gruner

Geotechnical Simulations with Moose

– DRAFT –

This whitepaper collects information and code
for geotechnical simulations with the software Moose.

Jörg Meier

Gruner AG, Geotechnical Engineering
Basel, Switzerland

10.10.2024

Contents

Chapter 1

Introduction

The open-source, parallel finite element framework Moose (mooseframework.inl.gov) can also be used to perform geotechnical simulations. This document aims to collect options and techniques that are suitable for creating such geotechnical simulations with mosses. Main focus are quasi-static simulations.

This document is primarily intended as a collection of information rather than a textbook or step-by-step guide.

Chapter 2

Geometry

2.1 General Considerations

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.

2.2 'Subdomains' resp. 'Blocks'

Geometrically, a Moose model consists of one or more 'subdomains'. Due to the use of libmesh by Moose, the term 'block' is also used as a synonym for 'subdomain' in many places. Element types and Moose objects such as materials, kernels, etc. are assigned to these subdomains in turn. Furthermore, each FE element belongs to exactly one subdomain.

It is therefore necessary to create at least one subdomain for each element type and each material. 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.

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.

2.3 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 2.1 for the code required in a Moose input file for the import of an MSH file.

Listing 2.1: Read mesh from a file

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

2.4 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 sideset or as a 'real' element block. To introduce this distiction Moose recognizes the `lower_dimensional_block` codeword (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 2.2 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 pat of the physical name itself.

Listing 2.2: Fragment of an MSH file containing 'lower dimensional blocks'

```
$MeshFormat
4.1 0 8
$EndMeshFormat
$PhysicalNames
⋮
0 1 "PointA"                                    ← sideset (node)
1 2 "topleftedge"                               ← sideset (line)
1 3 "lineSignPole000" lower_dimensional_block   ← block with beam-elements
1 4 "lineSignPole001_lower_dimensional_block"   ← block with beam-elements
2 5 "BoundaryZMin"                              ← sideset (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 2.1: Commonly used compatible element types for tetrahedral/triangular discretisation

2.5 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 2.1), elements with triangular faces are often 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 2.1.

Currently, Moose has full support for TET10 elements, but not for EDGE3 and TET6 elements. EDGE3 and TRI6 elements can currenty 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

2.6 Groups of geometrical objects

Moose does not explicity 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 2.3.

Listing 2.3: 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 3

Modelling of geometric entities, structures, loads, etc.

This chapter collects information on how different types of geometric entities, structures, loads etc. can be modelled.

(to be written)

3.1 Overview

3.2 Entities

3.2.1 Volumes

3.2.2 FixedEndSprings

https://mooseframework.inl.gov/source/nodalkernels/CoupledForceNodalKernel.html

https://onlinelibrary.wiley.com/doi/pdf/10.1002/pamm.202200045

3.2.3 Springs between nodes (NodeToNodeAnchors)

https://mooseframework.inl.gov/mastodon/source/materials/LinearSpring.html

Entity	Options for Modelling	Reference
soil volume	• cluster of volume elements	subsection 3.2.1
excavation pit wall	• shell elements	subsection 3.2.5
	• cluster of volume elements in case of a 'thick' wall (e.g. slurry wall, bored pile wall)	subsection 3.2.1
pile	• cluster of volume elements in case of a pile with large diameter (e.g. bored pile)	subsection 3.2.1
prop	• spring between nodes	subsection 3.2.3
	• fixed end spring	subsection 3.2.2
	• beam	subsection 3.2.4

Table 3.1: Selected geotechnical entities and their respective modelling

3.2.4 Beams (line element)

3.2.5 Shell (surface element)

3.2.6 Interfaces (surface element)

3.2.7 Contact

3.2.8 Pore Pressure Boundaries

Chapter 4

Material Models

(to be written)

Chapter 5

General Model Setup

(to be written)

Calculation type:

- Incremental Small Strains Strains.html
- Stress Stresses.html

Chapter 6

Initial Conditions

(to be written)

- Initial Stress State

Chapter 7

Construction Stages

(to be written)

- Construction Stages
a. Activating and Deactivating Elements
b. Replacing Elements
c. Adjusting Prestress

Chapter 8

Postprocessing

(to be written)