Geotechnical Simulations with Moose

– DRAFT –

This whitepaper is a collection of information and code for geotechnical simulations using the open source finite element framework 'Moose'.

Jörg Meier

Gruner AG, Geotechnical Engineering
Basel, Switzerland

github.com/jmeier/geotechnical-simulations-with-moose
03.02.2025

Contents

1 Introduction **3**

2 General Remarks **4**
2.1 Sign convention . . . 4

3 Model Configuration **5**
3.1 Strain calculation . . . 5
3.2 Stress calculation . . . 5
3.3 Fixation of the model boundaries . . . 5

4 Geometry **6**
4.1 General considerations on geometry . . . 6
4.2 'Subdomains' resp. 'Blocks' . . . 6
4.3 Getting the mesh into Moose . . . 7
4.4 Lower dimensional blocks in MSH files . . . 7
4.5 Choice of element types . . . 7
4.6 Groups of geometrical objects . . . 8

5 Modelling of geometric entities, structures, loads, etc. **10**
5.1 Overview . . . 10
5.2 Entities . . . 10
5.2.1 Volumes . . . 10
5.2.2 FixedEndSprings . . . 10
5.2.3 Springs between nodes (NodeToNodeAnchors) . . . 10
5.2.4 Beams (line element) . . . 10
5.2.5 Shell (surface element) . . . 11
5.2.6 Interfaces (surface element) . . . 11
5.2.7 Contact . . . 11
5.2.8 Pore Pressure Boundaries . . . 11
5.3 Groundwater . . . 12

6 Initial Conditions **13**
6.1 Initial conditions for Moose objects . . . 13
6.2 Gravitation and initial stress state . . . 13
6.2.1 Gravity . . . 13
6.2.2 Initial stress state using ComputeEigenstrainFromInitialStress . . . 14
6.2.3 Initial stress state for anisotropic stresses . . . 14
6.2.4 Checking the initial stress state . . . 14

7 Construction Stages **17**
7.1 General comments on construction stages . . . 17
7.2 Time steps without changes . . . 18
7.3 Abrupt versus gradual changes . . . 18

7.4 Activating and deactivating FE-elements 18
7.5 Replacing FE-elements 18
7.6 Adjusting external loads 18
7.7 Adjusting prestress 19

8 Material Models for Soil and Rock 20
8.1 Preliminary remarks on the material models 20
8.2 NEML2 20
8.3 Mohr-Coulomb (linear-elastic ideal-plastic) 20

9 Proven Patterns 21
9.1 Partial input files 21
9.2 Physical units 21
9.3 Moose code plugins 23
9.4 Convergence criteria and variable scaling 23
9.4.1 Nonlinear and (nested) linear solves 23
9.4.2 Residuals 23
9.4.3 Variable scaling 25
9.4.4 Tolerances and their default values 26

10 Postprocessing 27
10.1 Paraview 27
10.2 Blender using the BVtkNodes addon 27
10.3 Directly accessing ExodusII files 27

11 Outlook 28

12 References 29

Appendices 30

Appendix A Stiffness Reduction 31
A.1 Problem statement 31
A.2 Analytical solution 31
A.3 Numerical solution with Moose 32

Appendix B Bended Plate 33
B.1 Problem statement 33
B.2 Analytical solution 33
B.3 Moose 34
B.4 Plaxis 3D 34

Appendix C Fully Saturated Undrained Compression 35
C.1 Problem statement 35
C.2 Analytical solution 35
C.3 Numerical solution with Moose 37
C.4 Numerical solution with Plaxis 3D 38

Appendix D Bi-axial Shear Test 40
D.1 Problem statement 40
D.2 Analytical solution 40
D.3 Moose 40
D.4 Plaxis 3D 40

Chapter 1

Introduction

By enabling massively parallel multiphysics simulation the open-source, finite element framework Moose (mooseframework.inl.gov, [2]) can also be used for geotechnical design practice. This document aims to collect information and techniques that are suitable for creating such geotechnical simulations with Moose.

In the geotechnical planning practice, finite element models are often used for deformation prognosis. Furthermore, such FE models can help to identify complex failure mechanisms. The requirements for these models are accordingly a sufficiently good representation of the geotechnical structure including the subsoil, the groundwater conditions, pre-existing structures and similar. These FE models simulate the construction process over time. As this design process is generally intended to take place in a controlled and quasi-static form, the associated simulations are usually also transient and quasi-static. Therefore, scope and main focus of this document are transient quasi-static simulations.

> This document is primarily intended as a collection of information rather than a textbook or step-by-step guide. As such, it is intended to be a 'living' document that will be continually edited and updated.

Chapter 2

General Remarks

2.1 Sign convention

In the 'SolidMechanics' module and the 'PorousFlow' module the sign convention is as follows:

- Positive strain is extensional, which (usually) results from a positive (tensile) stress.
- Negative strain is compressional, which (usually) results from a negative (compressive) stress.
- Compressive pore-pressures (e.g. below a free water table) have a positive sign.
- Suction is represented by pore-pressures having a negative sign.

Chapter 3

Model Configuration

3.1 Strain calculation

In the 'SolidMechanics' module, Moose supports three different types of strain calculation:

- Small Linearized Total Strain
- Incremental Small Strains
- Finite Large Strains

Listing 3.1: Setting up incremental small strains within the Physics/SolidMechanics block

```
[Physics]
  [SolidMechanics]
    [QuasiStatic]
      [./all]
        strain = SMALL
        incremental = true
        ⋮
      []
    []
  []
[]
```

3.2 Stress calculation

Stresses.html

The stress calculation chosen by the user must be compatible with the strain calculation (section 3.1).

ComputeFiniteStrainElasticStress for incremental and finite strain formulations.

3.3 Fixation of the model boundaries

Fix the model boundaries (usually all boundaries are only fixed in the direction of the normal and the lower boundary can also be fixed horizontally).

(to be written)

Chapter 4

Geometry

4.1 General considerations on geometry

When modelling geotechnical problems, it is often desirable to make a meaningful geometric abstraction to avoid time-consuming modelling, but also to avoid inaccuracies due to oversimplification. However, experience shows that the effort involved in creating such a model geometry should not be underestimated and that the models consist of a large number of individual parts and components - often called 'geometrical objects'. Some of these objects have trivial shapes (e.g. rectangular surfaces for a sheet pile wall segment), others can have non-trivial curvilinear boundaries (e.g. geological bodies). All of these objects must be shaped to fit each other (without gaps or overlaps) and together form the FE model. Usually, geotechnical models are managed (e.g materials, staging, etc) on the level of these objects.

4.2 'Subdomains' resp. 'Blocks'

In Moose, so-called 'subdomains' are one of the central elements for the setup and management of models. For example, materials are not assigned directly to the FE elements, but to one or several 'subdomains'. The same applies to kernels, variables and many other Moose objects. The FE elements are also assigned to subdomains, whereby each FE element can only be associated with one subdomain. The entirety of the Moose objects and settings assigned to a subdomain thus defines the element type and mechanical behaviour of all FE elements within this subdomain. Therefore, a Moose model consists of at least one 'subdomain'. Conversely, it is necessary to create at least one subdomain for each element type and physical material.

It is worth mentioning that FE elements that are assigned to a subdomain can, but do not have to, be geometrically connected. The bodies built up by the FE elements of a subdomain may therefore also have openings or internal cavities or form several individual bodies. With this concept, a subdomain must be understood as a logical, rather than a geometric, organisational entity.

Due to the use of libmesh by Moose, the term 'block' is used as a synonym for 'subdomain' in many places.

To simplify the handling of construction stages, it has also proved useful to further subdivide element sets into individual subdomains so they match the 'geometrical objects' mentioned before so they can to be used independently during the simulation (e.g. deactivated or activated at different times). This document follows this approach that Moose models are managed on the basis of subdomains/blocks.

Listing 4.1: Read mesh from a file

```
[Mesh]
  [file]
    type = FileMeshGenerator
    file = "source.msh"
  []
  second_order = true
[]
```

As the shape of these individual parts and components directly influences the shape of the FE elements they are build of, further prerequisites must be considered during geometrisation: For instance, as few 'unfavourably shaped' (e.g. too acute-angled) FE elements as possible should result in order to avoid numerical instabilities or load increments that are too small. A common reason for acute-angled elements and large numbers of elements are corner or end points that are close together and unfavourable intersections of the objects. Another requirement for geometrisation is that the number of FE elements should remain within a range that enables acceptable calculation times.

4.3 Getting the mesh into Moose

Although Moose's 'Mesh System' provides a variety of tools for generating FE meshes on the basis of input file commands, the creation of geometrically complex models with these commands is often confusing and error-prone.

Therefore, the mesh including all subdomains and FE-elements is prepared outside of Moose and imported using the `FileMeshGenerator`. The MSH file format has proven itself in this context. One possible workflow is to create the geometry of the subdomains in Blender, then generate the mesh using gmsdh and save it as an MSH file. See listing Listing 4.1 for the code required in a Moose input file for the import of an MSH file.

4.4 Lower dimensional blocks in MSH files

A notable quirk of the MSH file format when used with Moose is the definition of lower dimensional blocks. The term 'lower dimensional blocks' is used here to describe subdomains with FE elements with a lower dimensionality (e.g. shells and beams in a model with 3D solid elements). The MSH file format makes no internal distinction as to whether blocks with lower dimensional elements are to be interpreted as a side-set or as a 'real' element block. To introduce this distinction, Moose recognises the codeword `lower_dimensional_block` (although this codeword is not an official part of the MSH file format). If this codeword is found in the line of an entry in the `$PhysicalNames`, Moose assumes that this block does not represent a sideset, but contains lower dimensional elements. As shown in Listing 4.2 the codeword may be just appended at the end of the line (without beeing part of the name of the physical name) or even be part of the physical name itself.

4.5 Choice of element types

For obvious reasons, the element types of the individual subdomains must be compatible with each other (e.g. the number and distribution of nodes must match, etc). Given the often complex geometry of geotechnical models (see section 4.1), elements with triangular faces are often

Listing 4.2: Fragment of an MSH file containing ‘lower dimensional blocks’

```
$MeshFormat
4.1 0 8
$EndMeshFormat
$PhysicalNames
⋮
0 1 "PointA"                                         ← side-set (node)
1 2 "topleftedge"                                    ← side-set (line)
1 3 "lineSignPole000" lower_dimensional_block        ← block with beam-elements
1 4 "lineSignPole001_lower_dimensional_block"        ← block with beam-elements
2 5 "BoundaryZMin"                                   ← side-set (surface)
2 6 "Sign" lower_dimensional_block                   ← block with shell-elements
⋮
```

Subdomain type	Element type	Comments
volume	TET10	tetrahedral element with 10 nodes and 4 integration points
surface (e.g. shell)	TRI6	triangular element with 6 nodes
line (e.g. beam)	EDGE3	line element with 3 nodes

Table 4.1: Commonly used compatible element types for tetrahedral/triangular discretisation

preferred. In this case, volume elements are correspondingly tetrahedral (TET) and surface elements are triangular (TRI). However, since linear TET and TRI elements (TET4 and TRI3) have very poor numerical properties (e.g. unrealistically high stiffness, locking), ‘quadratic’ elements (TET10 and TRI6) must be used. This leads to the commonly used compatible element types for tetrahedral/triangular discretisation show in Table 4.1.

Currently, Moose has full support for TET10 elements, but not for EDGE3 and TET6 elements. EDGE3 and TRI6 elements can currently be used in a Moose-app by cloning the following Moose code-plugins as a ‘contrib’:

- beams: github.com/Kavan-Khaledi/moose-codeplugin-beam
- shells: github.com/Kavan-Khaledi/moose-codeplugin-shell

4.6 Groups of geometrical objects

Moose does not explicitly support groups geometrical objects but allows to create variables in the input file holding names of several objects on one hand and to use these variables directly for block-restriction (e.g. assignment of materials).

Due to the fact that Moose throws an error if unused variables are found in the input file, but the user may want to define groups even if they are not (currently) used, it has proven useful to add ’fake users’ using a ParsedFunction as shown in Listing 4.3.

Listing 4.3: Moose input file fragment: variable holding several block names and fake-user

```
[Mesh]
    # Variable holding names of all blocks with volumes
    Volumes = 'Base Column Suzanne'
[]

[Functions]
    # Fake user for the variable above
    [FakeUser_Volumes]
        type = ParsedFunction
        expression = 'a'
        symbol_names = 'a'
        symbol_values = '1'
        control_tags = ${Mesh/Volumes}
    []
[]
```

Chapter 5

Modelling of geometric entities, structures, loads, etc.

5.1 Overview

This chapter collects information on how different types of geometric entities, structures, loads etc. can be modelled. Table 5.1 contains a list of common geotechnical components and associated options that can be used for modelling them.

5.2 Entities

5.2.1 Volumes

To model a volume, such as a soil volume, concrete volume, etc., the following is required:

- subdomain: A subdomain of FE-elements of an appropriate element type (e.g. TET10).
- materials: Assignment of Moose materials defining the constitutive behaviour.

5.2.2 FixedEndSprings

Fixed end springs are currently not directly supported by Moose. As a workaround one can define a spring between nodes (subsection 5.2.3) and fix the other node.

CoupledForceNodalKernel

pamm.202200045

5.2.3 Springs between nodes (NodeToNodeAnchors)

(to be written)

mastodon: LinearSpring

5.2.4 Beams (line element)

To model beams, the following is required:

- subdomain: A subdomain of FE-elements of an appropriate element type (e.g. EDGE3).
- materials: Assignment of Moose materials defining the constitutive behaviour of the beam.

Entity	Options for Modelling	Reference
soil volume	• cluster of volume elements	sec. 5.2.1
excavation pit wall	• shell elements	sec. 5.2.5
	• cluster of volume elements in case of a 'thick' wall (e.g. slurry wall, bored pile wall)	sec. 5.2.1
concrete volume	• cluster of volume elements	sec. 5.2.1
	• shell elements in case of a 'thin' walls	sec. 5.2.5
sprayed concrete	• shell elements	sec. 5.2.5
pile, soil nail	• cluster of volume elements in case of a pile with large diameter (e.g. bored pile)	sec. 5.2.1
prop	• spring between nodes	sec. 5.2.3
	• fixed end spring	sec. 5.2.2
	• beam	sec. 5.2.4
ground anchor	• spring between nodes + (embedded) beam	sec. 5.2.3

Table 5.1: Selected geotechnical entities and their respective modelling

Currently, Moose has no support for appropriate beam elements and materials for EDGE3. In your MooseApp, the beam material for EDGE may be included as an 'contrib' by cloning into github.com/Kavan-Khaledi/moose-codeplugin-beam

5.2.5 Shell (surface element)

To model shells (or "plates"), the following is required:

- subdomain: A subdomain of FE-elements of an appropriate element type (e.g. TRI6).
- materials: Assignment of Moose materials defining the constitutive behaviour of the shell.

Currently, Moose has no support for appropriate shell elements and materials for TRI6. In your MooseApp, the shell material for TRI6 may be included as an 'contrib' by cloning into github.com/Kavan-Khaledi/moose-codeplugin-shell

5.2.6 Interfaces (surface element)

(to be written)

5.2.7 Contact

(to be written)

5.2.8 Pore Pressure Boundaries

Boundary conditions for pore pressures are to be given via the `[BCs]` block.

> When using boundary conditions in models where subdomains are to be deactivated/activated, the boundary conditions must also deactivated/activated accordingly.

(to be written)

5.3 Groundwater

Groundwater can affect subsoil and geotechnical structures in many ways. Below the water table it is generally assumed that the pore space is completely saturated with water. In this fully saturated zone, the pore water pressure caused by gravity acts on the grain structure in all directions, creating buoyancy and thereby influencing the contact stresses between the grains. If the pore water is in motion, for instance as a consequence of an external pressure gradient, flow forces will act upon the grain structure. A variety of additional effects can be triggered by temperature changes (e.g. freezing), quick changes in the external loading including cyclic or dynamic loading, and many more.

In partially saturated zones only a part of the pore space is filled with water and the rest with gas (usually air). In geotechnics, the effects of the gas is often neglected. This region has its own set of effects, including an often reduced permeability in relation to the fully saturated zone. The reduced permeability can be explained by the fact that the surface tension causes the water to retreat into the corners of the grain structure, leaving fewer direct connections between individual water droplets and thus fewer pathways for mass transport.

The complete absence of the liquid phase (e.g. water) is referred to as a dry state. Again, in geotechnics the effects of the gas is often neglected.

For partially and fully saturated conditions, a fraction of the external loads on a body of soil is transferred via the grain to grain contacts and the rest via the pore pressure. This observation leads to the concept of 'effective stresses' (e.g. [1], [3]) where the total stress σ_{tot} (often denoted also just by σ) equals the sum of the effective stress σ_{eff} (often denoted also just by σ') and the pore water pressure p_{w}:

$$\sigma_{\text{tot}} = \sigma_{\text{eff}} + p_{\text{w}} \tag{5.1}$$

or, using the shorter notation:

$$\sigma = \sigma' + p_{\text{w}} \tag{5.2}$$

(to be written)

Chapter 6

Initial Conditions

6.1 Initial conditions for Moose objects

For definition of the initial (starting) conditions for the variables a Moose simulation the ICs system is to be used.

(to be written)

6.2 Gravitation and initial stress state

The initial stress state of geotechnical simulations is often non-trivial. This is because the section of geosphere to be modelled often has a history of millions of years under different loading conditions, including gravity and tectonic processes. In addition, the Earth's surface is often non-horizontal, so that stress trajectories near the surface are oriented accordingly. Furthermore, the initial stress state may be anisotropic with a specific orientation. Gravity and the initial stress state are normally taken into account by a combination of the following elements:

- Definition of appropriate displacement boundary conditions for the model (see section 3.3).
- Activation of gravitational body force (see subsection 6.2.1).
- Definition of eigenstrains from the initial stress field (e.g. using ComputeEigenstrainFromInitialStress, see subsection 6.2.1 and subsection 6.2.3).

> ❗ In simulations including pore water, 'effectice stresses' are to be used. This includes the definition of the initial stress field, where also effective stresses are to be given.

6.2.1 Gravity

To consider gravity, the following aspects must be included in the model:

- When compiling the MooseApp: As the effect of gravity is taken into account in interaction with the material density, the Moose module `MISC` must be active in addition to the Moose 'SolidMechanics' module.
- Gravity must be activated in the input file. This is done by adding a special kernel of the type `Gravity` as shown in Listing 6.1. The direction and strength of the gravitational field is to be defined here.

Listing 6.1: Gravity kernel in a Moose inut file

```
[Kernels]
    [gravity]
        type = Gravity
        use_displaced_mesh = false
        variable = disp_z          # displacement variable the gravity is associated with
        value = -9.81              # in m/s^2
    []
[]
```

Listing 6.2: Assignment of a density to subdomain 'block1'

```
[Materials]
    [undrained_density]
      type = GenericConstantMaterial
      block = 'Block1'
      prop_names = density
      prop_values = 2500         # in kg/m^3
    []
[]
```

- The materials must be assigned a material density (Listing 6.2).

6.2.2 Initial stress state using ComputeEigenstrainFromInitialStress

In a uninfluenced homogeneous horizontal half-space (or a section thereof) under the effect of gravity, the initial stress state corresponds to a 'geostatic stress field'. This geostatic stress field can be represented analytically for a stress point as follows:

$$\sigma'_{ini,vert} = \sigma'_{ini,vert,h=0} + \gamma' h \tag{6.1}$$

$$\sigma'_{ini,horiz} = \sigma'_{ini,vert} K_0 \quad \text{with} \quad K_0 = 1 - \sin\varphi \tag{6.2}$$

In these equations, $\gamma' = \rho' g$ corresponds to the weight of the soil, h to the overburden height, $\sigma'_{ini,vert,h=0}$ to the effective vertical stress at $h = 0$ and K_0 to the earth pressure coefficient. This earth pressure coefficient is usually estimated as a function of the angle of internal friction φ (simplified equation according to Jaky).

Listing 6.3 shows the definition of a geostatic initial stress state using two functions `ini_xx_yy` and `ini_zz` and their subsequent use in a `ComputeEigenstrainFromInitialStress`.

6.2.3 Initial stress state for anisotropic stresses

Align the model to the orientation of the initial stresses

(to be written)

6.2.4 Checking the initial stress state

In geotechnics, the initial state including the initial stress state should be in equilibrium. Therefore, in a first time step without any changes to the model 'nothing' should happen (only neglectable deformations and changes in the stress field).

Listing 6.3: Definition of a geostatic initial stress state using 'ComputeEigenstrainFromInitialStress'

```
[Functions]
    [ini_xx_yy]  # function describing the initial effective horizontal stress field
      type = ParsedFunction
      vars = 'sig_top z_top  rho     g    k0 '
      vals = '-1.5    0      0.0025  10   0.3'
      value = '(sig_top - rho * g * (z_top - z)) * k0'
    []
    [ini_zz]     # function describing the initial effective vertical stress field
      type = ParsedFunction
      vars = 'sig_top z_top  rho     g'
      vals = '-1.5    0      0.0025  10'
      value = '(sig_top - rho * g * (z_top - z))'
    []
[]

[Materials]
    [strain]
      type = ComputeIncrementalSmallStrain
      volumetric_locking_correction = false
      eigenstrain_names = ini_stress     # pointing to the initial stress field
    []
    [ini_stress]  # initial stress field (effective stresses)
      type = ComputeEigenstrainFromInitialStress
      eigenstrain_name = ini_stress
      initial_stress = 'ini_xx_yy 0 0 0 ini_xx_yy 0 0 0 ini_zz'
    []
[]
```

This can be checked by introducing such a time step without any changes as a first time step and assessing the system response (e.g. in Paraview).

Chapter 7

Construction Stages

7.1 General comments on construction stages

The type of geotechnical simulation primarily discussed in this document aims to model a construction process. Accordingly, such a simulation must reproduce the various stages of construction over time. Since this document assumes that Moose models are organised on the basis of subdomains (section 4.2), the state changes also refer primarily to adjustments that affect entire subdomains.

The most common state changes in this context are:

- Activating FE-elements (e.g. installation of sheet piles is usually modelled by activating the corresponding set ot FE-elements, section 7.4)
- Deactivating FE-elements (e.g. excavation of a soil volume is usually modelled by deactivating the corresponding set of FE-elements, section 7.4)
- Replacing FE-elements (e.g. installation of a slurry wall modelled via volume elements leads to the need to replace the soil volume with the volume representing the slurry wall, section 7.5)
- Adjusting external loads (e.g. loads are used to model various influences. As these influences change over time, the corresponding loads have to be adjusted, section 7.6)
- Adjusting prestress (e.g. for props and stand anchors the prestress is to be adjusted, section 7.7)

As a consequence of the Moose input file concept, these status adjustments must be triggered at various points in an input file if Moose is used 'out of the box'. To provide a more centralised way of defining construction stages, the `[Stages]` MooseApp code plugin linked below can be used:

github.com/jmeier/moose-codeplugin-stages

This document primarily describes the procedure for the definition of construction stages, with the use of the `[Stages]` code plugin where applicable. The procedure without this plugin is only mentioned in selected places.

7.2 Time steps without changes

Avoid time steps (and therefore stages) without changes to the model. Time steps without changes may not converge, even if a convergence was found for the previous time step - compared to which no changes were made (due to the relative error being 'high').

elaborate on timesteps without changes

7.3 Abrupt versus gradual changes

For transient simulations, the speed at which changes are applied to the model is usually important. When elements are deactivated, the self-weight of these elements is removed, all loads and stresses transmitted by these elements must be redistributed, and so on. If many elements are deactivated, the resulting changes in the stress field will be correspondingly large. In a not purely linear-elastic model, comprehensive changes can lead to numerical instability, resulting in slow or no convergence.

For geotechnical problems, on the other hand, changes in reality are usually gradual.

(to be written)

7.4 Activating and deactivating FE-elements

As mentioned in section 4.2, Moose models are organized and managed by means of subdomains. By assigning FE elements and materials to a subdomain, for example, the behaviour of these FE elements is defined. However, Moose does not offer the option of activating or deactivating FE elements or subdomains during a simulation run. However, Moose allows FE elements to be reassigned from one subdomain to another during a simulation run using the MeshModifiers system.

On the other hand, Moose allows for subdomains to exist without kernels, variables, materials and other Moose objects, but for FE elements to be present in these subdomains. FE-elements in those subdomains 'without physics' will simply be ignored my Moose.

Activation and deactivation of FE elements can thus be modelled by combining the option to have subdomains 'without physics' and the option to reassign FE elements to another subdomain. For deactivation FE elements are moved from a subdomain 'with physics' to a subdomain 'without physics'. For activation vice versa.

ToDo:

- Activation / Deactivation with [Stages]
- Ramp up / ramp down material properties, section 7.3

7.5 Replacing FE-elements

(to be written)

7.6 Adjusting external loads

(to be written)

7.7 Adjusting prestress

(to be written)

Chapter 8

Material Models for Soil and Rock

8.1 Preliminary remarks on the material models

The built-in library of material models suitable for modelling soil and rock is currently very limited.

8.2 NEML2

Moose supports the use of materials defined by means of the external material modeling library NEML2 (Messner and Hu). Details on the integration of NEML2 into Moose are not scope of this document and can be found in the corresponding corresponding section of the Moose documentation.

8.3 Mohr-Coulomb (linear-elastic ideal-plastic)

(to be written)

Chapter 9

Proven Patterns

9.1 Partial input files

When a Moose model is defined as an 'input file' (usually with the file extension '*.i'), the size of such an input file can quickly grow to several hundred lines, even without the model geometry. Editing and troubleshooting becomes correspondingly confusing. To mitigate, Moose supports 'partial input files', where other input files can be called up from a 'central' input file. For example, the material parameters or variables with groups of geometrical objects could be maintained in a separate file.

The recommended option is to use the `!include` keyword within the input file. This option is shown in Listing 9.1. Please note, that neither path nor filename are allowed to contain a newline character, `#` character, or `[` character.

Listing 9.1: Include another input file

```
!include "path/to/file.i"
```

The alternative and not recommended option is to call a Moose app with several input files as arguments. These input files are interpreted by simply concatenating them. This functionality was included in Moose for test purposes, but is not suitable for productive models, as this call may not be not documented and may not be reproducible:

```
./moose-opt -i input1.i -i input2.i -i input3.i
```

9.2 Physical units

In Moose, all numerical values must be entered in a system of units which is consistent in itself - but which is selected by the user. Without additional options, the user must enter a number in the input file without a unit being explicitly labelled.

In addition, Moose - like other FE programmes - calculates internally with a finite precision. To avoid rounding errors and inaccuracies in the numerical calculation, it is therefore important that the numerical values entered are not too large. With a careful choice of the system of physical units, the user can avoid such numerical problems.

It is important to emphasise at this point that, in addition to the various material parameters, initial conditions and boundary conditions, the convergence criteria are also affected by the choice of physical units (see section 9.4).

From the user's point of view, the challenge is to enter the numerical values in the 'correct' system of physical units everywhere in the input file. This conversion and the non-trivial verifiability is a source of error that should not be underestimated, particularly in the case of physical units with a time reference. Experience shows, that this is a frequent source of problems!

To mitigate this source of error, Moose offers the 'MooseUnits' utility. In the input file this utility allows to convert the unit 'on the fly'. The input file fragment shown in Listing 9.2 converts $1000 \times 10^3\,\mathrm{kN/m^2}$ into $\mathrm{MN/m^2}$ and assigns the result to the variable `E`.

Listing 9.2: Converting $1000 \times 10^3\,\mathrm{kN/m^2}$ into $\mathrm{MN/m^2}$

```
E = ${units 1000E3 kN/m^2 -> MN/m^2}
```

In connection with variables holding the desired physical units for the model this utility can be used to enforce a consistent system of physical units in the input file and the user can simultaneously use the physical units of his source data (for better verifiability). Listing 9.3 shows the use of a variable used to hold the desired model unit for lengths.

Listing 9.3: Using a variable to hold the physical unit for lengths

```
# variable holding the pyhsical unit for lengths
modelunit_length = 'm'

# conversion of 10 mm into the desired model unit (m)
a = ${units 10 mm -> ${modelunit_length}}
```

A full set of pysical units can be setup as shown in Listing 9.4. If not all variables are used in the input file, Moose will issue an error. This can be avoided by adding fake-users to the variables.

Listing 9.4: Set of consistent physical units in a Moose inut file

```
# model units
modelunit_length = 'm'
modelunit_time = 's'  #s = seconds, h = hours, day = days
modelunit_mass = 'kg' # Mg = tons; Gg = kilotons

# derived units (this may be moved into a !include)
modelunit_force = ${raw ${modelunit_mass} * ${modelunit_length} / ${modelunit_time} ^ 2}
modelunit_pressure = ${raw ${modelunit_force} / ${modelunit_length} ^ 2}
modelunit_acceleration = ${raw ${modelunit_length} / ${modelunit_time} ^ 2}
modelunit_density = ${raw ${modelunit_mass} / ${modelunit_length} ^ 3}

# Fake users for the variables above
[Functions]
    [FakeUser_modelunit_force]
        type = ParsedFunction
        expression = 'a'
        symbol_names = 'a'
        symbol_values = '1'
        control_tags = ${modelunit_force}
    []
    ⋮
[]
```

A selection of model unit combinations can be found in Table 9.1.

Table 9.1: selection of model unit combinations and derived units

Length	Time	Mass	Force	Pressure	Acceleration	Density
m	s	kg	$kg \cdot m/s^2 = N$	$N/m^2 = Pa$	m/s^2	kg/m^3
m	s	t	$t \cdot m/s^2 = kN$	$kN/m^2 = kPa$	m/s^2	t/m^3

9.3 Moose code plugins

If MooseApps are extended with new MooseObjects that should be shared with other users, e.g. in workgroups, the need arises to be able to transfer this functionality to other MooseApps as a plug-in. Preferably without having to manually copy and adapt many code files. Furthermore, it is often good if there is independent source code management or versioning. The idea is to realise such code plugins for MooseApps as git submodules.

To address this need and provide versioning via a git submodule, Moose code plugins to be found in the following link can be used: github.com/jmeier/mooseapp-code-plugin

9.4 Convergence criteria and variable scaling

Due to the fact that the finite element method will always provide approximate solutions, residuals will remain after each simulation step. To allow Moose to recognize converge, suitable convergence criteria (e.g. tolerated residuals) must be set. If these convergence criteria are set too small, moose will appear not to converge and will need very small time steps to do anything at all. If set too large, Moose may converge to the wrong result.

Closely related to the selection of tolerated residuals is the scaling of variables. If only one physical aspect is to be modelled (e.g. mechanics), an increased tolerated residual is equal to an downscaling of the corresponding model variables. If several pysical aspects are to be modelled (e.g. mechanics and flow) variable scaling and adjusting tolerated residuals must be applied in combination.

It should also be noted that the convergence criteria (e.g. tolerated residuals and variable scaling) must be set in the context of the chosen system of physical units (section 9.2).

Large parts of this section is taken from mooseframework.inl.gov.

9.4.1 Nonlinear and (nested) linear solves

9.4.2 Residuals

In this subsection data is collected on how to estimate the absolute residuals to be tolerated (e.g. `nl_abs_tol` of an executioner) for selected physical aspects.

ToDo: Using `Debug/show_var_residual_norms = true` the residuals of the individual variables are printed.

ToDo: Moose will print "Outlier Variable Residual Norms", explained here:

> The outliers are those variables whose square of their individual L2-norm is "too far" from the square of the average norm. Right now "too far" is some fixed value (currently 2.0) determined empirically by a computer scientist ;) If you want to see the actually calculation of that value, it's in NonlinearSystem.C. We print them because they might be too big to reliably pass a tolerance check. It's sort of like a yellow flag

if you see a bunch of those coming out. You might try tighter tolerances or different solver options, etc .

Residuals of a variable are affected by the scaling of the variable. (subsection 9.4.3).

Fluid

The residual for the fluid phase should be a measure of what is still considered a *steady state* in the pore pressure field. This residual for the fluid phase can be estimated using Equation 9.1, where beside some material constants for the fluid (density ρ_f and dynamic viscosity μ_f) and the porosity κ, the accepted pressure gradient ϵ_f (e.g. $\epsilon_\mathrm{f} = 1\,\mathrm{N}/(\mathrm{m}^2\,\mathrm{m})$) must be definded for a volume of interest V. In geotechnics, this volume of interest is often only a part of the full mesh.

$$R_f \approx V \cdot \det(\kappa) \cdot \epsilon_\mathrm{f} \cdot \rho_\mathrm{f}/\mu_\mathrm{f} \tag{9.1}$$

with:

R_f estimate of residual for the fluid

V Volume of interest (physical unit: volume, e.g m^3)

κ tensor of permeabilities (physical unit of the tensor items: area, e.g. m^2)

ϵ_f desired precision (physical unit: pressure per length, e.g. $\mathrm{Pa/m} = \mathrm{N}/(\mathrm{m}^2{\cdot}\mathrm{m})$)

ρ_f fluid density (physical unit: mass per volume, e.g. kg/m^3)

μ_f fluid dynamic viscosity (physical unit: mass per length and time, e.g. $\mathrm{kg}/(\mathrm{m}\cdot\mathrm{s}) = \mathrm{Pa}\cdot\mathrm{s}$)

For water ($\rho_\mathrm{f} \approx 1000\,\mathrm{kg/m}^3$, $\mu_\mathrm{f} \approx 1\,\mathrm{mPa/s}$) this equation simplifies to:

$$R_{water} \approx V \cdot \det(\kappa) \cdot \epsilon_\mathrm{water} \cdot 10^{-6}\,(\mathrm{kg}\cdot\mathrm{s})/(\mathrm{m}^3{\cdot}\mathrm{Pa}) \tag{9.2}$$

Mechanics

Very similar to the residuals of the other phases, the residual for the solid phase should be a measure of what is still considered a *steady state* in the stress field. This residual for the solid phase can be estimated using Equation 9.3, again using a volume of interest V and an acceptable residual stress gradient ϵ_s (e.g. the error accepted in $\nabla\sigma$).

$$R_s \approx V \cdot |\epsilon_\mathrm{s}| \tag{9.3}$$

with:

R_s estimate of residual for the solid (physical unit: force, e.g. N)

V Volume of interest (physical unit: volume, e.g m^3)

ϵ_s desired precision (physical unit: pressure per length, e.g. $\mathrm{Pa/m} = \mathrm{N}/(\mathrm{m}^2{\cdot}\mathrm{m})$)

9.4.3 Variable scaling

To allow efficient nonlinear and linear solves, the Jacobians of the different physics of a Moose model should have comparable magnitudes. Ideally, these should all be as close to unity as possible. With the possibility of variable scaling, Moose offers a tool that allows this to be achieved.

Variable scaling primarily affects the internal calculation processes (e.g. Jacobian and residuals). The simulation results, however, are generally not scaled, which means that variable scaling does not need to be considered in post-processing.

Manual variable scaling

When defining the variables, the 'scaling' property can be used to specify a manual scaling factor for each variable individually. Listing 9.5 shows a snipped of an input file with a scaling factor of 1×10^5 applied to the variable 'porepressure'.

Listing 9.5: Variable definition with scaling on porepressure in a Moose inut file

```
[Variables]
    [disp_x]
        family = LAGRANGE
        order = SECOND
    []
    [disp_y]
        family = LAGRANGE
        order = SECOND
    []
    [disp_z]
        family = LAGRANGE
        order = SECOND
    []
    [porepressure]
        family = LAGRANGE
        order = SECOND
        scaling = 1E+5
    []
[]
```

Manual variable scaling can be determined as follows:

1. Variable scaling of all variables is set to unity.
2. Using `Debug/show_var_residual_norms = true` the residuals of the individual variables are printed for a test run of the model. The residuals for the first time step ("Time Step 1") may look like the following example:

```
|residual|_2 of individual variables:
          disp_x:       19.4404
          disp_y:       19.4404
          disp_z:       110.432
          porepressure: 7.33977e-05
0 Nonlinear |R| = 1.138024e+02
```

3. In the example in item 2, the magnitude of the residuals for the `disp`-variables are roughly the same, while the magnitude of the `porepressure` is much smaller. Therefore, without scaling, the contribution to the residual of the `porepressure` would be strongly under-represented and one would need very small `abs_norm_tol` to make sure that the hydraulic

equation converges. On the other hand, this may cause convergence problems for the displacement equations as they have larger residuals.
It is therefore advisable for this model to scale the `porepressure` in such a way that the associated residuals are comparable with those of the `disp` variables using (while leaving scaling of all `disp` variables at unity): `Variables/porepressure/scaling = 1E+5`

Automatic variable scaling

As alternative to manual variable scaling, Moose offers the functionality of automatic variable scaling, calculating the scaling factors automatically and overriding any - if present - manually given scaling factors. Automatic scaling can be activated by setting `automatic_scaling = true` in the `[Executioner]` block. To output the calculated scaling factors, set `verbose = true` in the `[Executioner]` block.

In the author's experience, at least for hydro-mechanically coupled models, automatic scaling often leads to suboptimal convergence compared to manual scaling. Therefore, it is recommended to use manual scaling instead.

9.4.4 Tolerances and their default values

Table 9.2: Default values for tolerances and maximum number of iterations

solve	absolute tolerance	relative tolerance	maximum iterations
linear	`l_abs_tol = 1e-50`	`l_tol = 1e-05`	`l_max_its = 10000`
nonlinear	`nl_abs_tol = 1e-50`	`nl_rel_tol = 1e-08`	`nl_max_its = 50`

Physical meaning of a (absolute) tolerance depends on the variable the tolerance is definded for. Solid mechanics variables (often called `disp_x`, `disp_y`, and `disp_z`) is:

$stress \cdot strain = \mathrm{kN/m^2 \cdot m\,m^{-1}} = \frac{Energy}{Volume} = kN * m/m$

Some collected notes on the tolerances and how to choose:

- Assuming the residual close to unity, `nl_abs_tol` and `l_abs_tol` should be larger than 1e-15 to be not too close to numerical precision.
- The linear tolerance should be looser than the nonlinear tolerance (the linear tolerance should be a larger number than for the nonlinear). In fact since the residual is checked by the nonlinear solver, one don't need to converge the linear solve anywhere near as tight.
- From experience, `l_tol = 1e-3` is often enough.
- According to petsc.org, the default convergence algorithm reaches convergence when both (absolute and relative) tolerances are satisfied and the following equation holds:
  $rnorm < \max(\texttt{rel\_tol} \cdot rnorm_0, \texttt{abs\_tol})$.
- Relative tolerances are often chosen to be 1 % in other geotechnical finite element codes.

Chapter 10

Postprocessing

10.1 Paraview

Directly use the ExodusII-files produced by Moose.

(to be written)

10.2 Blender using the BVtkNodes addon

The BVtkNodes addon for Blender that wraps the Visualization Toolkit (VTK) library for scientific visualization in Blender.

(to be written)

10.3 Directly accessing ExodusII files

The primary output format of Moose is ExodusII. Accessing the data directly in this format is therefore a logical step. To work with ExodusII-files under Windows, one might want to compile the corresponding code in the seacas-repo under Windows. The step by step instructions on how to compile ExodusII for windows can be found here: seacas#375.

(to be written)

Chapter 11

Outlook

The following topics are currently not covered, but may be of interest:

- Restart a simulation run from a given time step.
- Branching simulations runs (closely related to restarting simulation runs)
- Changing the solver and convergence criteria depending on (simulation) time

Chapter 12

References

Publications

[1] J. E. B. Jennings and J. B. Burland. Limitations to the use of effective stresses in partly saturated soils. *Géotechnique*, 12(2):125–144, 1962. doi: 10.1680/geot.1962.12.2.125. URL https://doi.org/10.1680/geot.1962.12.2.125.

[2] Cody J. Permann, Derek R. Gaston, David Andrš, Robert W. Carlsen, Fande Kong, Alexander D. Lindsay, Jason M. Miller, John W. Peterson, Andrew E. Slaughter, Roy H. Stogner, and Richard C. Martineau. Moose: Enabling massively parallel multiphysics simulation. *SoftwareX*, 11:100430, 2020. ISSN 2352-7110. doi: https://doi.org/10.1016/j.softx.2020.100430. URL https://www.sciencedirect.com/science/article/pii/S2352711019302973.

[3] A. W. Skempton. *Effective Stress in Soils, Concrete and Rocks*, pages 106–118. Thomas Telford Publishing, 1984. ISBN 978-0-7277-3982-7. doi: 10.1680/sposm.02050.0014. URL https://www.icevirtuallibrary.com/doi/abs/10.1680/sposm.02050.0014.

Appendices

Appendix A

Stiffness Reduction

A.1 Problem statement

The stiffness of materials can change over time as a result of certain processes. One example is the temperature rise of a rubber block, which becomes ‘softer’ and ‘slumps down’ as a result of this temperature load. Similar effects may be observed as a result of weathering of geomaterials.

To simulate this effect in a simplified way, the reduction of stiffness of a block of $1\,\mathrm{m} \times 1\,\mathrm{m} \times 1\,\mathrm{m}$ is to be simulated. While the (vertical) sides are fixed horizontally and the bottom is fixed vertically, the top surface loaded with a pressure of $\sigma_{top} = 1.0\,\mathrm{kN/m^2}$. Two variants are to be simulated, the first without dead load and the second with the block also loaded by gravity. The soil material is assumed to be purely linear-elastic and no water present. Material parameters are given in Table A.1. The stiffness of the material was chosen with an arbitrary initial value of $E_0 = 1000\,\mathrm{kN/m^2}$ to be reduced to $E_1 = 500\,\mathrm{kN/m^2}$.

Vertical displacement u_z at the top of the block is to be determined as simulation result.

Table A.1: Material parameters

Property	Physical unit	Value
initial Youngs modulus E_0	$\mathrm{kN/m^2}$	1000
final Youngs modulus E_1	$\mathrm{kN/m^2}$	500
Poisson’s ratio ν	-	0.0
density (weightless | finite) ρ_s	$\mathrm{kg/m^3}$	0 | 2500

Note that due to the way this example is modelled, changes in Young’s modulus will affect the displacements even if the loading remains constant.
A decrease in Young’s modulus will make the material ’softer’ and it will ’sink’. An increase in Young’s modulus will make the material ’stiffer’ and it will ’expand’.

A.2 Analytical solution

In the case of $\nu = 0.0$ given here, the vertical deformation Δh can be analytical determined from Hooke’s law (Equation A.1):

$$E = \frac{\sigma_{zz}}{\varepsilon_{zz}} \rightarrow \varepsilon_{zz} = \frac{\sigma_{zz}}{E} = \frac{\Delta h}{h} \tag{A.1}$$

For the weightless material this leads to:

$$u_z = \frac{\sigma_{top}}{E_1} - \frac{\sigma_{top}}{E_0} = 0.001\,\mathrm{m} = 1\,\mathrm{mm} \tag{A.2}$$

For the material with $\rho_s = 2500\,\mathrm{kg/m^3}$ also subjected to a gravitational acceleration of $g = 9.81\,\mathrm{m/s^2}$, the settlement of the upper boundary can be calculated as for a spring under its own weight:

$$u_z = \left[\frac{\sigma_{top}}{E_1} - \frac{\sigma_{top}}{E_0}\right] + \left[\frac{\rho_s \cdot g \cdot h}{2 \cdot E_1} - \frac{\rho_s \cdot g \cdot h}{2 \cdot E_0}\right] = 1\,\mathrm{mm} + 12.26\,\mathrm{mm} = 13.26\,\mathrm{mm} \tag{A.3}$$

A.3 Numerical solution with Moose

The system of physical units for the Moose model was set to metre (m), seconds (s), and tons (t). Due to this, the physical unit for stress and pore-pressures is kN/m^2.

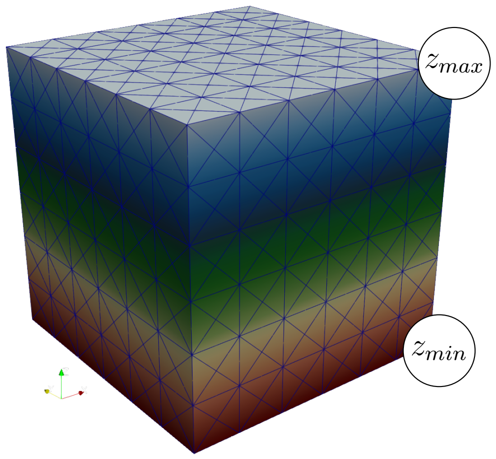
The mesh used for Moose consists of 8245 nodes forming 5184 TET10 elements leading to 24 735 DOF. Due to its pure linear-elastic setup, this model was calculated using the 'NEWTON' solver.

The Moose input file for this model is attached to this document by the name of 'stiffness-reduction.i'.

stiffness-reduction.i

Table A.2: Selected results of the Moose models

Model response	Material density	
	0 kN/m^3	2500 kN/m^3
Initial Conditions:		
$\sigma'_{zz,z_{max}}$ (kN/m^2)	±0.0	−1.0
$\sigma'_{zz,z_{min}}$ (kN/m^2)	±0.0	−25.53
Boundary at z_{max} after stiffness reduction:		
u_z (m)	-1.034×10^{-3}	-13.41×10^{-3}



Appendix B

Bended Plate

B.1 Problem statement

An initially horizontal and unloaded quadratic plate of $a = 2\,\mathrm{m} \times b = 2\,\mathrm{m}$ with a thickness of $t = 0.1\,\mathrm{m} = 10\,\mathrm{cm}$ is only held at one edge and is subjected at the opposite edge with a static point load of $F = 100\,\mathrm{kN}$ directed upwards. The material behaviour is purely linear-elastic.

The deflection of the plate at the loaded edge is to be determined as a simulation result.

This problem is closely related to classical beam bending theory for a cantilever (fully fixed at one end) with a point load at the (opposite) free end. This conveniently provides an analytical solution (section B.2) against which the numerical results can be checked (section B.3 and section B.4). With this problem setup, the plate may be modelled using volume elements or shell elements.

Table B.1: Material parameters

Property	Physical unit	Value
Youngs modulus E	$\mathrm{kN/m^2}$	1×10^6
Poisson's ratio ν	-	0.3
density ρ	$\mathrm{kg/m^3}$	0

B.2 Analytical solution

Neglecting geometrical non-linearity, the well-known analytical solution for the deflection is given in Equation B.1.

$$u_z = F \cdot \frac{a^3}{3EI} = 1.6\,\mathrm{m} \tag{B.1}$$

with:

u_z vertical deflection at the point load to be estimated.

F point load ($100\,\mathrm{kN}$)

a size of the plate ($2\,\mathrm{m}$)

E Young's modulus ($1 \times 10^6\,\mathrm{kN/m^2}$; see Table B.1)

I Second moment of area, for this geometry $I = \frac{b \cdot t^3}{12}$

B.3 Moose

In Table B.2 the deflection and some additional metrics of selected Moose models for this problem is shown.

For this purely linear-elastic problem, the 'NEWTON' solver is used. The Moose input files for this model are attached to this document by the name of 'bended-plate.i'.

Table B.2: Resulting deflection for selected Moose models (geometrical non-linearity neglected)

Discretisation	deflection (m)	CPU-count / wall time (s)	RAM (MB)
xxxxxxx TRI6		0 /	
xxxxxxx TET10		0 /	
xxxxxxx TET10		0 /	

B.4 Plaxis 3D

Table B.3

Table B.3: Resulting deflection for selected Plaxis models (geometrical non-linearity neglected)

Discretisation	deflection (m)	CPU-count / wall time (s)	RAM (MB)
xxxxxxx TRI6		0 /	
xxxxxxx TET10		0 /	
xxxxxxx TET10		0 /	

Appendix C

Fully Saturated Undrained Compression

C.1 Problem statement

A one-dimensional compression test of a $6\,\mathrm{m} \times 6\,\mathrm{m} \times 6\,\mathrm{m}$ fully saturated block is to be simulated as a variant of an Oedometer test. While the (vertical) sides are fixed horizontally and the bottom is fixed vertically, the top surface is lowered by $0.01\,\mathrm{m} = 1\,\mathrm{cm}$. Initially, the block is loaded only by gravity. The soil material is assumed to be purely linear elastic and fully saturated with water. Material parameters are given in Table C.1. Stiffness of the material had been chosen in accordance with Opalinus clay (a sedimentary rock), while permeability should been chosen high enough to avoid significant time-dependent effects.

The effective vertical stress σ'_{zz} and pore pressure u are to be determined as simulation result at the top of the block.

Table C.1: Material parameters

Property	Physical unit	Value
soil: Youngs modulus E	$\mathrm{GN/m^2}$	18
soil: Poisson's ratio ν	-	0.25
soil: density ρ_s	$\mathrm{kg/m^3}$	2500
soil: permeability k	$\mathrm{m^2}$	(realistic high)
water: bulk modulus K_w	$\mathrm{GN/m^2}$	2.2
water: dynamic viscosity μ_w	$\mathrm{mPa \cdot s}$	0.9
water: density ρ_w	$\mathrm{kg/m^3}$	998.21
porosity n	-	0.11

C.2 Analytical solution

For the one-dimensional deformation conditions and the linear-elastic material behaviour defined for this problem the definition of the stiffness modulus in Equation C.1 holds:

$$E_s = \frac{\sigma'_{zz}}{\varepsilon_{zz}} = E \cdot \frac{1-\nu}{(1+\nu)(1-2\nu)} \tag{C.1}$$

with:

E_s Stiffness modulus

σ'_{zz} effective vertical stress

ε_{zz} vertical strain

E Young's modulus

ν Poisson's ratio

Using the values of Table C.1 for the second part of Equation C.1 leads to:

$$\begin{aligned} E_s &= E \cdot \frac{1-\nu}{(1+\nu)(1-2\nu)} \\ &= 18\,\mathrm{GN/m^2} \cdot \frac{1-0.25}{(1+0.25)(1-2\cdot 0.25)} \\ &= 21.6\,\mathrm{GN/m^2} \end{aligned} \tag{C.2}$$

Rearranging the first part of Equation C.1 for σ'_{zz} allows to estimate the effective stresses at the top of the block:

$$\begin{aligned} \sigma'_{zz} &= E_s \cdot \varepsilon_{zz} \\ &= 21.6\,\mathrm{GN/m^2} \cdot \frac{0.01\,\mathrm{m}\cdot 6\,\mathrm{m}\cdot 6\,\mathrm{m}}{6\,\mathrm{m}\cdot 6\,\mathrm{m}\cdot 6\,\mathrm{m}} \\ &= 0.036\,\mathrm{GN/m^2} = 3.6\times 10^4\,\mathrm{kN/m^2} \end{aligned} \tag{C.3}$$

To estimate the pore-pressure, Equation C.4 can be used:

$$C = \frac{\Delta u}{\Delta \sigma_{zz}} = \frac{1}{1+n\cdot\frac{m_w}{m_v}} \tag{C.4}$$

with:

C Pore pressure parameter

Δu pore pressure

$\Delta\sigma_{zz}$ difference of total vertical stress, $\Delta\sigma_{zz} = \Delta(\sigma'_{zz} + u)$

n Porosity

m_w Coefficient of volume compressibility for water, $m_w = 1/K_w \equiv 1/2.2\,\mathrm{GN/m^2}$

m_v Coefficient of volume compressibility for the solid; here assumed to be linear-elastic and therefore not dependent on the stress and/or initial porosity: $m_v = 1/E_s$

leading to:

$$C = \frac{1}{1+n\cdot\frac{E_s}{K_w}} \approx 0.48\overline{076923} \tag{C.5}$$

and due to the fact the initial stress state is Zero, Equation C.4 can be rearranged to calculate the pore pressure u:

$$u = \frac{C\cdot\Delta\sigma_{zz}}{1-C} = 3.\overline{3}\times 10^4\,\mathrm{kN/m^2} \tag{C.6}$$

C.3 Numerical solution with Moose

The system of physical units for the Moose model was set to metre (m), seconds (s), and tons (t). Due to this, the physical unit for stress and pore-pressures is $\mathrm{kN/m^2}$. Because of the physical unit of seconds (s) and to keep model time to numbers with small magnitude, a high permeability of $k = 1 \times 10^{-7}\,\mathrm{m^2}$ was chosen (comparable with a gravel) and model time was set to $10\,\mathrm{s}$.

The mesh used for Moose consists of 5184 TET10 elements leading to 32 980 DOF.

Residuals for the fluid (water with a dynamic viscosity of $\mu_\mathrm{f} \approx 1\,\mathrm{mPa \cdot s}$) with an accepted pressure gradient of $\epsilon_\mathrm{f} \equiv 10\,\mathrm{Pa/m}$ may be estimated as:

$$\begin{aligned} R_\mathrm{f} &\approx V \cdot \kappa \cdot \epsilon_\mathrm{f} \cdot \rho_\mathrm{f} / \mu_\mathrm{f} \\ &\approx (216\,\mathrm{m^3}) \cdot (1 \times 10^{-7}\,\mathrm{m^2}) \cdot (10\,\mathrm{Pa/m}) \cdot (1000\,\mathrm{kg/m^3}) / (1\,\mathrm{mPa \cdot s}) \\ &\approx 2.16 \times 10^{2}\,\mathrm{kg \cdot m/s} \qquad \text{(in SI units)} \\ &\approx 2.16 \times 10^{-1}\,\mathrm{t \cdot m/s} \qquad \text{(in accordance with the selected model units)} \end{aligned} \tag{C.7}$$

For this model, variable scaling was determined as follows:

1. Variable scaling of all variables (`disp_x`, `disp_y`, `disp_z`, and `porepressure`) is set to unity.
2. Using `Debug/show_var_residual_norms = true` the residuals of the individual variables are printed for a test run of the model. The residuals for the first time step (Time Step 1) are:

```
|residual|_2 of individual variables:
          disp_x:       19.4404
          disp_y:       19.4404
          disp_z:       110.432
          porepressure: 7.33977e-05
0 Nonlinear |R| = 1.138024e+02
```

3. The magnitude of the residuals in item 2 for the `disp`-variables are roughly the same, while the magnitude of the `porepressure` is much smaller. Therefore, without scaling, the contribution to the residual of the `porepressure` would be strongly underrepresented and one would need very small `abs_norm_tol` to make sure that the hydraulic equation converges. On the other hand, this may cause convergence problems for the displacement equations as they have larger residuals.
   It is therefore advisable for this model to scale the `porepressure` in such a way that the associated residuals are comparable with those of the `disp` variables using (while leaving scaling of all `disp` variables at unity): `Variables/porepressure/scaling = 1E+5`

In Moose, this model was calculated and analysed separately using both the 'NEWTON' solver and the 'PJFNK' solver. The Moose input files for this model are attached to this document by the name of 'fully-saturated-undrained-compression-NEWTON.i' and 'fully-saturated-undrained-compression-PJFNK.i'.

fully-saturated-undrained-compression-NEWTON.i

fully-saturated-undrained-compression-PJFNK.i

Selected results of the Moose models are summarized in Table C.2 (both solvers used lead to the same results within the specified precision). Performance data can be found in Table C.3.

Table C.2: Selected results of the Moose models
(identical for solvers ‘NEWTON’ and ‘PJFNK’ for the shown precision)

Model response	Location	
	z_{max}	z_{min}
Initial Conditions:		
σ'_{zz} (kN/m^2)	± 0.0	-90.0
u (kN/m^2)	± 0.0	-60.0
Boundary at z_{max} lowered by 1 cm ($t = 11$ s):		
σ'_{zz} (kN/m^2)	-3.600×10^4	-3.608×10^4
u (kN/m^2)	-3.335×10^4	-3.341×10^4

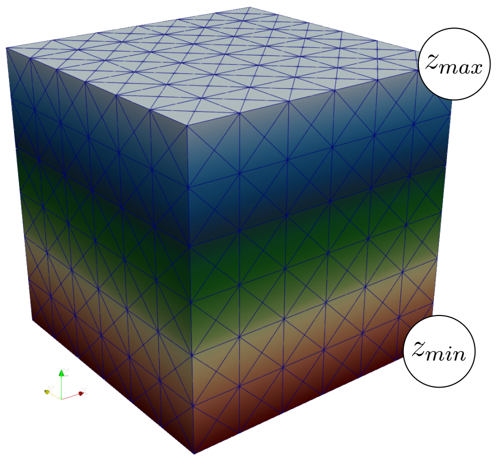


Table C.3: Performance of Moose

Property	Solver "NEWTON"			Solver "PJFNK"		
	1 CPU	3 CPU	5 CPU	1 CPU	3 CPU	5 CPU
wall time	105 s	45 s	32 s	169 s	69 s	46 s

C.4 Numerical solution with Plaxis 3D

For Plaxis, the software-default system of physical units was used (metre – m, days – d, and kilogram – kg). From this system of units stresses and pore-pressures are automaticly converted by Plaxis to kN/m^2. Because of the physical unit of days (d) and to keep model time to numbers with small magnitude, a hydraulic conductivity of $k_f = 1\,\mathrm{m/d}$ was chosen (comparable with a gravel) and model time was set to 1 d.

The mesh used for Plaxis3D consists of 5130 TET10 elements.

Hydraulic conductivity, often used in geotechnical practice, can be calculated from permeability as follows:

$$k_f = k \frac{\rho_w g}{\mu_w} \tag{C.8}$$

with

k_f hydraulic conductivity (usually in m/s or m/d)

k permeability (usually in m^2)

ρ_w fluid density, for water $\rho_{water} \approx 998.21\,\mathrm{kg/m^3}$

μ_w fluid dynamic viscosity, for water $\mu_{water} \approx 0.9\,\mathrm{mPa \cdot s}$

therefore, for water:

$$\begin{aligned} k_f &\approx k \cdot 1.09 \times 10^{-7}\,\mathrm{m^{-1}\,s^{-1}} \\ &\approx k \cdot 9.4176 \times 10^{-3}\,\mathrm{m^{-1}\,d^{-1}} \end{aligned} \tag{C.9}$$

The Plaxis3D commands file for this example is attached to this document by the name of ‘fully-saturated-undrained-compression.p3dlog’. fully-saturated-undrained-compression.p3dlog

Selected results of this Plaxis model are summarized in Table C.4. Performance data can be found in Table C.5.

Table C.4: Selected results of Plaxis3D v2023.2.1.1079

Model response	Location z_{max}	Location z_{min}
Initial Conditions:		
σ'_{zz} (kN/m^2)	± 0.0	-90.0
u (kN/m^2)	± 0.0	-60.0
Boundary at z_{max} lowered by 1 cm:		
σ'_{zz} (kN/m^2)	-3.600×10^4	-3.609×10^4
u (kN/m^2)	-3.333×10^4	-3.339×10^4

Note: In Plaxis, compressive pore pressures have a negative sign.

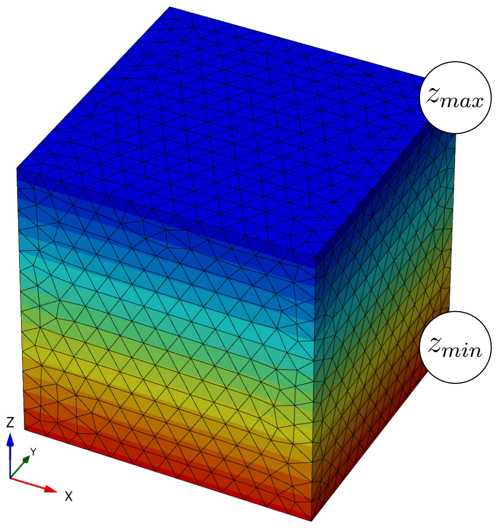


Table C.5: Performance of Plaxis3D v2023.2.1.1079

Property	Solver "Picos" (multicore iterative) 1 CPU	Solver "Picos" (multicore iterative) 3 CPU	Solver "Pardiso" (multicore direct) 1 CPU	Solver "Pardiso" (multicore direct) 24 CPU
steps	123	123	123	123
wall time	518 s	229 s	282 s	80 s
RAM	315 MB	355 MB	469 MB	519 MB

Appendix D

Bi-axial Shear Test

D.1 Problem statement

D.2 Analytical solution

D.3 Moose

D.4 Plaxis 3D