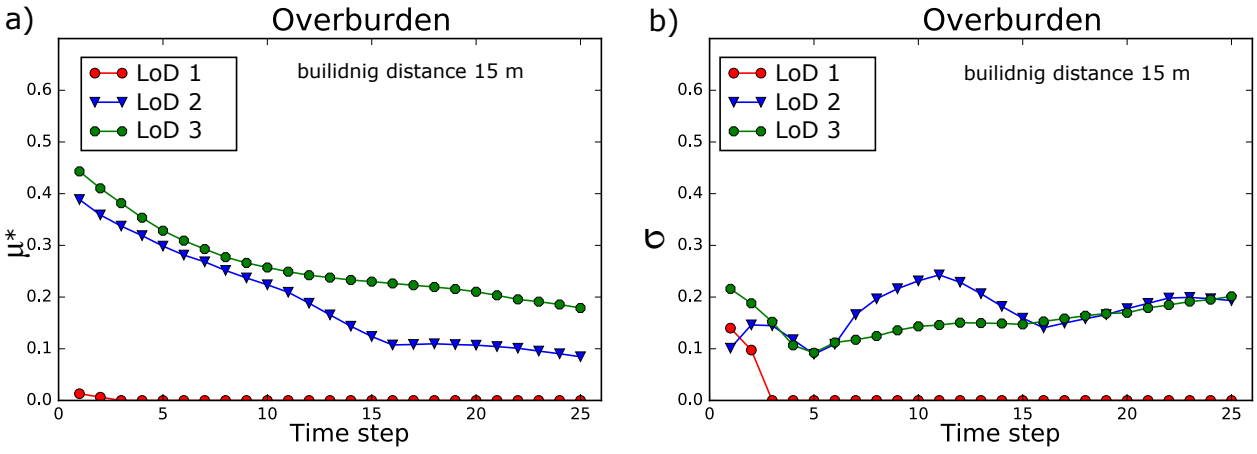


Figure 90: Sensitivity of the tunnel overburden to building inclination for different building LoDs for tunnel distance of 15 m: a) Absolute mean of EE_i; b) Standard deviation of EE_i.

6 Validation based on a real industry project

6.1 Project data

For the validation of the SATBIM framework a project data of the real tunnel are used. The data is related to a railway tunnel that is currently under construction. Construction began in 2013 and it is expected to open in 2022. The 4,270 metre tunnel will pass under the urban area. The distance between the track-centres of the two single-track tunnels is 26.5 metres and they are linked at 500 metre intervals by cross passages. To the north and south are trough structures with a length of 800 or 895 m connecting to the rail tracks on the surface.

The planned cover of the tunnels is between 3 to 20 metre. The tunnels will run through sandy-gritty subsoil, mostly under the water table.

6.2 Tunnel Information Model for the industry project

6.2.1 Modelling of the soil

The data about the surrounding soil is given in Geotechnical reports. The ground along the alignment consists of almost homogeneous, horizontally layered soil. For the information model the ground model has been generated from 25 geo-referenced boreholes along the tunnel route with approximate lenght of 1 km (from $96 + 8$ to $97 + 8$ km tunnel trace).

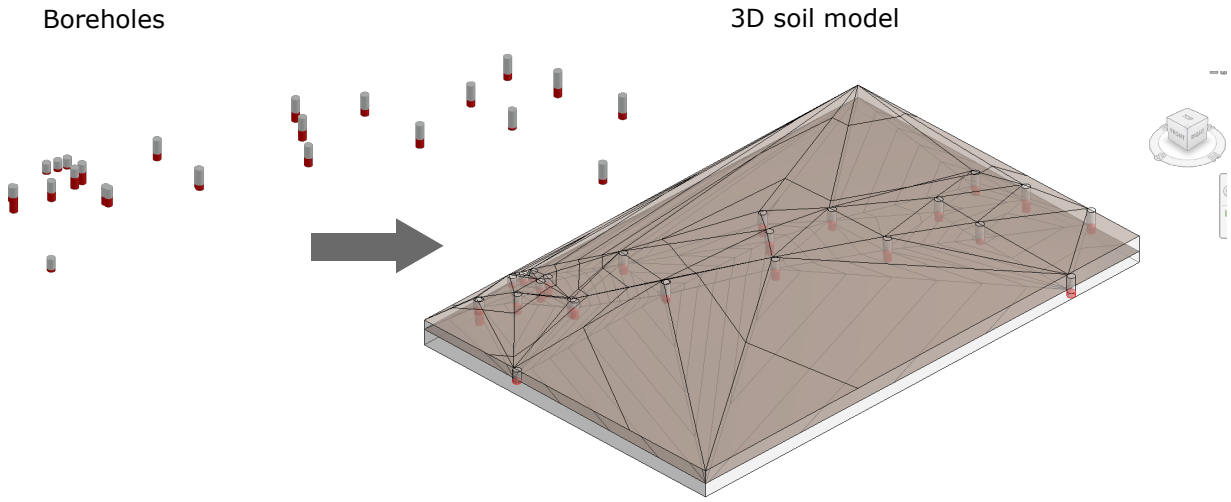


Figure 91: Soil boreholes for tunnel section and Revit model for the soil layers generated based on those boreholes.

The borehole data is shown in Figure 91 left. Based on this data a three-dimensional ground model, contacting two soil layers, is generated with a help of Dynamo as shown in Figure 91 right. In Figure 92 the Dynamo nodes for generation of 3D soil bodies representing the soil is shown. The procedure consists of two step:

- selection of the borehole faces representing the boundaries between the layers, and generation of Topology based on midpoints of those faces;
- based on generated Topological surfaces between the layers, as well as additional surfaces in XZ and YZ pane, which are closing the volumes between vertical layered topological surfaces, solids are generated and unified.

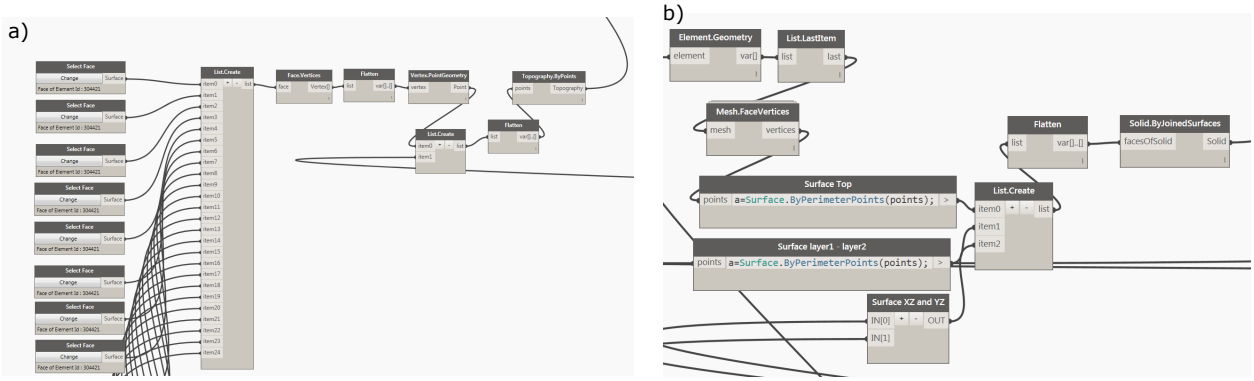


Figure 92: Dynamo node for generation of soil layers based on imported Borehole data.

A unified solid body, composed out of solid volumes representing the soil layers is than exported as *.sat* geometry, and later will be imported in TIM model as optional representation of the soil on LoD3 (see Section 2.4) .

6.2.2 Modelling of the tunnel alignment and lining

General routing for the tunnel is given in a full length of 4,270 m, as shown in Figure 93. In a zoomed section in the same figure we may see that the lining rings are already aligned along tunnel route in the design drawings. The lining ring with outer diameter of 10.6 m and thickness of 0.5 m and length of 2.0 m is composed of 7 segments i.e. with 6+1 segment pattern. The ring are conical with bottom length of 2.0225 m and top length of 1.9775 m. This continuity leads to curvature of ≈ 400 m, i.e. the angle $\alpha = 1.29°$.

Based on all geometrical data describing the lining ring (inner and outer radius, length, conicity and number of rings), without importing the actual ring design, the parametric modeller for lining presented in Section 2.3 is used to generate the lining model just based on imported tunnel alignment from Auto CAD drawings (see Figure 95). This approach leaves us the flexibility to choose the LoD of the lining structure representation w.r.t the perspective we observe during the modelling and the problem we want to solve.

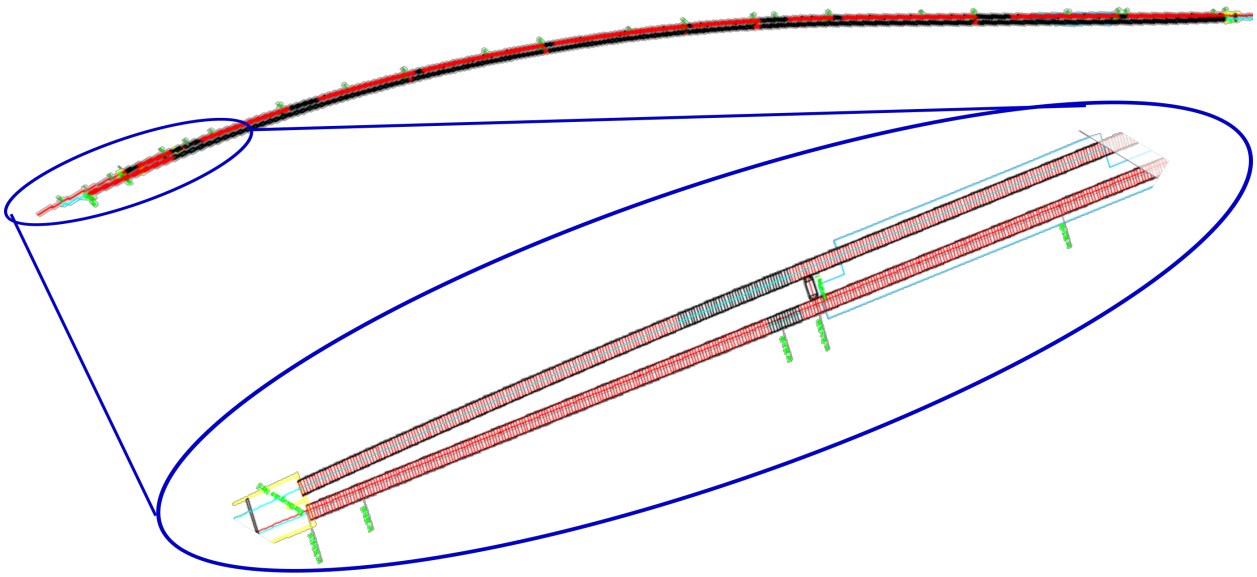


Figure 93: Auto CAD drawings of tunnel alignments and distribution of lining rings .

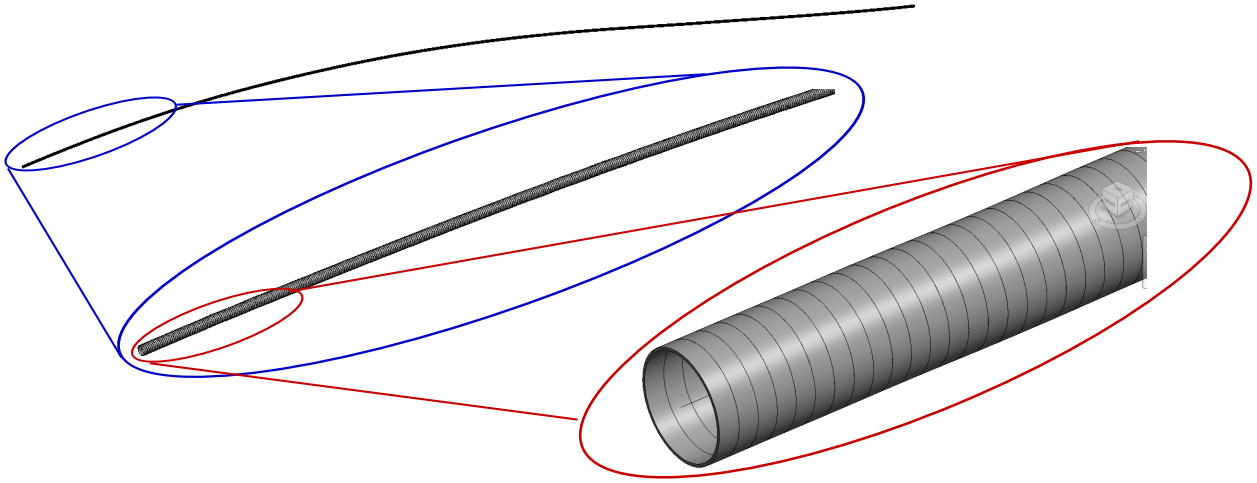


Figure 94: Revit model of the tunnel lining structure based on the alignment in Figure 93.

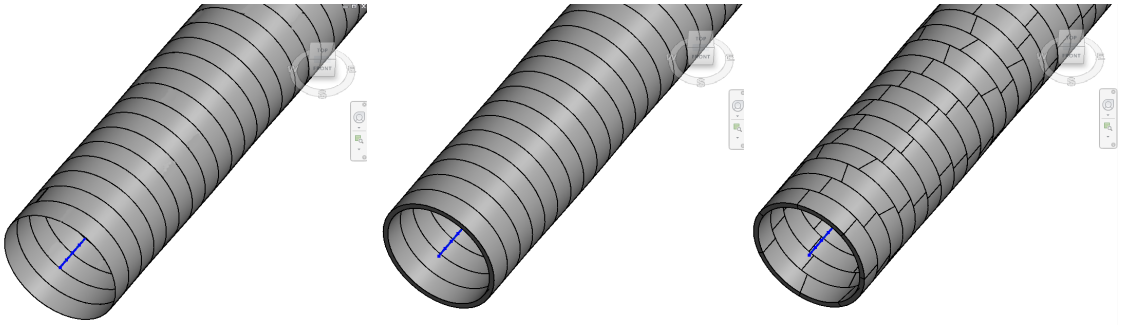


Figure 95: Different levels of detail of lining structure for the tunnel project .

6.2.3 Modelling of the existing infrastructure

The building model for the project includes the above-ground buildings in the area influenced by the tunnel. These buildings, especially their stiffness and mass, have a great influence on the size and shape of the settlement trough. The Auto CAD City model of City infrastructure is shown in Figure 97. As it can be observed, the building mass is represented with quite details, which is an excellent foundation of building the multi-level information model for infrastructure using SATBIM concept.

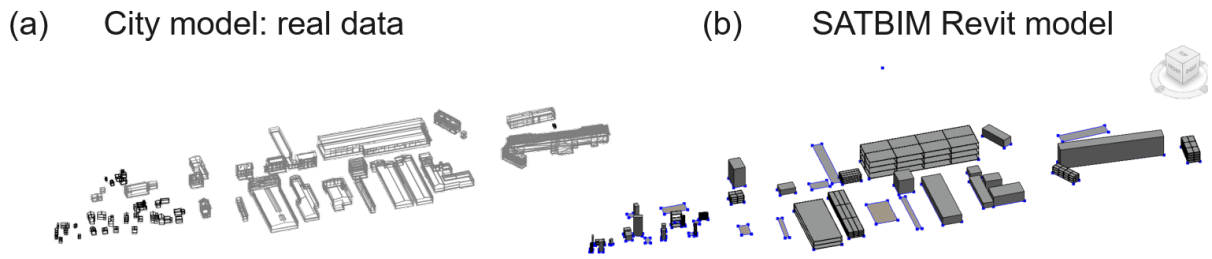


Figure 96: a) City model in Auto Cad; b) Revit model of the existing infrastructure on different LoDs based on the City model

In order to generate Revit models for the infrastructure the footing of all buildings in the area of the tunnel section of interest is imported as a CAD file. This surface is than used to build the information model based on predefined Revit Families as described in in Section 2.5. For each building separately the respective LoD is to be assigned, based on building sensitivity, distance of tunnel from building or other criteria defined. The final output is three-dimensional information model of the infrastructure as shown in Figure 98 b).

Generation of this information model based on Auto CAD City model is accomplished via three steps in Dynamo (see Figure 99):

- importing the CAD file with footing of buildings for the section of interest and assessment of the properties of billings (height, weight);
- selecting the LoD for each individual building;
- generation of buildings based on pre-defined Revit families corresponding building parameters.

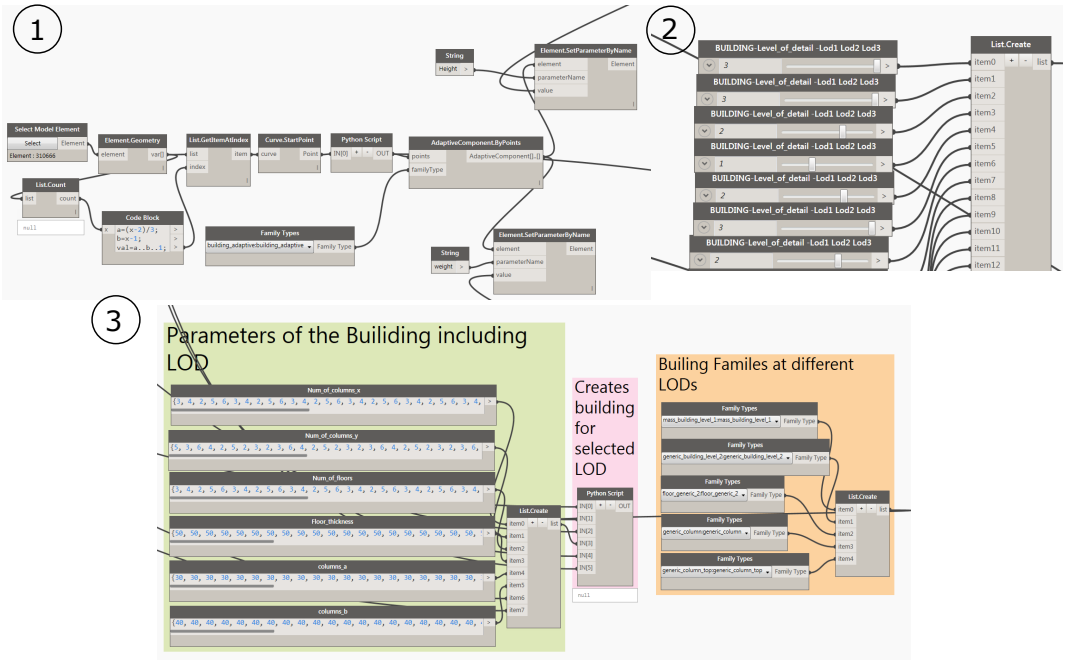


Figure 97: Dynamo node for generation of exiting infrastructure based on City model from Figure 97.

6.2.4 Modelling of the TBM

6.2.5 Complete Model of the tunnel project

Combining all individual tunnel components leads to a complete tunnel information model (see Figures 100 and 101). On Figure 100a we can see the Auto CAD design layout with combined components, while on Figure 100b we can see how the same model is transferred to Revit information model using the predefined Revit families of SATBIM. This model is further going to be used for generation of the simulation models and design assessment of the tunnel project.

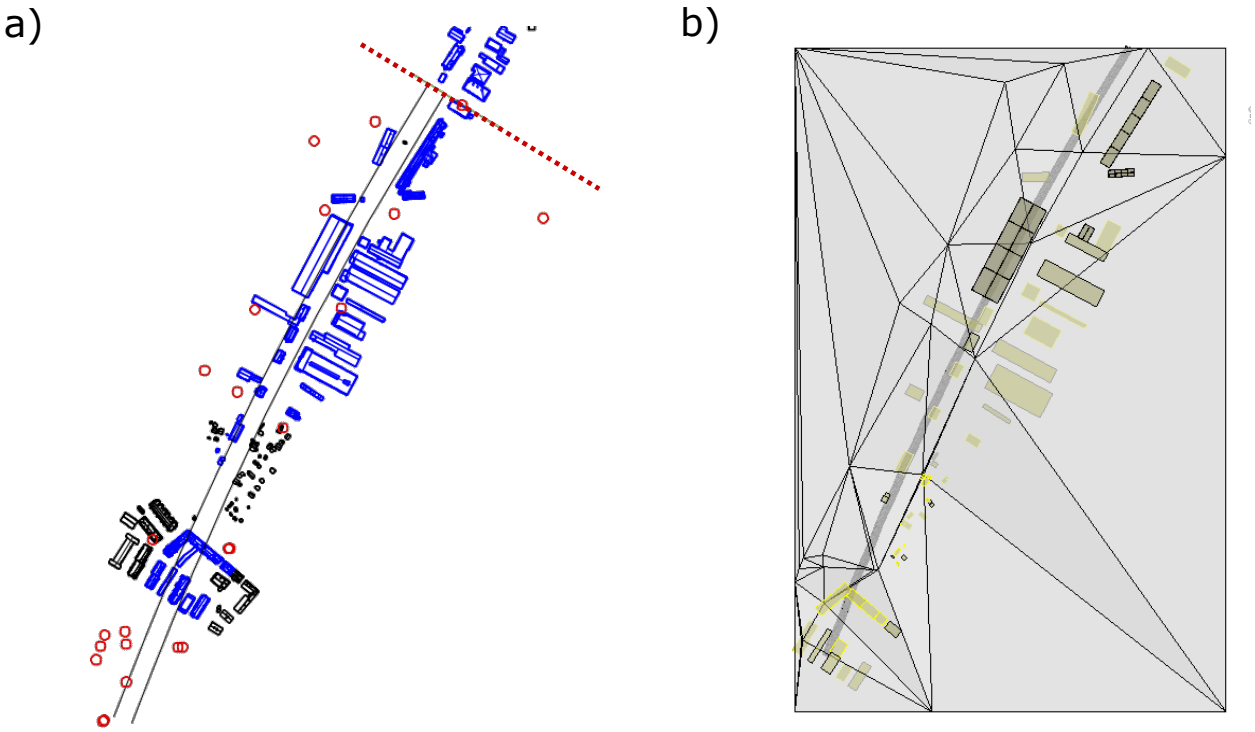


Figure 98: a) Merged Auto CAD model of the tunnel alignment, soil boreholes and city model; b) Revit model for the same section including, soil model, lining structure, infrastructure and TBM

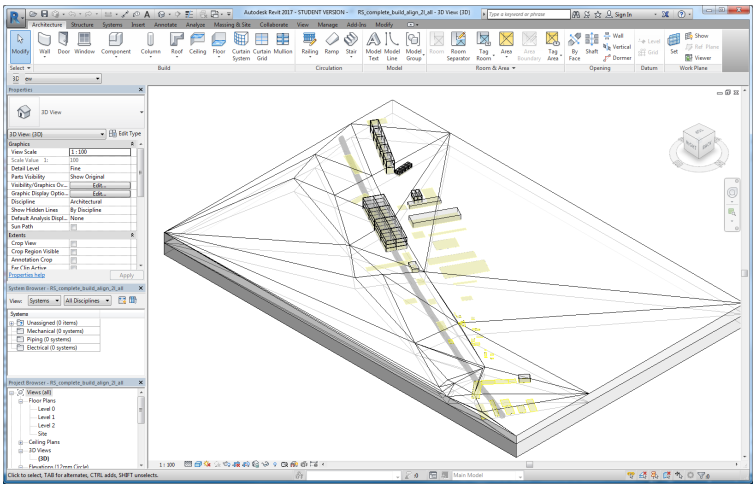


Figure 99: Revit model of the complete system for the tunnel project data.

6.3 Assessment of the design using SATBIM platform

6.3.1 Refinement of the soil geometry geometry

Based on the given borehole data the soil geometry is generated in TIM model as explained in Section6.2.1. The soil soil volumes obtains by interpolation of the soil layer boundaries

indicated in boreholes and the respective ACIS geometry which is a basis for the numerical model is shown in Figure 102a. In this geometry quite irregular triangulation can be observed, with surface element with very sharp angles, which is an extremely difficult basis for good quality of the FE mesh in the phase of numerical model generation. Namely, having in mind that the soil geometry is arbitrary and irregular, the only possible way of re meshing soil volumes is using unstructured mesh formed by tetrahedra elements. In GiD re meshing tools we can asign parameter like size of element, however, if those tetrahedra elements are generated based on bad geometries, this will lead to low quality of mesh, or even meshes which is not possible to use at all for numerical simulations because it contains elements with negative Jacobian. Therefore, the refinement of geometry was necessary in order

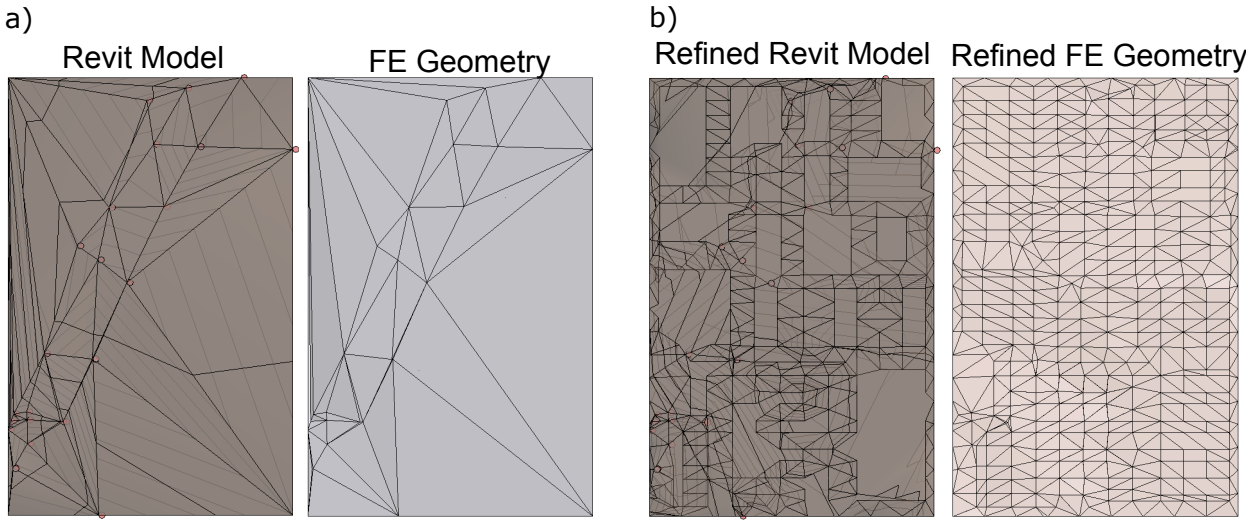


Figure 100: a) Soil Revit model and ASCI geometry based on borehole data; b) refined soil Revit model and ASCI geometry.

to have robust mesh generation and good quality of FE mesh for the numerical analysis. This is done by implementation of additional Dynamo node, which uses the existing surfaces generated based on borehole layer interfaces and adding additional points on surfaces for their refinement. The algorithm is defining potential points on existing surfaces with raster $a \times b$ in $X \times Y$ plane denoted as $point\_set\_1_{ij}$. In the second steps, on each surface k number of points ($point\_set\_2^k_{uv}$) is generated using $Surface.PointAtParameter(u, v)$ method. Finally, the algorithm selects for $point\_set\_2^k_{uv}$ the set of closest points to $point\_set\_1_{ij}$. This final points set, together with initial points set defined by borehole layer interfaces is used to generate surface topology which is a boundary of the soil surface. The refined Revit Model fro the soil and ASCI geometry generated uing described procedure is shown in Figure 102b.

6.3.2 Selection of buildings for the analysis

In Section 6.2.3 the generation of the multi-level representation of the built environment based on the City Model is explained. However, for the project assessment, it is sometimes of interest to select only the particular buildings with higher potential risk of damage (masonry buildings) or with higher impact (historical buildings) to be analysed in details. This has been

enabled with simple selection nodes depicted in Figure 103a, where the *SelectModelElement* is used to select the buildings of interest, the sliders are used to set building LoD and Python script for building generation is adjusted to create Family Object instances only for those selected buildings. Moreover, expiring nodes are dusted to write geometry and semantic files only for the selected buildings. Finally, the `SatBim Modeller` is adjusted to generate numerical models for the multiple (selected) buildings in FE simulation model.

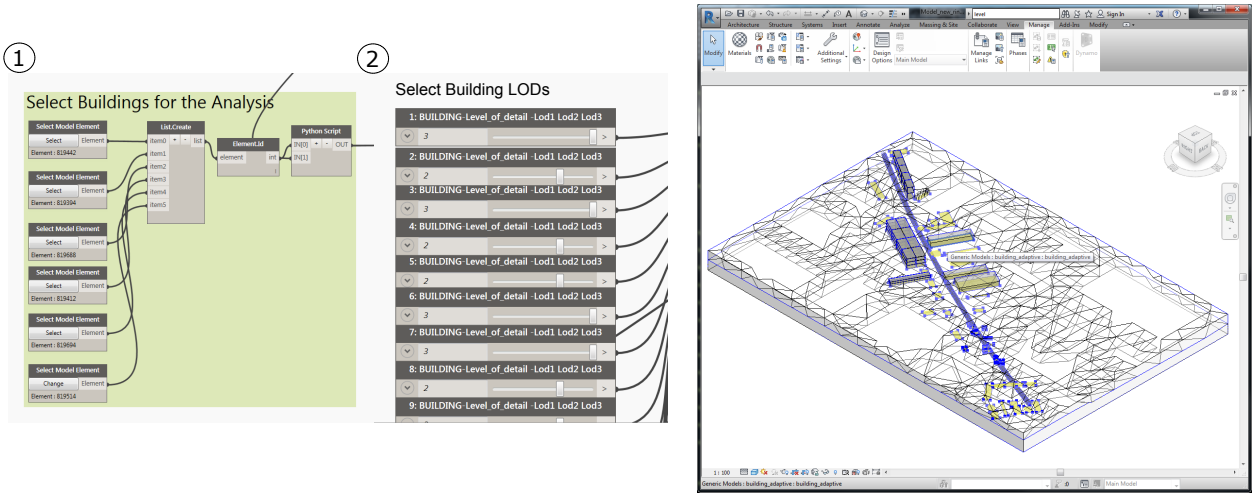


Figure 101: Selection of buildings for further analysis using Dynamo and building footprints imported from Auto CAD model.

6.3.3 Numerical analysis of the soil-structure interaction due to tunnel excavation

In Section 6.1 is described that the two single-track tunnels are to be constructed. Therefore, in next examples, both single track tunnels are going to be analysed separately, with considering overlaying buildings in the vicinity of the respective tunnel track.

In this example, a simple representation of the tunnel construction with the shield tunnelling technology is modelled with the the volume loss method. In this method, the support by lining is modeled without explicit consideration of the lining structure. Yet, the confinement is described with the volume loss coefficient $V_l = (V_0 - V_{def})/V_0 \cdot 100\%$. The more accurate of the volume loss method is used, where after the deconfinement due to soil excavation, the deformed area of the tunnel is continuously calculated at each computation cycle, and deformations of the excavation boundaries are fixed when the volume loss value of the tunnel boundary is reached Do et al. (2014b). For the first tunnel track (green line in Auto CAD model in Figure 104 left) ten buildings along the tunnel alignment are selected for detailed analysis of the soil-structure interaction effects due to tunnelling. Here, the LoDs of analysed buildings are selected arbitrarily, as well as the properties of the structural frames of buildings (number of floors and columns in X and Y direction and dimensions of columns and floors). The soil is represented using Mohr Coulomb soil model, typical for modelling of the non-linear behaviour of the sands. For fully saturated soft soils a two-field finite element formulation is used. Figure 104 shows the transition from Auto CAD model as a

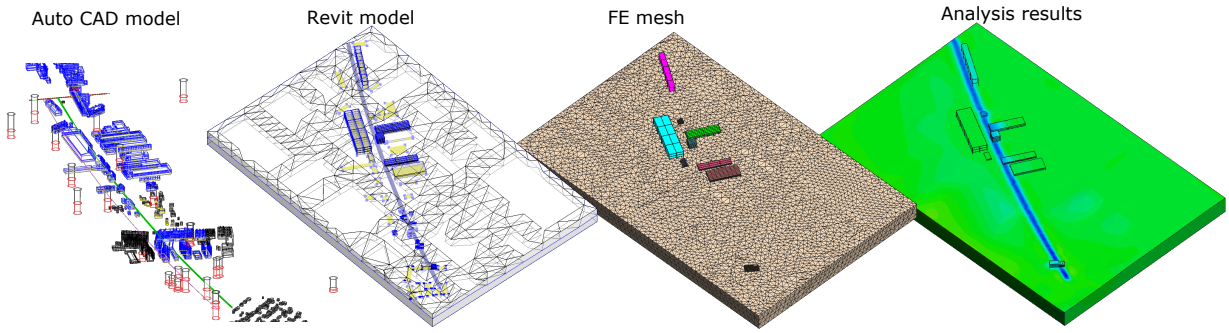


Figure 102: Track one: from Auto Cad model, over TIM and FE geometry up to analysis results.

basis for generation of the information model using TIM (see Section 2) over FE simulation model generated using `SatBim Modeller` (see Section 3.1) up to simulation results calculated using simulation model KRATOS. Form this figure it can be concluded that high consistency between design and information and numerical model is preserved while with the high level of automation the time for generation of such analysis is minimised.

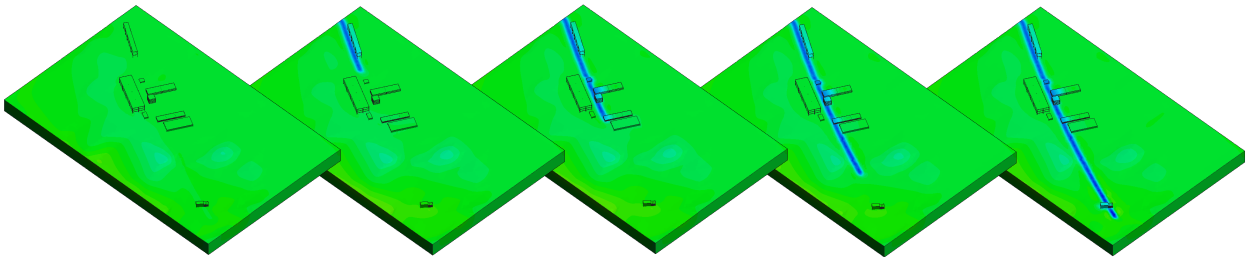


Figure 103: Track one: evolution of vertical displacements and soil-structure interaction effects due to tunnel construction.

Figure 105 shows the evolution of tunnelling induced settlements and effects of the soil-structure interaction in different stages of tunnel construction. Depending on the position of the building w.r.t. tunnel alignment, as well as stiffness of the building depending on material properies and structural frame stiffness, the tunnelling-induced settlements cause different level of deformation and damage on the exiting infrastructure.

For the second tunnel track (magenta line in Auto CAD model in Figure 106 left) twelve buildings along the tunnel alignment are selected for detailed analysis of the soil-structure interaction effects due to tunnelling. Again, building LoDs and properties describing geometry and material of the buildings are selected arbitrarily. Figure 106 shows the transition from Auto CAD model over the information model and FE simulation model up to simulation results of vertical displacements induced by tunnelling. Figure 107 illustrates the evolution of vertical displacements and the soil-structure interaction effects due to tunnel construction

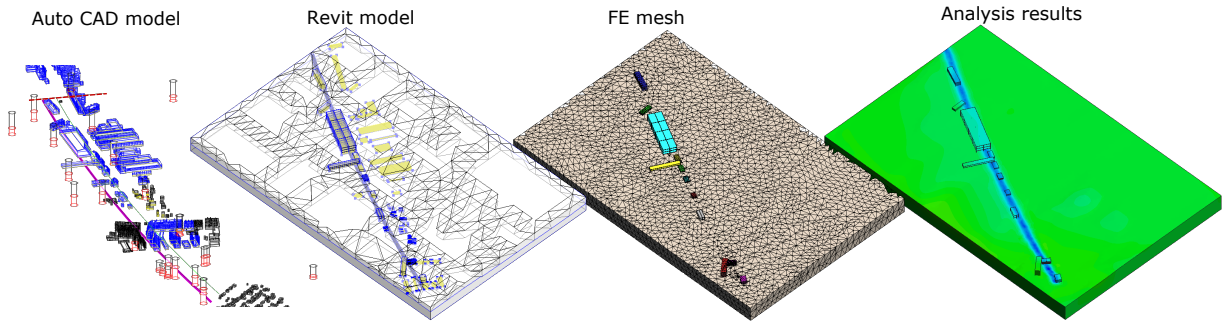


Figure 104: Track two: from Auto Cad model, over TIM and FE geometry up to analysis results.

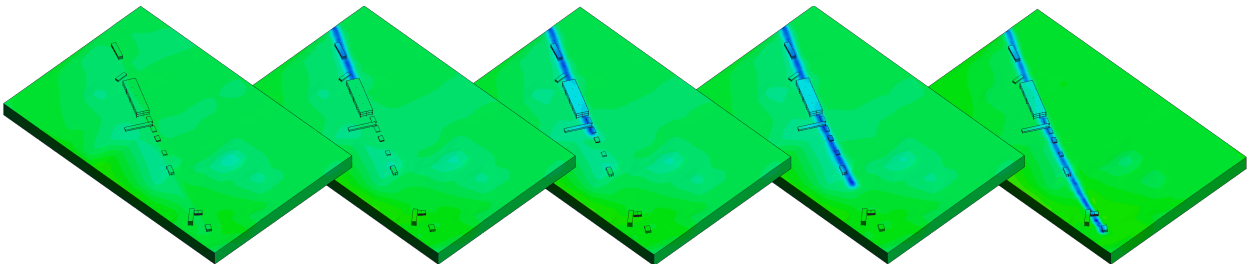


Figure 105: Track two: evolution of vertical displacements and the soil-structure interaction effects due to tunnel construction.

6.3.4 Detailed analysis of the selected tunnel section

For the first estimation of the soil-structure interactions induced by tunnelling an tunnel construction is simulated applying simplest representation of the lining (loD 1) - volume loss method while 30 buildings are represented on medium LoD 2. The simulation model of this approx. 1 km long tunnel section (488 lining rings of 2.0 m length) is characterized with 638700 Degrees of Freedom (DoFs). The average calculation time per step is approx 4 min and the calculation of the model took approx 35 hours. The evolution of the surface settlements in time and soil structure interaction effects for the model shown in Figure 108 is shown in Figure 109.

For more detailed analysis it was not possible to calculate complete 1 km long tunnel section with available computational resources. Therefore, for one of the "critical" sections where the largest influence on existing infrastructure is identified, the bounding box is defined as shown in Figure 110. For the selected tunnel section more detailed analysis of the soil-structure interaction effect are conducted. In this example two models are created, where in the Model 1, the detailed model for all buildings (LoD 3) is adopted, while in Model 2 the LoD of the buildings is optimised based on results shown in Figure 82.

In both models, LoD 2 is selected for the representation of the tunnel lining structure and the TBM. This model accounts for the shield as a deformable body moving through the soil and interacting with the ground through surface-to-surface contact. The tunnel advance is modelled by means of deactivation of soil elements and installation of the lining rings and grouting elements. Tunnelling-induced deformations are controlled by applying the

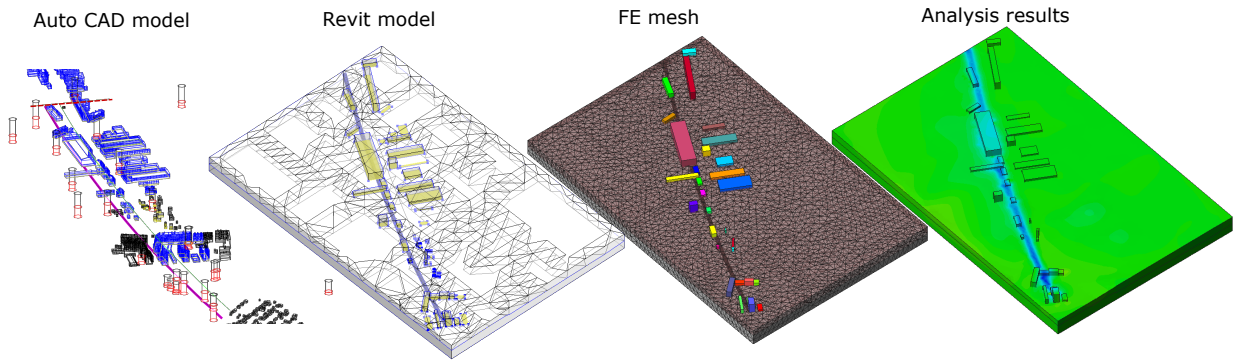


Figure 106: Track two: from Auto Cad model, over TIM and FE geometry up to analysis results for building model LoD 2.

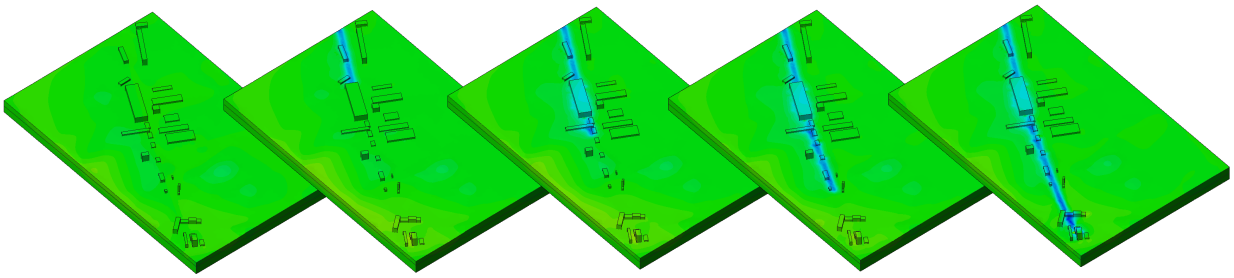


Figure 107: Track two: evolution of vertical displacements and soil-structure interaction effects due to tunnel construction for buildings model LoD 2.

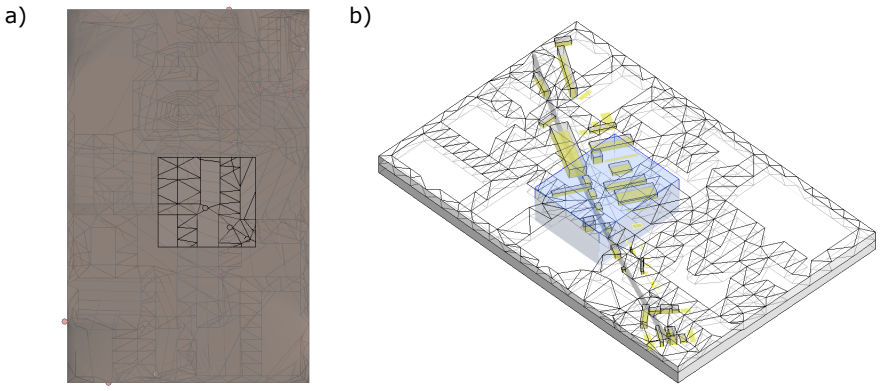


Figure 108: Selection of the tunnel section for detailed analysis. a) Top view of the soil; b) Bounding box for selected section of the TIM for tunnel project.

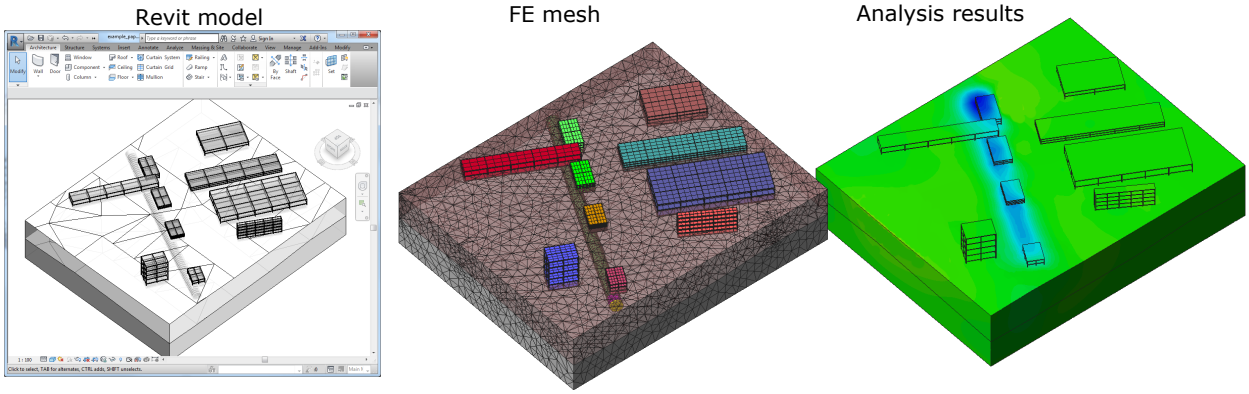


Figure 109: Model 1: from TIM and FE geometry up to analysis results for detailed analysis of tunnel sections with building model LoD 3.

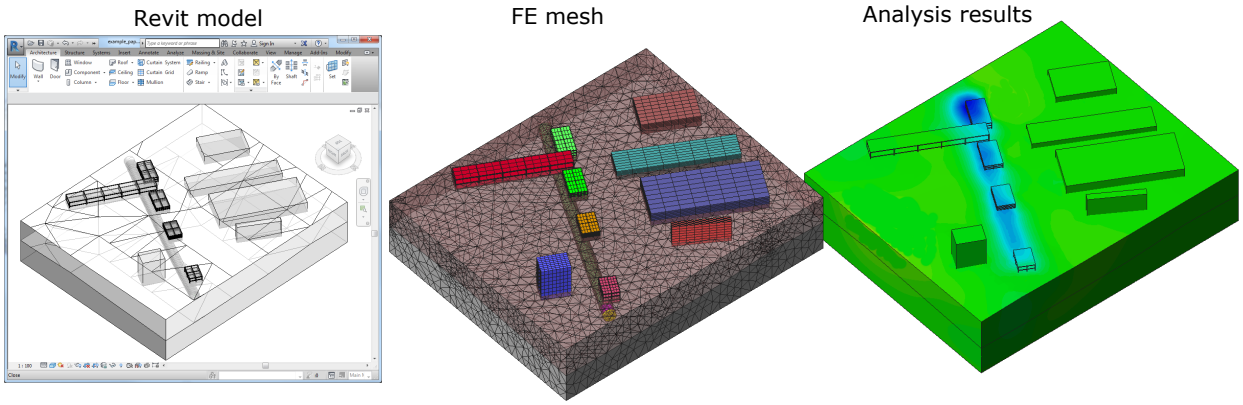


Figure 110: Model 2: from TIM and FE geometry up to analysis results for detailed analysis of tunnel sections with building model with optimized LoD.

face support pressure and the grouting pressure at the tunnel face and in the steering gap, respectively. The Mohr Coulomb model is used as the constitutive relation for effective stress and strain in the soil. The tunnel is constructed with 82 lining rings of 2.0 m length and 10.8 m excavation radius are excavated under approx 20 m of soil overburden.

Considering a spatial discretization of all components (soil, lining, TBM and buildings) the models are finally described with 477,026 and 400,998 DoFs for Model 1- high (LoD3) and Model 2- optimised (LoD2 and LoD3) representation of buildings, respectively. The model size strongly influences the computational costs as shown in Table 4, where the individual as well as the total time for the solution are listed.

Although the size of the model and consequently the computational costs differ significantly, the final output of the numerical analysis is identical for Models 1 and 2 as shown in Figures 111 and 112. This is due to fact that the complexity of the model is optimised without affecting the important, i.e. the influencing features of the model w.r.t. the objective of our analysis, which in this case is tunnelling-induced settlements and interaction with existing buildings.

Computational costs	Building LoD 3	Building LoD optimized
Number of DoFs	477026	400998
Reading input per step [s]	15.31	6.48
Assembly time per step [s]	10.22	5.44
Solve time per step [s]	36.79	32.04
Writing mesh & results per step [s]	1.87	1.45
Total time [min]	308	236

Table 4: Runtime for the solution steps of the model with building LoD 3 (Figure 111) and model with optimized LoD of buildings(Figure 112)

Acknowledgement

This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No [702874]. This support is gratefully acknowledged.

7 Appendix

The Python scripts with algorithms for modelling tunnel excavation process within SATBIM project are given in listings that follow:

Listing 14: Calculation of 3D tunnel alignment

```
rotg=rot*math.pi/180.0
d_alpha_x=alpha*math.cos(rotg)/2.0
d_alpha_z=alpha*math.sin(rotg)/2.0

lx_new.append(lx[0])
lx_new.append(lx[1])
ly_new.append(ly[0])
ly_new.append(ly[1])
lz_new.append(lz[0])
lz_new.append(lz[1])

for i in range (2,len(IN)):
        if list_or[i-2]==0:
                rotg1=60*math.pi/180.0
                rotg2=300*math.pi/180.0
        elif list_or[i-2]==1:
                rotg1=120*math.pi/180.0
                rotg2=0*math.pi/180
        elif list_or[i-2]==2:
                rotg1=180*math.pi/180.0
                rotg2=60*math.pi/180.0
        elif list_or[i-2]==3:
                rotg1=240*math.pi/180.0
                rotg2=120*math.pi/180.0
        elif list_or[i-2]==4:
                rotg1=300*math.pi/180.0
                rotg2=180*math.pi/180.0
        else :
                rotg1=0*math.pi/180.0
                rotg2=240*math.pi/180.0


        rel_x1=dist*math.cos(beta+d_alpha_x+alpha/2.0*math.cos(rotg1))
        *math.cos(gamma+d_alpha_z+alpha/2.0*math.sin(rotg1))
        rel_y1=dist*math.sin(beta+d_alpha_x+alpha/2.0*math.cos(rotg1))
        *math.cos(gamma+d_alpha_z+alpha/2.0*math.sin(rotg1))
        rel_z1=dist*math.sin(gamma+d_alpha_z+alpha/2.0*math.sin(rotg1))
        *math.cos(beta+d_alpha_x+alpha/2.0*math.cos(rotg1))
        x1_n=lx_new[i-1]+ rel_x1
        y1_n=ly_new[i-1]+rel_y1
        z1_n=lz_new[i-1]+rel_z1

        rel_x2=dist*math.cos(beta+d_alpha_x+alpha/2.0*math.cos(rotg2))
        *math.cos(gamma+d_alpha_z+alpha/2.0*math.sin(rotg2))
        rel_y2=dist*math.sin(beta+d_alpha_x+alpha/2.0*math.cos(rotg2))
        *math.cos(gamma+d_alpha_z+alpha/2.0*math.sin(rotg2))
        rel_z2=dist*math.sin(gamma+d_alpha_z+alpha/2.0*math.sin(rotg2))
        *math.cos(beta+d_alpha_x+alpha/2.0*math.cos(rotg2))
        x2_n=lx_new[i-1]+ rel_x2
        y2_n=ly_new[i-1]+rel_y2
        z2_n=lz_new[i-1]+rel_z2

        dist1=  math.sqrt((x1_n-lx[i])*(x1_n-lx[i])+(y1_n-ly[i])
        *(y1_n-ly[i])+(z1_n-lz[i])*(z1_n-lz[i]))
        dist2=  math.sqrt((x2_n-lx[i])*(x2_n-lx[i])+(y2_n-ly[i])
        *(y2_n-ly[i])+(z2_n-lz[i])*(z2_n-lz[i]))
        if dist1<dist2:
                lx_new.append(x1_n)
```

```
                ly_new.append(y1_n)
                lz_new.append(z1_n)
                beta=beta+d_alpha_x+alpha/2.0*math.cos(rotg1)
                gamma= gamma+d_alpha_z+alpha/2.0*math.sin(rotg1)
                d_alpha_x=alpha*math.cos(rotg1)/2.0
                d_alpha_z=alpha*math.sin(rotg1)/2.0
                if list_or[i-2]==5:
                        list_or.append(0)
                else:
                        list_or.append(list_or[i-2]+1)
        else:
                lx_new.append(x2_n)
                ly_new.append(y2_n)
                lz_new.append(z2_n)
                if list_or[i-2]==0:
                        list_or.append(5)
                else:
                        list_or.append(list_or[i-2]-1)

                beta=beta+d_alpha_x+alpha/2.0*math.cos(rotg2)
                gamma= gamma+d_alpha_z+alpha/2.0*math.sin(rotg2)
                d_alpha_x=alpha*math.cos(rotg2)/2.0
                d_alpha_z=alpha*math.sin(rotg2)/2.0
        list_beta.append(beta)
        list_gamma.append(gamma)
print(list_or)

for i in range (0, len(lx_new)):
        point=Point.ByCoordinates(lx_new[i],ly_new[i],lz[i])
        list_points.append(point[i])

rot=60.0*math.pi/180.0
for i in range (1, len(lx_new)):
        dist1=math.sqrt((lx_new[i]-lx_new[i-1])*(lx_new[i]-lx_new[i-1])
        +(ly_new[i]-ly_new[i-1])*(ly_new[i]-ly_new[i-1])+
        (lz_new[i]-lz_new[i-1])*(lz_new[i]-lz_new[i-1]))
        distance.append(dist1)
for i in range (0,len(lx_new)-1):
        f=list_or[i]
        m=lx_new[i]+distance[i]/2.0/(math.cos(alpha/4.0))*math.cos((list_beta[i]+(alpha/4.0)
        *math.cos(rot*(f+3))))*math.cos(list_gamma[i]+(alpha/4.0)*math.sin(rot*(f+3)))
        n=ly_new[i]+distance[i]/2.0/(math.cos(alpha/4.0))*math.sin((list_beta[i]+(alpha/4.0)
        *math.cos(rot*(f+3))))*math.cos(list_gamma[i]+(alpha/4.0)*math.sin(rot*(f+3)))
        k=lz_new[i]+distance[i]/2.0/(math.cos(alpha/4.0))*math.cos((list_beta[i]+(alpha/4.0)
        *math.cos(rot*(f+3))))*math.sin(list_gamma[i]+(alpha/4.0)*math.sin(rot*(f+3)))
        lx_mid.append(m)
        ly_mid.append(n)
        lz_mid.append(k)

for i in range (0, len(lx_mid)):
        point=Point.ByCoordinates(lx_mid[i],ly_mid[i],lz_mid[i])
```

Listing 15: Method 1: Assignment boundary conditions for prescription of the volume loss

```
volume_los=1.0
delta_r=excavation_radius-excavation_radius*math.sqrt(1-volume_los/100)
for step in range (1, number_of_slices):
        time=time+delta_time
        delta_x=(float(alignment[step+1][0])-float(alignment[step][0]))
        delta_y=(float(alignment[step+1][1])-float(alignment[step][1]))
        alpha=math.atan(delta_y/delta_x)
        for node_id in model.layer_nodes_sets['excavation_surface_'+str(step)]:
                node = model.model_part.Nodes[node_id]
                point=[node.X0,node.Y0,node.Z0]
                for i in range (0,3):
                        radius[i]=float(alignment[step+1][i])-point[i]
                theta=math.atan2( radius[1], radius[2] )
```

```
                delta_rz=delta_r*math.cos(theta)
                delta_ry_1=delta_r*math.sin(theta)
                delta_rx=delta_ry_1*math.sin(alpha)
                delta_ry=delta_ry_1*math.cos(alpha)
                #assign displacements
                node.SetSolutionStepValue(DISPLACEMENT_X, delta_rx)
                node.SetSolutionStepValue(DISPLACEMENT_Y, delta_ry)
                node.SetSolutionStepValue(DISPLACEMENT_Z, delta_rz)
```

Listing 16: Method 2: Assignment boundary conditions for prescription of the volume loss

```
def PoligonArea(points):
        area = 0         #Accumulates area in the loop
        j = len(points)-1;  # The last vertex is the 'previous' one to the first
        for i in range (0, len(points)):
                area = area +  abs((points[j][0]+points[i][0]) * (points[j][1]-points[i][1]))
                j = i  #j is previous vertex to i
        return (area/2)
#create list of cross sections along the excavation boundary
excavation_nodes=[[[] for y in range(3)]for z in range(nos)]
for step in range (1, nos):
        for node_id in model.layer_nodes_sets['excavation_surface_'+str(i)]:
                node = model.model_part.Nodes[node_id]
                ....
                        excavation_nodes[step-1][1].append(node_id)
index_node=[[0 for x in range(3)] for y in range(nos)]

##append displacements of the cross section nodels
excavation_nodes_disp=
[[[0 for x in range(2)]for f in range(index_node)] for z in range(nos)]
        ....
for step in range (1, nos+1):
        delta_x=(float(alignment[step][0])-float(alignment[step-1][0]))
        delta_y=(float(alignment[step][1])-float(alignment[step-1][1]))
        alpha=math.atan(delta_y/delta_x)

        list= [[0 for x in range(2)] for y in range(len(mid_nodes[step-1]))]
        m=0
        l=0
        for node_id in excavation_nodes[step-1]:
                node = model.model_part.Nodes[node_id]
                point0=[node.X0,node.Y0,node.Z0]
                point=[node.X,node.Y,node.Z]
                for i in range (0,3):
                        delta_disp[i]=float(point0[i]-point[i])
                x1=delta_disp[0]*math.cos(-alpha)-delta_disp[1]*math.sin(-alpha)
                y1=delta_disp[0]*math.sin(-alpha)+delta_disp[1]*math.cos(-alpha)
                z1=delta_disp[2]
                for i in range (0,3):
                        radius[i]=float(alignment[step][i])-point0[i]
                radius_r=math.sqrt(radius[0]*radius[0]+radius[1]*radius[1]+radius[2]*radius[2])
                # print(radius_r)
                theta=math.atan2( radius[1], radius[2] )
                if radius_r-tol< excavation_radius:
                        x11=excavation_radius*math.cos(theta)-y1
                        y11=excavation_radius*math.sin(theta)-z1
                        list[m][0]=x11
                        list[m][1]=y11
                        m=m+1
        print(list)
        list1=SortList(list)
        # calculate initial volume loss volume loss and check if exceeds prescribed volume loss
        vl0[step-1]=abs(PoligonArea(list1))
#solve the model
model.Solve(time, 0, 0, 0, 0)
model.WriteOutput(time)
```

```
for step in range (1, nos+1):
        time=time+1.0
        #calculate new volume loss
        vl=(vl0[step-1]-abs(PoligonArea(list1)))/vl0[step-1]
        print(vl)
        if vl*100.00>volume_loss:

                #if volume loss exceedded prescribe deipacement from previous step
                if vl*100.00>volume_los:
                        for node_id in excavation_nodes[step-1][k]:
                                node = model.model_part.Nodes[node_id]
                                node.SetSolutionStepValue(DISPLACEMENT_X,
                                excavation_nodes_disp[step-1][node_id][0])
                                node.SetSolutionStepValue(DISPLACEMENT_Y,
                                excavation_nodes_disp[step-1][node_id][1])
                                node.SetSolutionStepValue(DISPLACEMENT_Z,
                                excavation_nodes_disp[step-1][node_id][2])
```

Listing 17: Algorithm for simulation of TBM advance on LoD1

```
#Calculate relative gap between TBM and soil along the shield in corresponding rings
tbm_rings=int(tbm_lenght/round_length)
for i in range (0,tbm_rings):
        delta_r_tbm[i]=excavation_radius-(tbm_r-round_length*(tbm_rings-i-0.5)*tbm_conicity)
        # In excavation steps assign radial displacements for all excavation rings around TBM
        for step in range(1,number_of_excavation_steps+1):
                tbm_index=step+TBM_offset-tbm_rings
                for ring in range(0, tbm_rings):
                        for node in model1.layer_nodes_sets['excavation_surface_'
                        +str(tbm_index+ring)]:
                                #Transform displacements from Local to Global Coord. system
                                #Apply set of boundary conditions on the excavation boundary -
                                 prescribed radial displacements of the TBM-soil gap size
                for node_id in model1.layer_nodes_sets['excavation_surface_'+str(tbm_index-1)]:
                                #Free displacements of the excavation boundary behind the TBM
```

Listing 18: Algorithm for simulation of TBM advance on LoD2

```
for step in range(1,number_of_excavation_steps+1):
        #move TBM
        delta_x=(float(disp_vector[step+1][0])-float(disp_vector[step][0]))
        delta_y=(float(disp_vector[step+1][1])-float(disp_vector[step][1]))
        alpha=math.atan(delta_y/delta_x)
        delta_alpha=alpha-alpha_last
        alpha_last=alpha
        beta=beta+delta_alpha

        for moving_step in range (0,move_steps):
                time = time + move_delta_time
                advance = advance +  one_ring/float(move_steps)
                disp_x = disp_x +  delta_x/float(move_steps)
                disp_y = disp_y +  delta_y/float(move_steps)
                for node_id in shield_nodes:
                        node = model1.model_part.Nodes[node_id]
                        x=node.X0-float(insert_tbm[0])
                        y=node.Y0-float(insert_tbm[1])
                        x1=x*math.cos(beta)-y*math.sin(beta)-node.X0+float(insert_tbm[0])
                        y1=x*math.sin(beta)+y*math.cos(beta)-node.Y0+float(insert_tbm[1])
                        dx=disp_x+x1
                        dy=disp_y+y1
                        if node_id==17851:
                        node.SetSolutionStepValue(DISPLACEMENT_Y, dy)
                        node.SetSolutionStepValue(DISPLACEMENT_X, dx)
```

Listing 19: Assignment of heading face support grouting pressure and TBM resistance pressure

```
#setting pressure boundary condition onto the grouting faces behind the TBM
grouting_face_layer = 'grouting_surface_'+str(grouting_surface_index)
for node in model1.layer_nodes_sets[grouting_face_layer]:
                pressure = grouting_pressure-
                (model1.model_part.Nodes[node].Z)*grouting_gradient
                if( account_for_water == True ):
                        model1.model_part.Nodes[node].Fix(WATER_PRESSURE)
                        model1.model_part.Nodes[node].SetSolutionStepValue
                        (WATER_PRESSURE, pressure)
                model1.model_part.Nodes[node].SetSolutionStepValue(FACE_LOAD_X,-pressure)

#setting loading boundary condition onto the lining faces behind the TBM
simulating the abutment of the hydraulic jacks
for node in model1.layer_nodes_sets['lining_surface_'+str(lining_face_index)]:
        initial_pressure = heading_face_pressure
        depth = model1.model_part.Nodes[node].Z
        pressure = initial_pressure-depth*heading_face_gradient
        model1.model_part.Nodes[node].SetSolutionStepValue(FACE_LOAD_X,-pressure)

#setting pressure boundary condition onto the heading faces in front of the TBM

old_face_layer = 'heading_face_'+str(face_index-1)
for node in model1.node_groups[old_face_layer]:
                model1.model_part.Nodes[node].SetSolutionStepValue(FACE_LOAD_X,0.0)
                if( account_for_water == True ):
                        model1.model_part.Nodes[node].Free( WATER_PRESSURE )
current_face_layer = 'heading_face_'+str(face_index)
for node in model1.node_groups[current_face_layer]:
                pressure = heading_face_pressure-
                (model1.model_part.Nodes[node].Z)*heading_face_gradient
                model1.model_part.Nodes[node].SetSolutionStepValue(FACE_LOAD_X, pressure)
                if( account_for_water == True ):
                        model1.model_part.Nodes[node].Fix( WATER_PRESSURE )
                        model1.model_part.Nodes[node].
                        SetSolutionStepValue(WATER_PRESSURE, pressure)
```

Listing 20: Forward pass for artificial neural network meta model

```
def ReadMetamodel(filename):
        print("reading materials from file: "+str(filename))
        metadata = open( filename, 'r' ).readlines()
        weights=[]
        i= 0
        for a in range (0, len(metadata)):
                line= metadata[i]
                if (a==0):
                        columns = line.split()
                        input_array_size = int(columns[0])
                        i=i+1
                elif (a==1):
                        columns = line.split()
                        hidden_array_size = int(columns[0])
                        i=i+1
                elif (a==2):
                        columns = line.split()
                        output_array_size = int(columns[0])
                        i=i+1
                elif (a==3):
                        columns = line.split()
                        learning_rate = float(columns[0])
                        i=i+1
```

```
                elif (a==4):
                        columns = line.split()
                        for i in range (0, (len (columns)):
                                weights.append(columns[i])
                return ()
```

Listing 21: Forward pass for artificial neural network meta model

```
def ForwardPass(weights,input, hidden_array_size, output_array_size, active_func):
    print(hidden_array_size)
    number_of_input_patterns=len(input)
    input_array_size=len(input[0])
    temp_forw=0
    weight_ih=[[0 for x in xrange(hidden_array_size[0])]
                for x in xrange(input_array_size)]
    weight_ho=[[0 for x in xrange(output_array_size)]
                for x in xrange(hidden_array_size[len(hidden_array_size)-1])]
    hidden_1=[0]*hidden_array_size[0]
    bh1=[0]*hidden_array_size[0]
    if len(hidden_array_size)>1:
        weight_hh1=[[0 for x in xrange(hidden_array_size[1])]
                                for x in xrange(hidden_array_size[0])]
        hidden_2=[0]*hidden_array_size[1]
        bh2=[0]*hidden_array_size[1]
    if len(hidden_array_size)>2:
        weight_hh2=[[0 for x in xrange(hidden_array_size[2])]
                                for x in xrange(hidden_array_size[1])]
        hidden_3=[0]*hidden_array_size[2]
        bh3=[0]*hidden_array_size[2]
    bo=[0]*output_array_size
    output=[[0 for x in xrange(output_array_size)]
                for x in xrange(number_of_input_patterns)]

    for x in range (0, int(input_array_size)):
        for y in range (0, int(hidden_array_size[0])):
            weight_ih[x][y]=float(weights[hidden_array_size[0]*x+y])
    m=input_array_size*hidden_array_size[0]

    if (len(hidden_array_size)==1):
        for x in range (0, hidden_array_size[0]):
            for y in range (0, output_array_size):

                weight_ho[x][y]=float(weights[m+x+y])

        m=m+hidden_array_size[0]*output_array_size

        for y in range (0, int(hidden_array_size[0])):
                bh1[y]=float(weights[m+y]   )
        m=m+hidden_array_size[0]
        for y in range (0, output_array_size):
                bo[y]=float(weights[m+y]   )

    elif (len(hidden_array_size)==2):
        for x in range (0, hidden_array_size[0]):
            for y in range (0, hidden_array_size[1]):

                weight_hh1[x][y]=float(weights[m+x*hidden_array_size[1]+y])

        m=m+hidden_array_size[1]*hidden_array_size[0]
        for x in range (0, hidden_array_size[1]):
            for y in range (0, output_array_size):

                weight_ho[x][y]=float(weights[m+x+y])

        m=m+hidden_array_size[1]*output_array_size
```

```
        for y in range (0, int(hidden_array_size[0])):
                bh1[y]=float(weights[m+y]   )
        m=m+hidden_array_size[0]
        for y in range (0, int(hidden_array_size[1])):
                bh2[y]=float(weights[m+y]   )
        m=m+hidden_array_size[1]
        for y in range (0, output_array_size):
                bo[y]=float(weights[m+y]   )

    elif (len(hidden_array_size)==3):
        for x in range (0, hidden_array_size[0]):
            for y in range (0, hidden_array_size[1]):

                weight_hh1[x][y]=float(weights[m+x*hidden_array_size[1]+y])

        m=m+hidden_array_size[1]*hidden_array_size[0]
        for x in range (0, hidden_array_size[1]):
            for y in range (0, hidden_array_size[2]):

                weight_hh2[x][y]=float(weights[m+x*hidden_array_size[2]+y])
        m=m+hidden_array_size[1]*hidden_array_size[2]

        for x in range (0, hidden_array_size[2]):
            for y in range (0, output_array_size):

                weight_ho[x][y]=float(weights[m+x+y])

        m=m+hidden_array_size[2]*output_array_size

        for y in range (0, int(hidden_array_size[0])):
                bh1[y]=float(weights[m+y]   )
        m=m+hidden_array_size[0]
        for y in range (0, int(hidden_array_size[1])):
                bh2[y]=float(weights[m+y]   )
        m=m+hidden_array_size[1]

        for y in range (0, int(hidden_array_size[2])):
                bh3[y]=float(weights[m+y]   )
        m=m+hidden_array_size[2]
        for y in range (0, output_array_size):
                bo[y]=float(weights[m+y]   )



    for x in range (0, number_of_input_patterns):
    ## Evaluate input -> hidden 1 layer
        for z in range (0, hidden_array_size[0] ):
                for y in range (0, input_array_size):
                        temp_forw = temp_forw + (input[x][y] * weight_ih[y][z])
                temp_forw=temp_forw+bh1[z]

                if active_func=='relu' :
                  if temp_forw>0:
                      hidden_1[z] = temp_forw
                  else:
                      hidden_1[z] = 0
                elif active_func=='logistic' :
                   hidden_1[z] = (1.0 / (1.0 + exp(-1.0 * temp_forw)))
                temp_forw = 0.0

     ## Evaluate hidden1 -> hidden 2   layer
        if (len(hidden_array_size)>1)  :

                for z in range (0, hidden_array_size[1] ):
                        for y in range (0, hidden_array_size[0]):
                                temp_forw = temp_forw + (hidden_1[y] * weight_hh1[y][z])
                        temp_forw=temp_forw+bh2[z]

                        if active_func=='relu' :
                          if temp_forw>0:
```

```
                        hidden_2[z] = temp_forw
                    else:
                        hidden_2[z] = 0
                elif active_func=='logistic' :
                    hidden_2[z] = (1.0 / (1.0 + exp(-1.0 * temp_forw)))
                temp_forw = 0.0

    ## Evaluate hidden2 -> hidden 3  layer
        if (len(hidden_array_size)>2)  :

            for z in range (0, hidden_array_size[2] ):
                for y in range (0, hidden_array_size[1]):
                        temp_forw = temp_forw + (hidden_2[y] * weight_hh2[y][z])
                temp_forw=temp_forw+bh3[z]

                if active_func=='relu' :
                    if temp_forw>0:
                        hidden_3[z] = temp_forw
                    else:
                        hidden_3[z] = 0
                elif active_func=='logistic' :

                    hidden_3[z] = (1.0 / (1.0 + exp(-1.0 * temp_forw)))
                temp_forw = 0.0

    ## Evaluate hidden -> output layer
        for z in range (0,output_array_size ):
            for y in range (0,hidden_array_size[len(hidden_array_size)-1] ):
                if (len(hidden_array_size)==1)  :
                    temp_forw += (hidden_1[y] * weight_ho[y][z])
                elif (len(hidden_array_size)==2)  :
                    temp_forw += (hidden_2[y] * weight_ho[y][z])
                elif (len(hidden_array_size)==3)  :

                    temp_forw += (hidden_3[y] * weight_ho[y][z])
                else  :
                    prrint("error_-_too_much_hidden_layers")

            temp_forw=temp_forw+bo[z]
            if active_func=='relu' :
                if temp_forw>0:
                    output[x][z] = temp_forw
                else:
                    output[x][z] = 0.0
            elif active_func=='logistic' :
                output[x][z] = (1.0 / (1.0 + exp(-1.0 * (temp_forw ))))

    return output
```

Listing 22: Method 1: Calculation of the apsolute mean of elementary effect

```
def ComputeVariance(test_data, n_tp, ts):
    S=[0 for x in xrange (0,ts)]
    S_tot=[0 for x in xrange (0,ts)]
    for j in range (0,(n_tp-1)):
        for i in range (0, ts):
            S[i]= S[i] + math.sqrt(math.pow((test_data[(j+1)*ts+i][0]
                                             - test_data[i][0])/((j+1)*0.8/(n_tp)) , 2))
    for i in range (0, ts):
        S_tot[i] = S[i]/(n_tp-1)
    return S_tot
```

Listing 23: Method 1: Calculation of the standard deviation of elementary effect

```
def ComputeDeviation(test_data, n_tp, ts):
    S=[0 for x in xrange (0,ts)]
    Sigma=[0 for x in xrange (0,ts)]
    Sigma_tot=[0 for x in xrange (0,ts)]
    Mean=[0 for x in xrange (0,ts)]

    for j in range (0,(n_tp-1)):
        for i in range (0, ts):
            S[i]=S[i]+ (test_data[(j+1)*ts+i][0] - test_data[i][0])/((j+1)*0.8/(n_tp))
    for i in range (0, ts):
        Mean[i] = S[i]/(n_tp-1)

    S=[0 for x in xrange (0,ts)]
    for j in range (0,(n_tp-1)):
        for i in range (0, ts):
            S[i]= (test_data[(j+1)*ts+i][0] - test_data[i][0])/((j+1)*0.8/(n_tp))
            Sigma [i] = Sigma [i] + math.sqrt(math.pow((S[i]-Mean[i]),2))

    for i in range (0,ts):
        Sigma_tot[i] = math.sqrt(Sigma[i]/(n_tp-1))
    return Sigma_tot
```

References

(2016). Herrenknecht. https://www.herrenknecht.com/en/innovation/know–how/segmental–lining.html. Accessed: 2016–09–20.

Aldiss, D., M. Blac, D. Entwisle, D. Page, and R.L.Terrington (2012). Benefits of a 3D geological model for major tunnelling works: An example from Farringdon, east-central London, UK. *Quarterly Journal of Engineering Geology and Hydrogeology 45(4)*, 405–414.

Arnau, O. and C. Molins (2012). Three dimensional structural response of segmental tunnel linings. *Engineering Structures 44*(0), 210 – 221.

AUTODESK (2017). *Autodesk Revit*. AUTODESK. http://www.autodesk.co.uk/products/revit-family/.

Been, K. and M. Jefferies (1985). A state parameter for sands. *Geotechnique 35*(2), 99–112.

Biljecki, F., H. Ledoux, and J. Stoter (2016). An improved LOD specification for 3D building models. *Computers, Environment and Urban Systems 59*, 25 – 37.

Borrmann, A., M. Flurl, J. R. Jubierre, R.-P. Mundani, and E. Rank (2014). Synchronous collaborative tunnel design based on consistency-preserving multi-scale models. *Advanced Engineering Informatics 28*(4), 499 – 517.

Bui, G., J. Stascheit, and G. Meschke (2013). A Parallel Block Preconditioner for Coupled Simulations of Partially Saturated Soils in Finite Element Analyses. In B. Topping and P. Iványi (Eds.), *The Third International Conference on Parallel, Distributed, Grid and Cloud Computing for Engineering*, pp. paper 24. Civil-Comp.

Bui, H. G. and G. Meschke (2017). A parallelization strategy for hydro-mechanically coupled simulations in mechanized tunneling. In *EURO:TUN 2017*.

Burland, J. and C. Wroth (1975). Settlement of Buildings and Associated Damage. In *Conference on Settlement of Structures*.

Cavalaro, S., C. Blom, J. Walraven, and A. Aguado (2011). Structural analysis of contact deficiencies in segmented lining. *Tunnelling and Underground Space Technology 26*(6), 734 – 749.

CIMNE International Center for Numerical Methods in Engineering (2016). *GiD: the personal pre- and postprocessor*. CIMNE International Center for Numerical Methods in Engineering.

Dadvand, P., R. Rossi, and E. Oñate (2010). An Object-oriented Environment for Developing Finite Element Codes for Multi-disciplinary Applications. *Archives of Computational Methods in Engineering 17*, 253–297.

Dias, D., R. Kastner, and M. Maghazi (2000). Three dimensional simulation of slurry shield tunnelling. In O. Kusakabe, K. Fujita, and Y. Miyazaki (Eds.), *Geotechnical Aspects of Underground Construction in Soft Ground, Tokyo 1999*, Rotterdam, pp. 351–356. Balkema.

Do, N., D. Dias, P. Oreste, and I. Djeran-Maigre (2014a). Three-dimensional numerical simulation for mechanized tunnelling in soft ground: the influence of the joint pattern. *Acta Geotechnica 9*(4), 673–694.

Do, N.-A., D. Dias, P. Oreste, and I. Djeran-Maigre (2014b). 2D Tunnel Numerical Investigation: The Influence of the Simplified Excavation Method on Tunnel Behaviour. *Geotechnical and Geological Engineering 32*(1), 43–58.

Döllner, J. and H. Buchholz (2005). Continuous level-of-detail modeling of buildings in 3D city models. In *Proceedings of the 13th Annual ACM International Workshop on Geographic Information Systems*, New York.

Falgout, R. D. and U. M. Yang (2002). hypre: a Library of High Performance Preconditioners. In *Preconditioners,? Lecture Notes in Computer Science*, pp. 632–641.

Forberg, A. (2007). Generalization of 3D building data based on a scale-space approach. *Journal of Photogrammetry and Remote Sensing 62*(2), 104 – 111. Including Special Section:Young Author Award.

Founta, V., J.Ninic, A. J. Whittle, G. Meschke, and J. Stascheit (2013, April). Numerical Simulation of Ground Movements Due To EPB Tunnelling in Clay. In *EURO:TUN 2013, 3rd International Conference on Computational Methods in Tunneling and Subsurface Engineering*, Ruhr University Bochum, pp. 97–108.

Hefny, A. and H. Chua (2006). An investigation into the behaviour of jointed tunnel lining. *Tunnelling and Underground Space Technology 21*(3 – 4), 428. Safety in the Underground Space – Proceedings of the ITA – AITES 2006 World Tunnel Congress and 32nd ITA General Assembly.

Hejazi, Y., D. Dias, and R. Kastner (2008). Impact of constitutive models on the numerical analysis of underground constructions. *Acta Geotechnica 3*(4), 251–258.

International Center for Numerical Methods in Engineering (CIMNE) (2014, January). *Kratos – multi-physics.* (Website ed.). International Center for Numerical Methods in Engineering (CIMNE).

Jezyk, M. and the Dynamo Development team at Autodesk (2016). *The Dynamo Primer. V.1.1.0.* AUTODESK. http://dynamoprimer.com/.

Karypis, G. and V. Kumar (1998). A Fast and High Quality Multilevel Scheme for Partitioning Irregular Graphs. *SIAM J. Sci. Comput. 20*, 359–392.

Kasper, T. (2005). *Finite Elemente Simulation maschineller Tunnelvortriebe in wassergesättigtem Lockergestein.* Ph. D. thesis, Lehrstuhl für Statik und Dynamik, Ruhr-Universität Bochum.

Kasper, T. and G. Meschke (2004). A 3D finite element model for TBM tunneling in soft ground. *International Journal for Numerical and Analytical Methods in Geomechanics 28*, 1441–1460.

Kasper, T. and G. Meschke (2006). On the influence of face pressure, grouting pressure and TBM design in soft ground tunnelling. *Tunnelling and Underground Space Technology 21*(2), 160–171.

Kennedy, J. and R. C. Eberhart (1995, November-December). Particle swarm optimization. In I. Press (Ed.), *Proceedings of the IEEE International Conference on Neural Networks*, Piscataway, NJ, USA, pp. 1942 – 1948.

Kennedy, J. and W. M. Spears (1998). Matching Algorithms to Problems: An Experimental Test of the Particle Swarm and Some Genetic Algorithms on the Multimodal Problem Generator. In *Proceedings of the IEEE Congress on Evolutionary Computation*.

Kolbe, T. H. and G. Groeger (2004). Towards unified 3D city models. In *Joint ISPRS Commission IV Workshop on Challenges in Geospatial Analysis, Integration and Visualization II*, Stuttgart, Germany.

Kolymbas, D. (1998). *Geotechnik - Tunnelbau und Tunnelmechanik*. Springer.

Koyama, Y. (2003). Present status and technology of shield tunneling method in Japan. *Tunnelling and Underground Space Technology 18*(2-3), 145–159.

Lambrughi, A., L. M. Rodríguez, and R. Castellanza (2012). Development and validation of a 3D numerical model for TBM–EPB mechanised excavations. *Computers and Geotechnics 40*(0), 97 – 113.

Laursen, T. (2002). *Computational Contact and Impact Mechanics*. Springer, Berlin-Heidelberg.

Maidl, B., M. Herrenknecht, U. Maidl, and G. Wehrmeyer (2012). *Mechanised Shield Tunnelling*. Ernst und Sohn.

McCulloch, W. S. and W. Pitts (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics 5*(4), 115–133.

Meschke, G., C. Kropik, and H. Mang (1996). Numerical analyses of tunnel linings by means of a viscoplastic material model for shotcrete. *International Journal for Numerical Methods in Engineering 39*, 3145–3162.

Meschke, G., J. Ninic, J. Stascheit, and A. Alsahly (2013). Parallelized Computational Modeling of Pile-Soil Interactions in Mechanized Tunneling. *Engineering Structures 47*, 35 – 44. Invited paper for Special Issue Computational Mechanics,.

Miro, S., D. Hartmann, and T. Schanz (2014). Global sensitivity analysis for subsoil parameter estimation in mechanized tunneling. *Computers and Geotechnics 56*, 80–88.

Morris, M. (1991). Factorial sampling plans for preliminary computational experiments. *Technometrics 33*, 161 – 174.

Nagel, F. (2009). *Numerical modelling of partially saturated soil and simulation of shield supported tunnel advance*. Ph. D. thesis, Ruhr University Bochum.

Nagel, F. and G. Meschke (2010). An elasto-plastic three phase model for partially saturated soil for the finite element simulation of compressed air support in tunnelling. *International Journal for Numerical and Analytical Methods in Geomechanics 34*, 605–625. doi:10.1002/nag.828.

Nagel, F., J. Stascheit, and G. Meschke (2010). Process-oriented numerical simulation of shield tunneling in soft soils. *Geomechanics and Tunnelling 3*(3), 268–282.

Ninić, J. and G. Meschke (2015, January). Model Update and Real-Time Steering of Tunnel Boring Machines using Simulation-Based Meta Models. *Tunnelling and Underground Space Technology 45*, 138 – 152. Online.

Ninić, J., J. Stascheit, and G. Meschke (2011). Prediction of Tunnelling Induced Settlements using Simulation-Based Artificial Neural Networks. In Y. Tsompanakis and B. Topping (Eds.), *Proceedings of the Second International Conference on Soft Computing Technology in Civil, Structural and Environmental Engineering*, Chania, Greece, pp. paper 26. Civil-Comp Press, Stirlingshire. CD-ROM.

Panet, M. and A. Guenot (1982). Analysis of convergence behind the face of a tunnel. In *Tunnelling 82. IMM*, London, pp. 197?203.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research 12*, 2825–2830.

Potts, D. and T. Addenbrooke (1997). A structure's influence on tunnelling-induced ground movements. *Proceedings of the ICE-Geotechnical Engineering 125*(2), 109–125.

Potts, D. M. and L. Zdravkovic (1999). *Finite Element Analysis in Geotechnical Engineering*. Thomas Telford.

Powell, D., O. Sigl, and J. Beveridge (1997). Heathrow-Express-design and performance of platform tunnels at Terminal 4. In *Tunnelling?97. IMM*, London, pp. 565?593.

Roscoe, K. and J. Burland (1968). On the generalized stress strain behavior of ẃetćlay. In J. Heyman and F. A. Leckie (Eds.), *Engineering Plasticity*, Cambridge, pp. 535–609. Cambridge University Press.

Rosenblatt, F. (1962). *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*. Washingotn, D. C., Spartan Books.

Rowe, R., K. Lo, and G. Kack (1983). A Method of Estimating Surface Settlement Above Tunnels Constructed in Soft Ground. *Canadian Geotechnical Journal 20*, 11–22.

Rumelhart, D. E., G. E. Hinton, and R. J. Williams (1986). Learning representations by back-propagating errors. *Nature 323*, 533 – 536.

Saltelli, A., M. Ratto, T. Andres, F. Campolongo, J. Cariboni, D. Gatelli, M. Saisana, and S. Tarantola (2008). *Global sensitivity analysis. The primer*. John Wiley & Sons.

Schaufler, A., C. Becker, and H. Steeb (2013). Infiltration processes in cohesionless soils. *ZAMM - Journal of Applied Mathematics and Mechanics / Zeitschrift fr Angewandte Mathematik und Mechanik 93*(2-3), 138–146.

Schindler, S. and P. Mark (2013). Evaluation of Building Stiffness in the Risk-assessment of Structures Affected by Settlements. In *Proc. 3rd Int. Conf. on Comp. Meth. in Tunneling and Subsurface Engineering - EURO:TUN 2013, RUB*, Bochum, Germany, pp. 477–486.

Sheng, D., S. Sloan, and H. Yu (2000). Aspects of finite element implementation of critical state models. *Computational mechanics 26*(2), 185–196.

Shi, J., J. A. R. Ortigao, and J. Bai (1998). Modular Neural Networks for Predicting Settlements During Tunneling. *Journal of Geotechnical and Geoenvironmental Engineering 124*(5), 389–395.

Stascheit, J. (2010). *Parallelisation and model generation methods for large-scale simulations of shield tunnelling processes*. Ph. D. thesis, Ruhr-Universität Bochum.

Swoboda, G. (1979). Finite Element Analysis of the New Austrian Tunnelling Method (NATM). In *Third International Conference on Numerical Methods in Geomechanics*.

Teachavorasinskun, S. and T. Chub-Uppakarn (2010). Influence of segmental joints on tunnel lining. *Tunnelling and Underground Space Technology 25*(4), 490 – 494.

van Oosterom, P. and V. Schenkelaars (1995). The Development of an Interactive Multi-Scale GIS. *International Journal of Geographical Information Systems 9*(5), 489–507.

Vapnik, V. (2000). *The nature of statistical learning theory*. Springer-Verlag.

Verruijt, A. (2012). *Soil Mechanics*. Delft Academic Press. ISBN 9789065621634.

Vesic, A. B. (1963). Beams on Elastic Subgrade and the Winkler?s Hypothesis. In *Proceedings of 5th International Conference of Soil Mechanics*, pp. 845 – 850.

Widrow, B. and M. Hoff (1960). Adaptive Switching Circuits. *IRE WESCON Convention Record 4*, 96–104.

Xie, J., L. Zhang, J. Li, H. Wang, and L. Yang (2012). Automatic simplification and visualization of 3D urban building models. *International Journal of Applied Earth Observation and Geoinformation 18*(1), 222–231. cited By 5.

Yu, H. (1998). CASM: A unified state parameter model for clay and sand. *International Journal for Numerical and Analytical Methods in Geomechanics 48*, 773–778.

Zobl, F. and R. Marschallinger (2008). Subsurface Geo Building Information Modelling GeoBIM. *GEO Informatics 11*.