
OpenFDEM Solver

Release 3.5.3

Grasselli's Geomechanics Group

Jul 27, 2023

CONTENTS:

CHAPTER ONE

WHAT IS OPENFDEM

OpenFDEM aims to be a free finite and discrete element solver with object-oriented architecture for solving **multiscale**, **multiphase**, and **multiphysics** (3M) problems. The applications are, but not limited to, mechanical, thermal and fluid mechanics problems. The main features are:

Modular & Extensible FEM Kernel and DEM Kernel (OPENFDEMlib)

- **Fully extendable and portable** - The kernel can be extended in any “direction”. Adding new element types, new material models with any number of element types and internal history parameters, new boundary conditions (time-dependent, position-dependent, state-dependent, periodic and flow-in/out) or numerical algorithms (explicit and implicit) is possible, as well as the ability to add and manage arbitrary degrees of freedom is a matter of course. (It is not a Y-like code limited to constant-strain triangular elements, Q4 cohesive elements and triangle-triangle contacts; OpenFDEM is intended to be a more general FEM/DEM solver compatible with arbitrary scenarios.) Like other common open-source FEM solvers, the most important feature of OpenFDEM is its standardization and generality, which allows the continuum-discontinuum method to be used with more general scenarios.

Fig. 1: OpenFDEM supports 26 element types and 24 materials.

- **Flexible element definition** - OpenFDEM is not limited only to triangular elements, but allows Q4 cohesive element and triangle-triangle contact. OpenFDEM is a more general FEM/DEM solver can be compatible to arbitrary scenarios.

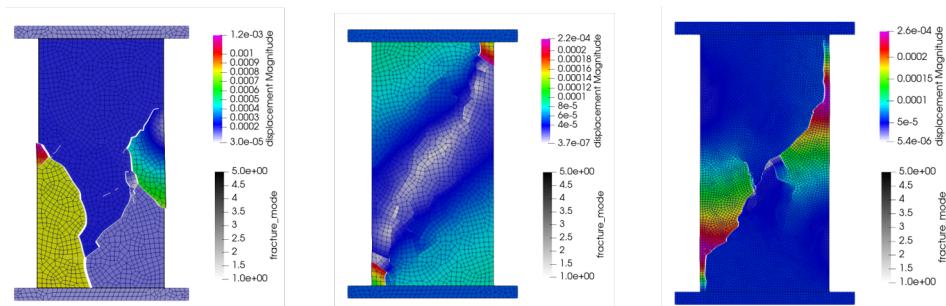


Fig. 2: Quadratic elements are supported in OpenFDEM. Combined linear element (quadrilateral and triangle elements, left), quadratic quadrilateral element (middle) and refined linear quadrilateral element (right).

- **Highly accurate and reliable** – The kernel can be extended in any “direction”. Adding new element types, new material models with any number of element types and internal history parameters, new boundary conditions (time-dependent, position-dependent, state-dependent, periodic and flow-in/out) or numerical algorithms (explicit and implicit) is possible, as well as the ability to add and manage arbitrary degrees of freedom is a matter of course. (It is not a Y-like code limited to constant-strain triangular elements, Q4 cohesive elements and triangle-triangle contacts; OpenFDEM is intended to be a more general FEM/DEM solver compatible with arbitrary scenarios.) Like other common open-source FEM solvers, the most important feature of OpenFDEM is its standardization and generality, which allows the continuum-discontinuum method to be used with more general scenarios.
- **Friendly preprocessing interface** - The Gmsh is provided to easily create meshes from computer-aided design (CAD), geometry file and third-party commercial software. The built-in commands are accessible to create many basic geometries (e.g. rectangular, circle, ellipse, polygon, line and particles) and initial discontinuities (e.g. single joint, joint sets, DFNs and DFNs from image mapping). This built-in mesh module can quickly assess mesh quality and identify the local bad meshes which will be further optimized by swap, node insertion, node delete, element split techniques, automatically or manually. Meanwhile, for very complex geometric models, users can also import mesh files from third-party software. OpenFDEM supports importing standard mesh files, such as .msh, .inp, .stl, .dxf and .step, etc. Principally, the geometry file can be processed by Gmsh can also be done by OpenFDEM.
- **Parallel processing support** - Most modules can be operated in parallel and very good performance scalability on various platforms. Instead of using no binary searching (NBS) contact method in OpenFDEM, a new element-wise contact searching algorithm with a complexity of $O(N \log N)$ is proposed to make the contact searching process parallelable. The GPU acceleration will also be added shortly.
- **Mesh adaptive analysis support** – Local adaptive mesh refinement (LAMR) and global adaptive mesh refinement(gAMR) are provided in OpenFDEM for mesh optimization and accuracy enhancement. Based on different remeshing criteria, OpenFDEM supports various error estimations, such as primary unknown, internal variables mapping, high-accuracy internal variable interpolation and fast unbalance equilibrium after refinement. The AMR supports fracture path consistent before and after remeshing.

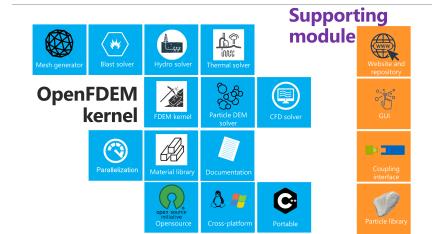
Fig. 3: Global adaptive mesh refinement (up-left) and local adaptive mesh refinement (down-right) in OpenFDEM.

Fig. 4: Flowing mesh refined based on the stress level in the bar.

- **Rich grain-based modelling support** – Voronoi tessellations can be created with the built-in Voronoi module. The optimization is deployed to match the laboratorial mineral distribution from measurements or digital image. The realistic grain-based model (GBM) can be reproduced directly in the project by inputting the binary sample images. The polygonal element type is available for representing the whole mineral individually. Furthermore, transgranular fracturing can be realized by element splitting techniques.

Fig. 5: OpenFDEM implements a Voronoi lib and is able to simulate grains directly based on polygonal elements.

- **Large material library including the state-of-the-art models for phase field of quasibrittle materials and rich element library** – currently, OpenFDEM supports 17 element materials (spanning elastic, hyperelastic, plastic, damage, nonlocal, viscous and phas—field models), 7 cohesive materials (spanning static, dynamic and fatigue problems), and 6 contact models (including Mohr-coulomb friction, hertz contact, rate friction, rough dilation shear law and so on). (The OpenFDEM is the only code supports materials beyond the community of rock mechanics)



| Capabilities | OpenFDEM | Other codes |
|---|----------|-------------|
| Ability to simulate the fracturing from continuum to discontinuum | ● | ● |
| Fracture material library, supporting over 24 materials | ● | ● |
| Time-dependent contact detection algorithm | ● | ● |
| Hydro model including gas flow | ● | ● |
| Thermal model and heat coupling (including latent thermal) | ● | ● |
| Fluid-solid interaction model and fluid dynamics | ● | ● |
| Blast and explosive DDS support | ● | ● |
| Grain-based model and image segmentation | ● | ● |
| Built-in module to create geometry, mesh and pre-processing | ● | ● |
| Parallelization (GPU, CPU and clusters) | ● | ● |
| Structure | ● | ● |

● OpenFDEM supports 17 element materials (spanning static, dynamic, plastic, damage, nonlocal, viscous and phase-field models) and 7 cohesive materials (spanning static, dynamic and fatigue problems). It contains 6 contact models (including Mohr-Coulomb friction, Hertz contact, rate friction, rough dilation shear law and so on).

● OpenFDEM uses explicit schemes and implicit methods. The explicit methods are used to solve contact which is a very stiff problem. The implicit methods are used to solve the maximum order of three to model the large deformation of the entity.

● Available for OpenFOAM, a very general kernel for complex problems.

● OpenFDEM has using US IN-FOAM after some modification. It is available at openfDEM.org.

Fig. 6: New material library in OpenFDEM.

- **Advanced Analysis Solvers** - Linear dynamic solver (implicit and explicit), linear static solver (PETSC), eigen-value problem (SLEPc), and nonlinear dynamic solver (explicit) are applicable for different problems (The OpenFDEM is the only project provides implicit solver for FDEM-like codes), the implicit solver currently can be run on Linux-like OS.

Particle Discrete Element Method (pDEM)

- **Rigid DEM support** - built-in module for rigid particles packing, kinematics and collision, the particle-based contact models include linear, Hertz, cohesive bond and rotation resistance model.

Fig. 7: Phasefield module in OpenFDEM.

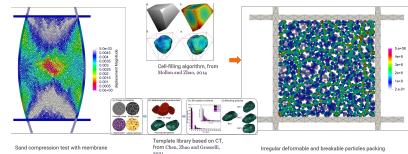


Fig. 8: Sand compression test with membrane (left) and irregular deformable and breakable particles packing (right).

- **Realistic Particle Modelling** - Overlapping particles and Fourier-Voronoi-based algorithm are used to generate realistic particles having complex shapes. The realistic particles can be rigid or deformable. The breakage of the particles are also possible.

Fig. 9: Debris flow of rigid particles due to gravity (left) and debris flow of irregular deformable fragments due to gravity (right, stl file from itasca).

Fluid Dynamic Module

- **Analysis Procedures** - Matrix flow for pore seepage, transient incompressible fracture flow, transient compressible fracture flow and gas flow problems.
- **Element Library** - Triangle, quadratic triangle, quadrilateral and quadratic quadrilateral element types are supported for Newtonian fluid and Bingham fluid.
- **Boundary Types** - Water level, pore pressure, flow rate, steady flow and impermeable boundary conditions are supported in hydro module.

Thermal Transportation Module

- **Analysis procedures** - Matrix thermal transportation, thermal resistance in fractures, heat conduction of fluid in fracture, heat advection of fluid, heat exchange between solid and fluid and contact thermal problems.

Fig. 10: Blast considering gas expansion, gas flow by hydro module (left) and without gas expansion by mechanical module (right).

Fig. 11: Fluid injection in fractured rock block.

- **Element Library** - Triangle, quadratic triangle, quadrilateral and quadratic quadrilateral element types are supported.
- **Boundary Types** - Constant temperature, flux, conduction, advection, radiation, source and adiabatic thermal conditions are supported.

Computational Fluid Dynamics

- **Material Point Method (MPM)** - is used to simulate the fluid transportation and large deformation. This mesh-free method does not encounter the drawbacks of mesh-based methods (high deformation tangling, advection errors etc.) which makes it a promising and powerful tool for large deformation problems. The coupling among FDEM and MPM makes the solid interacting with fluid is possible.

1.1 Post-Processing

- Export to VTK format for postprocessing on VTK based visualization tools such as ParaView is supported.
- Export to Tecplot format is supported.
- Export historic variables which are monitored at each step to .csv is supported.

1.2 Third-Party Packages Used in OpenFDEM

- GMSH - 2D and 3D mesh generator
- GSL - mathematical routines
- Eigen - matrix calculation
- PETSC - Portable, Extensible Toolkit for Scientific Computation
- ParaView - Parallel Visualization Application (for .vtk files)

Fig. 12: Water flowing in a tube using CFD in hydro module.

Fig. 13: Gabbro fracturing after microwave treatment in thermal module.

Fig. 14: Tapwater flow into a tank, MPM + FDEM.

1.3 Documentation

The documentation is auto-generated from the `.of` and `.rst` files throughout the codebase and the extensive comments in the source code `.h` and `.of` files. Sphinx is used to compile the documentation in HTML and PDF formats.

Fig. 15: Kármán vortex street example, periodic boundary.

SYSTEM REQUIREMENTS

2.1 Build Requirements

To compile OpenFDEM, you need a compiler supporting C++ 17 and the following packages are required:

- **CMake** (>=3.5.1)
- **OpenMP** - a high-performance, freely available package for multi core acceleration
- **Gmsh** - mesh generation and pre-processing, it is optional and the kernel is implemented in the source code (4.10)
- **Eigen** - a scientific matrix computation, it is optional and the headers are included in source code (>=3.4.0)

2.2 Post-Processing

To use the post-processing outputs (optional steps):

- **ParaView** - Parallel visualization application
- **Tecplot** - Commercial software for field results

2.3 Implicit Static/Non-Linear Solvers

To use the implicit static or nonlinear solvers, at least one of the following libraries is required:

- **PETSc** - portable, extensible toolkit for scientific computation
- **LAPACK** - a standard software library for numerical linear algebra

OpenFDEM is flexible and can be run on Windows or Linux-like systems. The released version is for Windows x64.

QUICK START FOR DEVELOPERS

This is the source code of C/C++ based OpenFDEM project developed by Dr. Xiaofeng Li, since 2017. This project is not limited to an FDEM solver for continuum-discontinuum problems, but is also capable of solving particulate DEM, material point method and phasefield problems.

The project can be complied by Visual Studio (> 2015) and compatible with Windows 7, 10 or Linux-like systems.

Tutorial examples can be found in `..src\test\...`. The main file is located at `src\solve\openfdem.cpp`. OpenFDEM is run by parsing the input file.

General steps to run OpenFDEM models:

1. Open `.sln` project in `\openfdem\src\of\OpenFDEM\OpenFDEM.sln` by your local Visual Studio software.
2. Compile the project in Visual Studio from the `src\solve\openfdem.cpp` main file.
3. The executable code should be in `x64\Release` (or `Debug`), be sure to keep the `.dll` files in the same folder with `.exe` file.
4. Drag the `.of` file into the `.exe` software then the model will starts to run, or use `openfdem example.of` in terminal to run a model.

3.1 Source Code Download

The source code is hosted on the University of Toronto Gitlab: [OpenFDEM Gitlab](#).

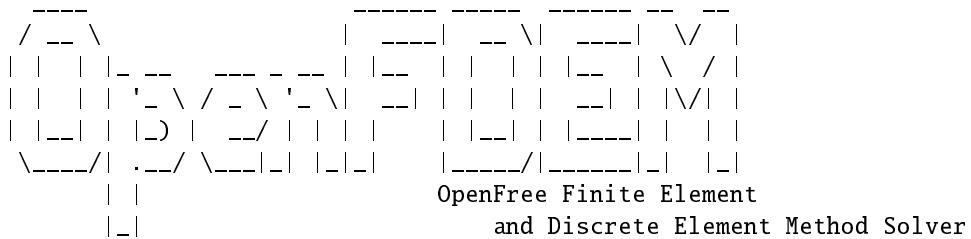
3.2 System Requirements

- Windows x64
- Visual Studio (> 2015)

All external dependencies are included in the OpenFDEM download.

**CHAPTER
FOUR**

COPYRIGHT



OpenFDEM : Object Oriented Open Free Finite Discrete Element Code

Copyright (C) 2017 - 2022 Xiaofeng Li
Email: xfli@whrsm.ac.cn

OpenFDEM Project Website: <https://openfdem.com/>
Geomechanics Group Website: <https://geogroup.utoronto.ca/>

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

TUTORIALS

5.1 Tutorial 0: Create Geometry

Uniaxial compression test is one of the most fundamental tests that will be used to get the material properties from the experimental stress-strain curve.

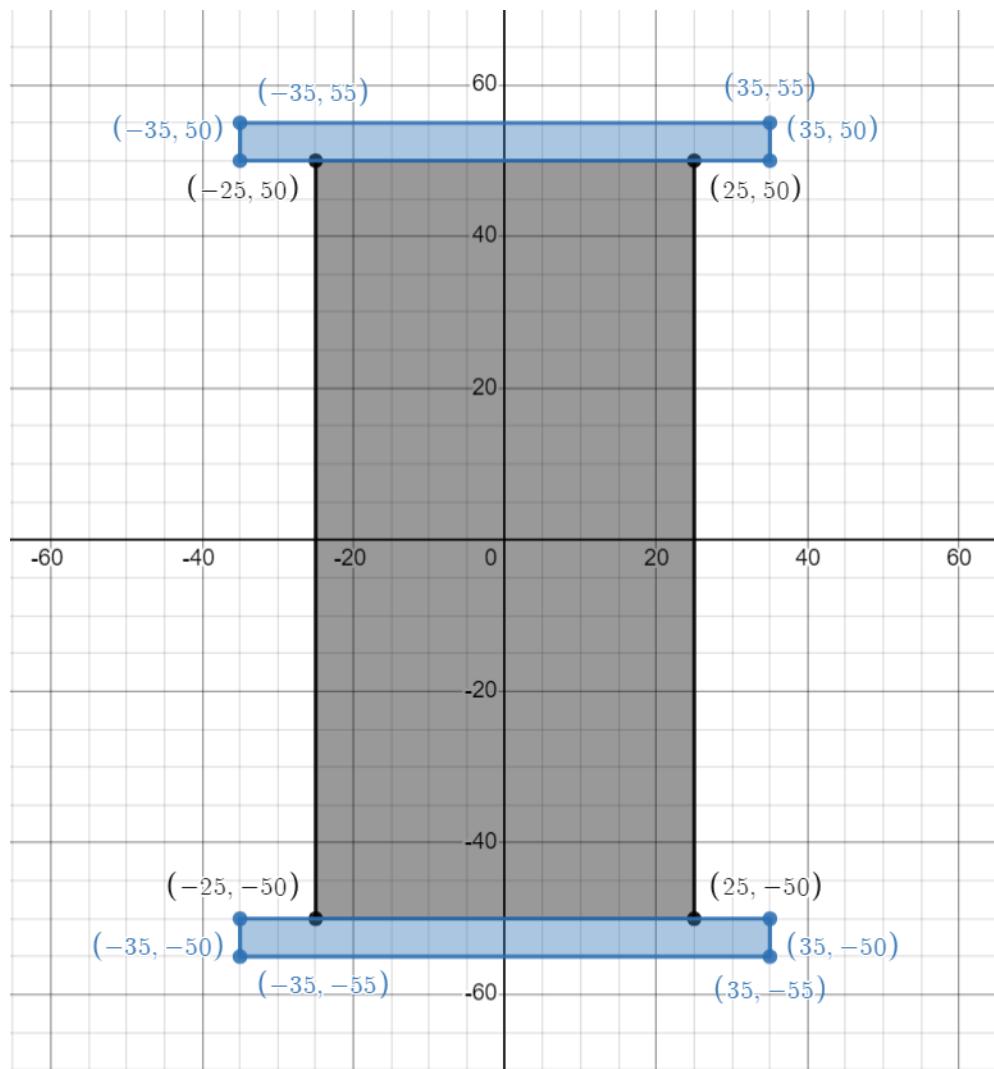


Fig. 1: Figure 1: Geometry of the uniaxial compression strength test

5.1.1 1.0 Main Steps

1. Initialize the model.
2. Create the geometry and the mesh
3. Assign material properties.
4. Create the group and assign boundary conditions.
5. Set the outputs.

5.2 Tutorial 1: Create Uniaxial Compression Test (UCS)

Uniaxial compression test is one of the most fundamental tests that will be used to get the material properties from the experimental stress-strain curve.

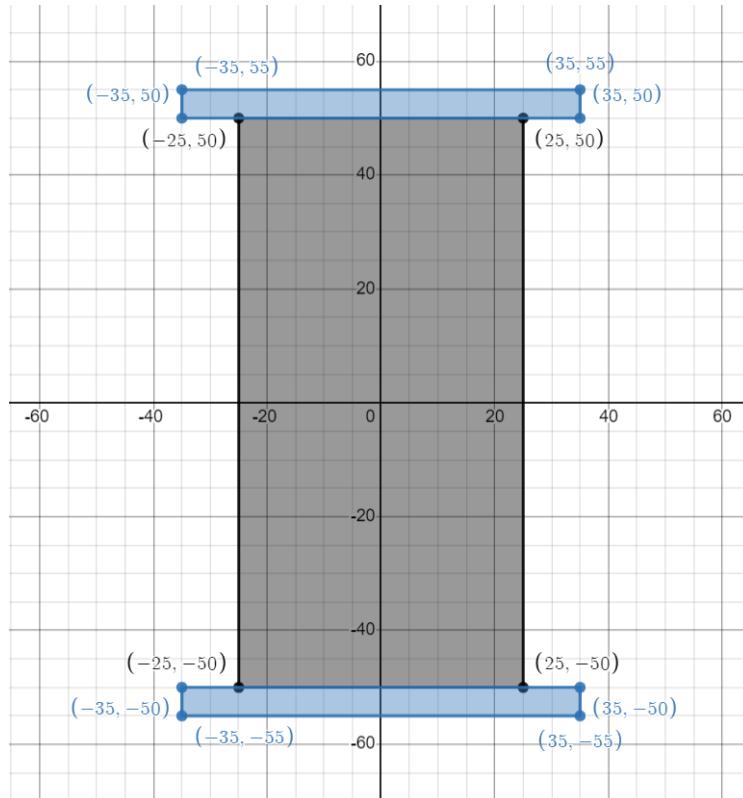


Fig. 2: Figure 1: Geometry of the uniaxial compression strength test

5.2.1 1.0 Main Steps

1. Initialize the model.
2. Create the geometry and the mesh
3. Assign material properties.
4. Create the group and assign boundary conditions.
5. Set the outputs.

5.2.2 2.0 Codes

Please note that, all the setting in the example is based on the developer's computer. You may need to change these setting based on your own conditions.

To start with programming, create a new empty text file (later add the *.of* extension). Begin writing the following commands:

2.1 Initialize the model

1. Create a new run and clean up the old memories.

```
of.new
```

2. Set up the folder name to save your results. Here the folder name is set to be "result".

```
of.set.result "result"
```

3. Set the interval of logging in the log file. This example uses the default interval 2000 steps.

```
of.log.interval 2000
```

4. Set the number of cores to be used for running this model.

Warning: In this example, 15 cores will be used for running. To prevent the crushing of your computer, please check your computer to fill in the number of cores you want to use from your computer. To check the number of cores your computer has, you can go Task Manager > Performance > CPU > Cores. If nothing is set here, serial computing will be used.

```
of.set.omp 15
```

2.2 Create the geometry and the mesh

1. See Figure 1 for the UCS test setup and draw the corresponding geometry.

Note:

The following three methods can be used to create the geometry:

1. xmin x_minvalue xmax x_maxvalue ymin y_minvalue ymax y_maxvalue
2. x [minvalue, maxvalue] y [minvalue, maxvalue]

3. [x_minvalue, x_maxvalue, y_minvalue, y_maxvalue]

This example used all three methods for demonstration. Users may choose any format to create geometries.

```
of.geometry.square 'rock' xmin -25e-3 xmax 25e-3 ymin -50e-3 ymax 50e-3
of.geometry.square 'up' x [-35e-3, 35e-3] y [50e-3, 55e-3]
of.geometry.square 'down' [-35e-3 35e-3 -55e-3 -50e-3]
```

2. Set the mesh size of the geometry and insert cohesive interfaces.

```
of.geometry.mesh.size 'default' 0.8e-3
of.geometry.mesh delaunay
of.mesh.insert 'rock'
```

2.3 Assign Material Properties

1. Set the material properties. For this test, the example uses the elastic constitutive model and assigned material for CZM and contact.

```
of.mat.element 'default' elastic den 2700 E 30e9 v 0.3 damp 1.0
of.mat.cohesive 'default' EM ten 1e6 coh 3e6 fric 0.3 GI 10 GII 50
of.mat.contact 'default' MC fric 0.3
```

2.4 Create Groups and Assign Boundary Conditions

1. Group the nodes to prepare for the next step.

```
of.group.nodal.from.element 'up' 'up'
of.group.nodal.from.element 'down' 'down'
```

2. Assign the nodal boundaries. For uniaxial test, top and bottom platens are moving toward the sample to compress the sample, so platens are not moving in the x direction and moving with 0.05 m/s in the y direction.

```
of.boundary.nodal.velocity 'up' x 0.0 y -0.05
of.boundary.nodal.velocity 'down' x 0.0 y 0.05
```

2.5 Set the Outputs

1. Set the output interval to be every 50000 steps and output all fields variables and fracture variables. Control the output interval to a reasonable size could shorten the computation time but get a good understanding of the model.

```
of.history.pv.interval 5000
of.history.pv.field default
of.history.pv.fracture default
```

2. Timestep can be fixed to 5e-9 in this example to accelerate the calculation in this example, but the default time step is recommended to use.

```
of.timestep fix 5e-9
```

3. The program will run 500000 steps in total. In other words, it will output 10 files for reference.

```
of.step 500000
```

4. Finalize the model and clear all the temporary memories.

```
of.finalize
```

5. Save the notepad and double click the .of file to run the program.

5.2.3 3.0 Run the Program

When you run the program, you can first check the mesh that was created by Gmsh as shown in Figure 2. If the mesh has a good quality, you can close the window to continue run the program.

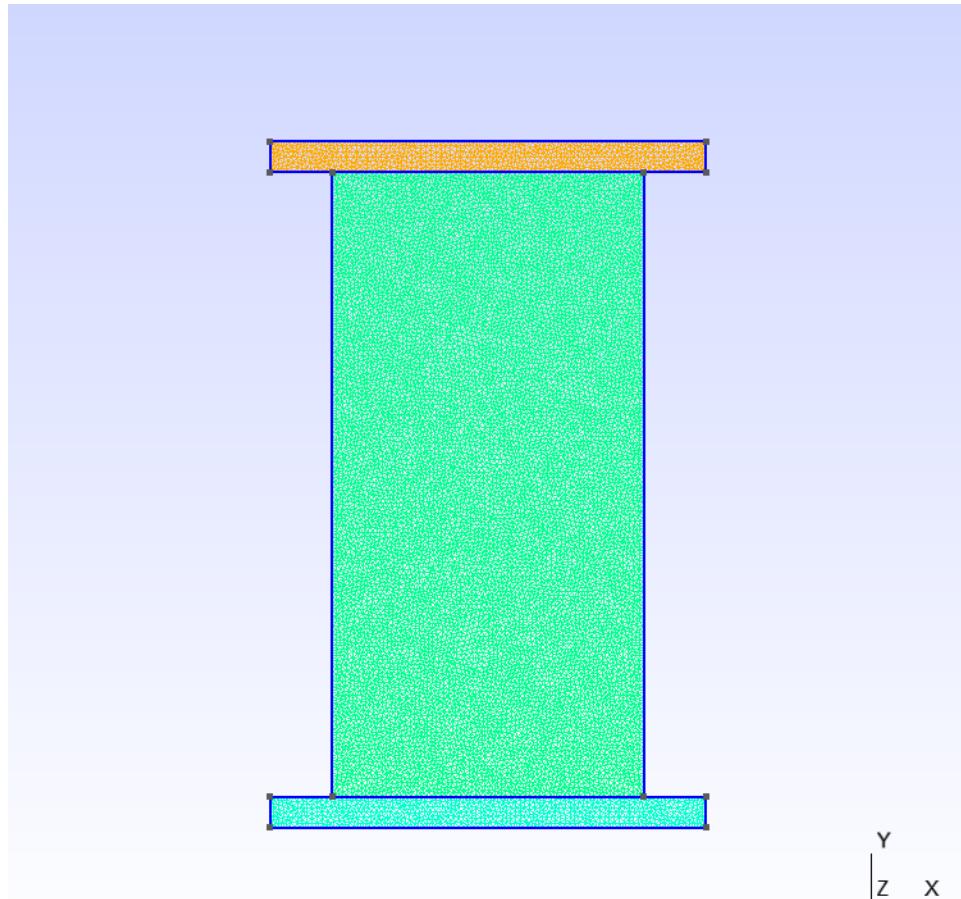


Fig. 3: Figure 2: Mesh created by Gmsh

The mesh is using triangle elements.

| group ID | Type | Number of Object | Tags | Extra |
|-----------------|---------|------------------|------|--------------|
| 0 | Node | 782 | up | |
| 1 | Node | 773 | down | |
| 2 | Element | 20622 | rock | Triangle, T3 |
| 3 | Element | 1372 | up | Triangle, T3 |
| 4 | Element | 1354 | down | Triangle, T3 |
| Total nodes | | 63421 | | |
| Total elements | | 23348 | | |
| Total cohesives | | 31121 | | |
| Total entities | | 20624 | | |

Fig. 4: Figure 3: Check the mesh setting from command window

Material properties you set on step 2.3 will be shown on the screen. You can confirm it while the program just starts to run.

| Checking model material information ... | | | | |
|---|----------------------------|-------------------|---------|--|
| ID | Type | Model type | Tags | Properties |
| 0 | Matrix material | Linear elastic | default | E = 30.00 GPa v = 0.30 density = 2700.00 kg/m^3 damp = 1.00 |
| 1 | Cohesive material | Evans-Marathé FPZ | default | pn = 500.00 GPa pt = 230.77 GPa tension = 1.00 MPa cohesion = 3.00 MPa GI = 10.00 J/m^2 GII = 50.00 J/m^2 friction = 0.30 |
| 2 | Contact material (default) | Mohr-Coulomb law | default | kn = 500.00 GPa, ks = 230.77 GPa friction = 0.30 |

Fig. 5: Figure 4: Check the material assignment from command window

Furthermore, node boundaries are shown at the header of the program.

| Checking model boundary information ... | | | | |
|---|------------------|---------------|------------|-----------------------|
| Bou ID | Category objects | Boundary type | Group Tags | Info and Extra |
| 0 | Node boundary | Velocity | up | x fixed y = -0.05 m/s |
| 1 | Node boundary | Velocity | down | x fixed y = 0.05 m/s |

Fig. 6: Figure 5: Check the boundary information from command window

Global timestep is shown here.

```

Info    :critical timestep from MATRIX is: [1.23e-07].
Info    :critical timestep from INTERFACE is: [1.83e-10].
Info    :critical timestep from CONTACT is: [1.4le-08].
Info    :Warning: the user-defined timestep is greater than the critical timestep required.
Info    :global timestep is: [5.00e-09].

```

Fig. 7: Figure 6: Check the timesteps from command window

Start to run the program. Results are logged for every 2000 steps.

| Start running ... | | | | | | |
|-------------------|------------------|--------------------|---------------|------------|---------------|-------------|
| step | physical time(s) | unbalance force(N) | kinetic ratio | broken CZM | contact pairs | run time(s) |
| 0 | 0.000e+00 | 0.000e+00 | 0.000e+00 | 0 | 621 | 0.000 |
| 2000 | 1.000e-05 | 2.846e+02 | 2.426e-04 | 0 | 626 | 29.437 |
| 4000 | 2.000e-05 | 3.472e+02 | 4.435e-04 | 0 | 630 | 60.029 |

Fig. 8: Figure 7: Start to run the program

5.2.4 4.0 Results

The stress and displacement results here are for reference.

5.2.5 5.0 Full Script

- UCS_test.of (click to download from GitHub)

```

# initialization, this command is to clear the memory in your last run, it is
# not mandatory
# but strongly recommend.
of.new

# Set the path folder of your output results, //result// in the same path of
# your input file
# will be created by default
of.set.result "result"

# Set the interval of logging in the log file, 2000 is by default, not
# mandatory
of.log.interval 2000

# Set the number of cores you want to use, the parallelization will be
# automatically turned on

```

(continues on next page)

(continued from previous page)

```

# when you use the following command, otherwise the serialization will be used
# by default
of.set.omp 15

#####
# Create a box having the group tag 'rock', the box is defined by the minimum
# and maximum
# coordinations of each edge.
of.geometry.square 'rock' xmin -25e-3 xmax 25e-3 ymin -50e-3 ymax 50e-3
# Create the upper platen
of.geometry.square 'up' x [-35e-3, 35e-3] y [50e-3, 55e-3]
# Create the down platen.
of.geometry.square 'down' [-35e-3 35e-3 -55e-3 -50e-3]
# Assign the mesh size
of.geometry.mesh.size 'default' 0.8e-3
# to execute the kernel for meshing, the kernel will automatically export a
# mesh file named
# "mess.msh" in the input folder. Users can of.import "mesh.msh" to run the
# model again
# without recreating the mesh
of.geometry.mesh delaunay

# Insert the cohesive elements.
of.mesh.insert 'rock'

#####
# Assign material parameters
# Assign material for matrix.
of.mat.element 'default' elastic den 2700 E 30e9 v 0.3 damp 1.0
# Assign material for CZM
of.mat.cohesive 'default' EM ten 1e6 coh 3e6 fric 0.3 GI 10 GII 50
# Assign materials for contacts
of.mat.contact 'default' MC fric 0.3

#####
# OpenFDEM can manually group the nodes, elements, cohesive elements and edges
# by using the
# region of box, circle and plane.
of.group.nodal.from.element 'up' 'up'
of.group.nodal.from.element 'down' 'down'

#####
# assign boundaries
# boundaries can be assigned after you define the groups
of.boundary.nodal.velocity 'up' x 0.0 y -0.05
of.boundary.nodal.velocity 'down' x 0.0 y 0.05

#####
# set output
# Set the interval of writing ParaView results.

```

(continues on next page)

(continued from previous page)

```

of.history.pv.interval 5000
# Output all results by default.
of.history.pv.field default
of.history.pv.fracture default

#####
# Manually set the timestep to overwirte the default timestep suggested by the
# It is not recommended to manually increase the size of timesteps unless the
# user has enough
# understanding of the calculation process.
of.timestep fix 5e-9
# to excuate the model
of.step 500000
# finalize the model and clear the memory
of.finalize

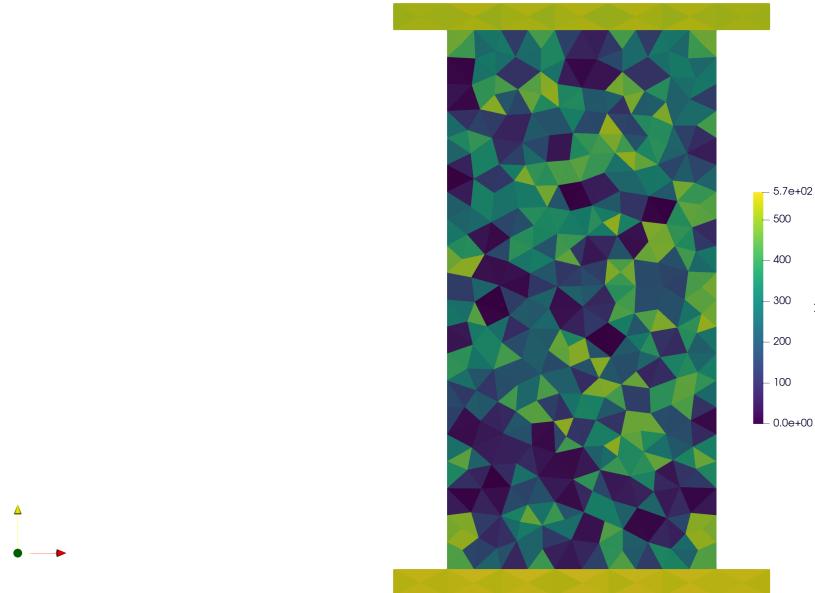
```

5.3 Tutorial 2: UCS model from Gmsh

OpenFDEM also allows users to import common mesh files such as .inp files, .msh files, and .geo files to run the model. This example will review how to setup a uniaxial compressive test example with output from Gmsh software.

Runtime: <10 min on i9 8-core Windows 10 Machine

Expected tutorial output (visualized in ParaView):



Note: OpenFDEM supports four main mesh pre-processing approaches:

-
1. A user defined command in OpenFDEM to create mesh automatically
 2. Importing a .geo file
 3. Importing a .msh file
 4. Mesh developed from other commercial softwares, including .inp, .dxf, .fdem, .tess (for grain-based model only) and .jpg (for grain-based module and DFN module only).
-

5.3.1 1.0 Tutorial Prerequisites

The following files are needed to follow along the tutorial:

- [Job-3.inp](#) (click to download from GitHub gist)

5.3.2 2.0 Codes

1. OpenFDEM supports various common mesh file format such as .inp files, .msh files, and .geo files. In this example, “Job-3.inp” file in the folder will be imported to the model, which has the same mesh as the first unconfined compression test example.

```
of.import "Job-3.inp"
```

2. Given input file was exported from ABAQUS without assigning groups, so users need to group elements manually here.

```
of.group.element 'up' range box.in [-35e-3 35e-3 50e-3 55e-3]
of.group.element 'rock' range box.in [-35e-3 35e-3 -50e-3 50e-3]
of.group.element 'down' range box.in [-35e-3 35e-3 -55e-3 -50e-3]
```

5.3.3 3.0 Full Script

- [UCS_test_Gmsh.of](#) (click to download from Gitlab)

```
# initialization, this command is to clear the memory in your last run, it is not
# mandatory
# but strongly recommend.
of.new

# Set the path folder of your output results, //result// in the same path of your input
# file
# will be created by default
of.set.result "result"

# Set the interval of logging in the log file, 2000 is by default, not mandatory
of.log.interval 2000

# Set the number of cores you want to use, the parallelization will be automatically
# turned on
# when you use the following command, otherwise the serialization will be used by default
of.set.omp 15
```

(continues on next page)

(continued from previous page)

```
#####
# import mesh #####
#
# Openfdem supports import common mesh files such as .inp .msh .geo and others
of.import "Job-3.inp"

# Given input file was exported from ABAQUS without assigning groups, so users need to
# group
# elements manually here.
of.group.element 'up' range box.in [-35e-3 35e-3 50e-3 55e-3]
of.group.element 'rock' range box.in [-35e-3 35e-3 -50e-3 50e-3]
of.group.element 'down' range box.in [-35e-3 35e-3 -55e-3 -50e-3]
# Insert the cohesive elements.
of.mesh.insert 'rock'

#####
# assign material parameters #####
#
# Assign material for matrix.
of.mat.element 'default' elastic den 2700 E 30e9 v 0.3 damp 2.0
# Assign rigid material to the two plattens
of.mat.element 'up' rigid den 2700
of.mat.element 'down' rigid den 2700
# assign material for CZM
of.mat.cohesive 'default' EM ten 1e6 coh 3e6 fric 0.3 GI 10 GII 50
# assign materials for contacts
of.mat.contact 'default' MC fric 0.3

#####
# create groups #####
#
# OpenFDEM can manually group the nodes, elements, cohesive elements and edges by using
# the
# region of box, circle and plane.
of.group.nodal.from.element 'up' 'up'
of.group.nodal.from.element 'down' 'down'

#####
# assign boundaries #####
#
# boundaries can be assigned after you define the groups
of.boundary.nodal.velocity 'up' x 0.0 y -0.05 r 0.0
of.boundary.nodal.velocity 'down' x 0.0 y 0.05 r 0.0

#####
# set output #####
#
# Set the interval of writing ParaView results.
of.history.pv.interval 5000
# Output all results by default.
of.history.pv.field default
of.history.pv.fracture default

#####
# execute model #####
#
# Manually set the timestep to overwirte the default timestep suggested by the program.
# It is not recommended to manually increase the size of timesteps unless the user has
```

(continues on next page)

(continued from previous page)

```
→ enough
# understanding of the calculation process.
of.timestep fix 5e-9
# to excuate the model
of.step 500000
# finalize the model and clear the memory
of.finalize
```

5.4 Tutorial 3: Create Uniaxial Compression Test with Quadrangle Element

By default, triangle elements will be used in the mesh and model. However, OpenFDEM also supports the quadrangle elements.

5.4.1 1.0 Code

To run the model with quadrangle elements, users just need to add the following line after setting up the mesh size and before creating the mesh.

```
of.geometry.recombine 'rock'
```

5.4.2 2.0 Run the Program

When the program runs, the mesh of the model will look like this.

The program will also show the mesh has quadrilateral elements.

5.4.3 3.0 Full Script

- UCS_test_quad.of (click to download from Gitlab)

```
# initialization, this command is to clear the memory in your last run, it is
→not mandatory
# but strongly recommend.
of.new

# Set the path folder of your output results, //result// in the same path of
→your input file
# will be created by default
```

(continues on next page)

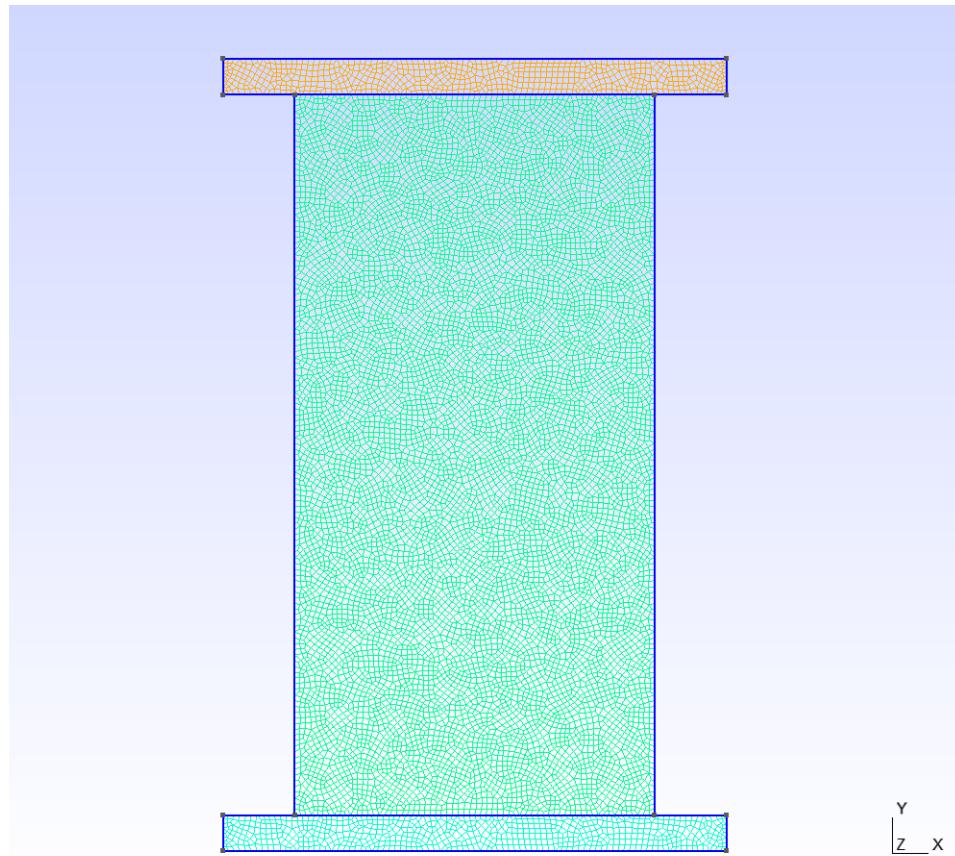


Fig. 9: Figure 1: Uniaxial Compression Test Mesh with Quadrangle Elements

| Checking model mesh information ... | | | | |
|-------------------------------------|---------|------------------|-------|-------------------|
| group ID | Type | Number of Object | Tags | Extra |
| 0 | Node | 766 | up | Rigid |
| 1 | Node | 767 | down | Rigid |
| 2 | Element | 10250 | rock | Quadrilateral, Q4 |
| 3 | Element | 669 | up | Quadrilateral, Q4 |
| 4 | Element | 670 | down | Quadrilateral, Q4 |
| | | Total nodes | 42533 | |
| | | Total elements | 11589 | |
| | | Total cohesives | 20690 | |
| | | Total entities | 10252 | |

Fig. 10: Figure 2: Uniaxial Compression Test Mesh with Quadrangle Elements

(continued from previous page)

```

of.set.result "result"

# Set the interval of logging in the log file, 2000 is by default, not
# mandatory
of.log.interval 2000

# Set the number of cores you want to use, the parallelization will be
# automatically turned on
# when you use the following command, otherwise the serialization will be used
# by default
of.set.omp 15

##### create mesh #####
# Create a box having the group tag 'rock', the box is defined by the minimum
# and maximum
# coordinations of each edge.
of.geometry.square 'rock' xmin -25e-3 xmax 25e-3 ymin -50e-3 ymax 50e-3
# Create the upper platten
of.geometry.square 'up' x [-35e-3, 35e-3] y [50e-3, 55e-3]
# Create the down platten.
of.geometry.square 'down' [-35e-3 35e-3 -55e-3 -50e-3]
# Assign the mesh size
of.geometry.mesh.size 'default' 0.8e-3
# combine the delaunay to quadrangle element types
of.geometry.recombine 'rock'
# to execute the kernel for meshing, the kernel will automatically export a
# mesh file named
# "mess.msh" in the input folder. Users can of.import "mesh.msh" to run the
# model again
# without recreating the mesh
of.geometry.mesh delaunay

# Insert the cohesive elements.
of.mesh.insert 'rock'

##### assign material parameters #####
# Assign material for matrix.
of.mat.element 'default' elastic den 2700 E 30e9 v 0.3 damp 2.0
# Assign rigid material to the two plattens
of.mat.element 'up' rigid den 2700
of.mat.element 'down' rigid den 2700
# assign material for CZM
of.mat.cohesive 'default' EM ten 1e6 coh 3e6 fric 0.3 GI 10 GII 50
# assign materials for contacts
of.mat.contact 'default' MC fric 0.3

##### create groups #####
# OpenFDEM can manually group the nodes, elements, cohesive elements and edges
# by using the

```

(continues on next page)

(continued from previous page)

```

# region of box, circle and plane.
of.group.nodal.from.element 'up' 'up'
of.group.nodal.from.element 'down' 'down'

#####
##### assign boundaries #####
# boundaries can be assigned after you define the groups
of.boundary.nodal.velocity 'up' x 0.0 y -0.05 r 0.0
of.boundary.nodal.velocity 'down' x 0.0 y 0.05 r 0.0

#####
##### set output #####
# Set the interval of writing ParaView results.
of.history.pv.interval 5000
# Output all results by default.
of.history.pv.field default
of.history.pv.fracture default

#####
##### execute model #####
# Manually set the timestep to overwirte the default timestep suggested by the
# program.
# It is not recommended to manually increase the size of timesteps unless the
# user has enough
# understanding of the calculation process.
of.timestep fix 5e-9
# to excuate the model
of.step 500000
# finalize the model and clear the memory
of.finalize

```

5.5 Tutorial 4: Brazilian Disc Test

Brazilian disc test is an indirect test method for assessing the tensile strength.

5.5.1 1.0 Tutorial Prerequisites

The following files are needed to follow along the tutorial:

- [example_br.geo](#) (click to download from Gitlab)

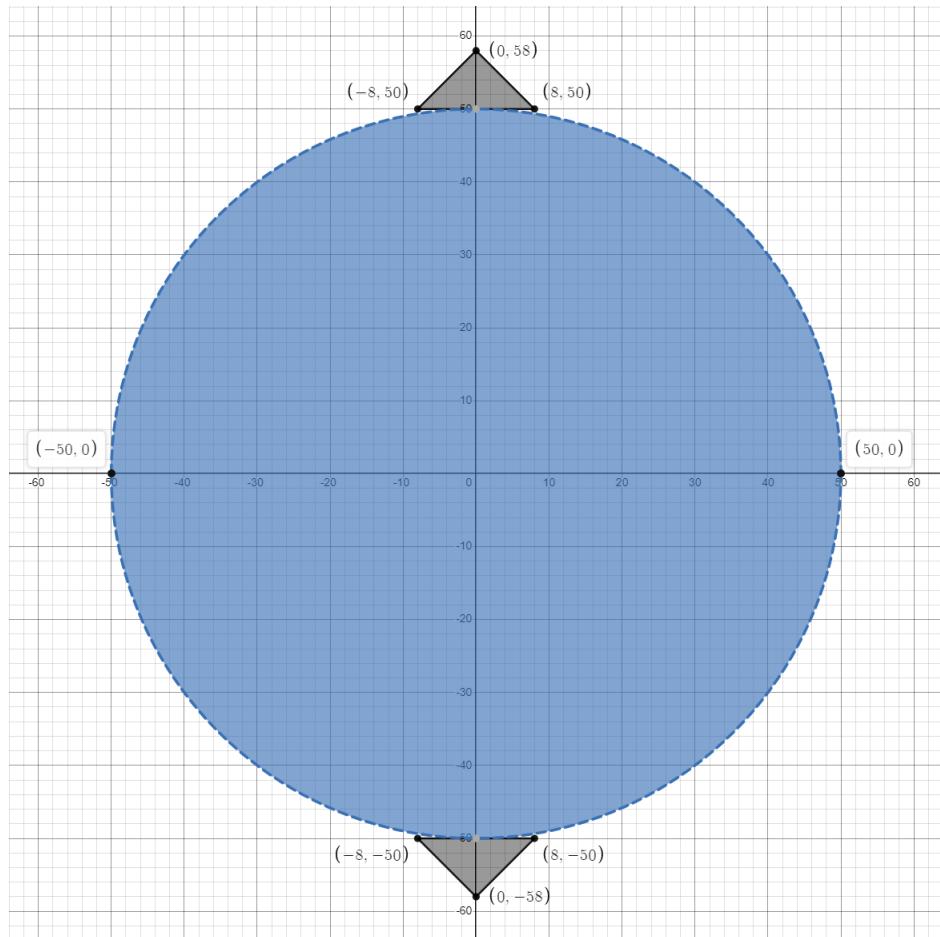


Fig. 11: Figure 1: Geometry of the Brazilian disc test

5.5.2 2.0 Main Steps

1. Initialize the model.
2. Import the geometry and the mesh
3. Assign material properties.
4. Create the group
5. Assign boundary conditions.
6. Set the outputs.

5.5.3 3.0 Codes

Please note that, all the setting in the example is based on the developer's computer. You may need to change these setting based on your own conditions.

To start with programming, create a new empty text file (later add the *.of* extension). Begin writing the following commands:

3.1 Initialize the model

1. Create a new run and clean up the old memories.

```
of.new
```

2. Set up the folder name to save your results. Here the folder name is set to be “result”.

```
of.set.result "result"
```

3. Set the number of cores to be used for running this model.

Warning: In this example, 15 cores will be used for running. To prevent the crushing of your computer, please check your computer to fill in the number of cores you want to use from your computer. To check the number of cores your computer has, you can go Task Manager > Performance > CPU > Cores. If nothing is set here, serial computing will be used.

```
of.set.omp 15
```

3.2 Import the geometry and the mesh

1. Import the geometry and mesh of the model from the “example_br.geo” file. The final geometry will be the same as Figure 1 shown.

```
of.import 'example_br.geo'
```

2. Group the elements to rock, top platen and bottom platen.

```
of.group.element 'rock' range box.in xmin -50.0 xmax 50.0 ymin -50.0 ymax 50.0
of.group.element 'up' range box.in xmin -500.0 xmax 500.0 ymin 50.0 ymax 100.0
of.group.element 'down' range box.in xmin -500.0 xmax 500.0 ymin -500.0 ymax -
→50.0
```

3. Insert the cohesive elements to the mesh.

```
of.mesh.insert 'rock'
```

3.3 Assign Material Properties

1. Set the material properties. For this test, the example uses the elastic constitutive model and assigned material for CZM and contact.

```
of.mat.element 'rock' elastic den 2700 E 30e9 v 0.3 damp 0.6
of.mat.element 'up' rigid den 7000
of.mat.element 'down' rigid den 7000
of.mat.cohesive 'default' EM ten 1e6 coh 3e6 fric 0.3 GI 10 GII 50
of.mat.contact 'default' MC fric 0.3
```

3.4 Create Groups and Assign Boundary Conditions

1. Group the nodes to prepare for the next step.

Note:

There are three methods to group nodals in the mesh.

1. box.in/box.on/box.out x [x_min x_max] y [y_min y_max] or [x_min x_max y_min y_max]
 2. circle.in/circle.on/circle.out center [x y] radius rad or [x y radius]
 3. plane.on/plane.above/plane.below p0 [x0 y0] p1 [x1 y1] or [x0 y0 x1 y1]
-

```
of.group.nodal 'up' range box.in xmin -100.0 xmax 100.0 ymin 50.0 ymax 500.0
of.group.nodal 'down' range box.in xmin -100.0 xmax 100.0 ymin -500.0 ymax -50.0
```

2. Assign the nodal boundaries. For uniaxial test, top and bottom platens are moving toward the sample to compress the sample, so platens are not moving in the x direction and moving with 0.05 m/s in the y direction.

```
of.boundary.nodal.velocity 'up' x 0.0 y -0.005
of.boundary.nodal.velocity 'down' x 0.0 y 0.005
```

3.5 Set the Outputs

1. Set the output interval to be every 50000 steps and output all fields variables and fracture variables. Control the output interval to a reasonable size could shorten the computation time but get a good understanding of the model.

```
of.history.pv.interval 2000
of.history.pv.field default
of.history.pv.fracture default
```

2. The program will run 500000 steps in total. In other words, it will output 10 files for reference.

```
of.step 500000
```

3. Finalize the model and clear all the temporary memories.

```
of.finalize
```

4. Save the notepad and double click the .of file to run the program.

5.5.4 4.0 Run the Program

When you run the program, you can first check the mesh that was created by Gmsh as shown in Figure 2. If the mesh has a good quality, you can close the window to continue run the program.

5.5.5 5.0 Results

The stress and displacement results here are for reference.

5.5.6 6.0 Full Script

- BD_test.of (click to download from GitHub gist)

```
# initialization, this command is to clear the memory in your last run, it is
# not mandatory
# but strongly recommend.
of.new

# Set the path folder of your output results, //result// in the same path of
# your input file
# will be created by default
of.set.result "result"

# Set the number of cores you want to use, the parallelization will be
# automatically turned on
# when you use the following command, otherwise the serialization will be used
# by default
of.set.omp 15
```

(continues on next page)

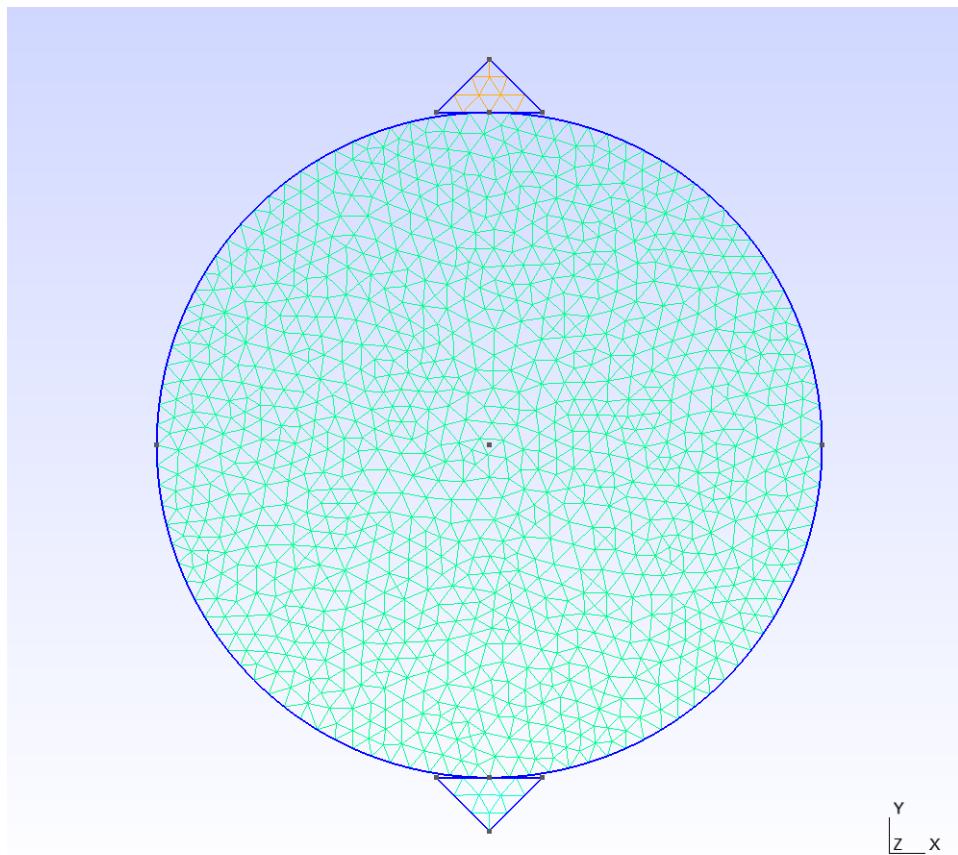


Fig. 12: Figure 2: Mesh created by Gmsh

(continued from previous page)

```

#####
# import mesh #####
# import .geo file
of.import 'example_br.geo'

# group elements
of.group.element 'rock' range box.in xmin -50.0 xmax 50.0 ymin -50.0 ymax 50.0
of.group.element 'up' range box.in xmin -500.0 xmax 500.0 ymin 50.0 ymax 100.0
of.group.element 'down' range box.in xmin -500.0 xmax 500.0 ymin -500.0 ymax -
→50.0

# Insert the cohesive elements.
of.mesh.insert 'rock'

#####
# assign material parameters #####
# Assign material for matrix.
of.mat.element 'rock' elastic den 2700 E 30e9 v 0.3 damp 0.6
of.mat.element 'up' rigid den 7000
of.mat.element 'down' rigid den 7000
# assign material for CZM
of.mat.cohesive 'default' EM ten 1e6 coh 3e6 fric 0.3 GI 10 GII 50
# assign materials for contacts
of.mat.contact 'default' MC fric 0.3

#####
# create groups #####
# OpenFDEM can manually group the nodes, elements, cohesive elements and edges,
# by using the
# region of box, circle and plane.
of.group.nodal 'up' range box.in xmin -100.0 xmax 100.0 ymin 50.0 ymax 500.0
of.group.nodal 'down' range box.in xmin -100.0 xmax 100.0 ymin -500.0 ymax -50.
→0

#####
# assign boundaries #####
# boundaries can be assigned after you define the groups
of.boundary.nodal.velocity 'up' x 0.0 y -0.005
of.boundary.nodal.velocity 'down' x 0.0 y 0.005

#####
# set output #####
# Set the interval of writing ParaView results.
of.history.pv.interval 2000
# Output all results by default.
of.history.pv.field default
of.history.pv.fracture default

#####
# execute model #####
# to excuate the model

```

(continues on next page)

(continued from previous page)

```
of.step 500000
# finalize the model and clear the memory
of.finalize
```

5.6 Tutorial 5: Apply In-situ Stress

In-situ stress is applied to model the original state of the rock before excavation or disturbance.

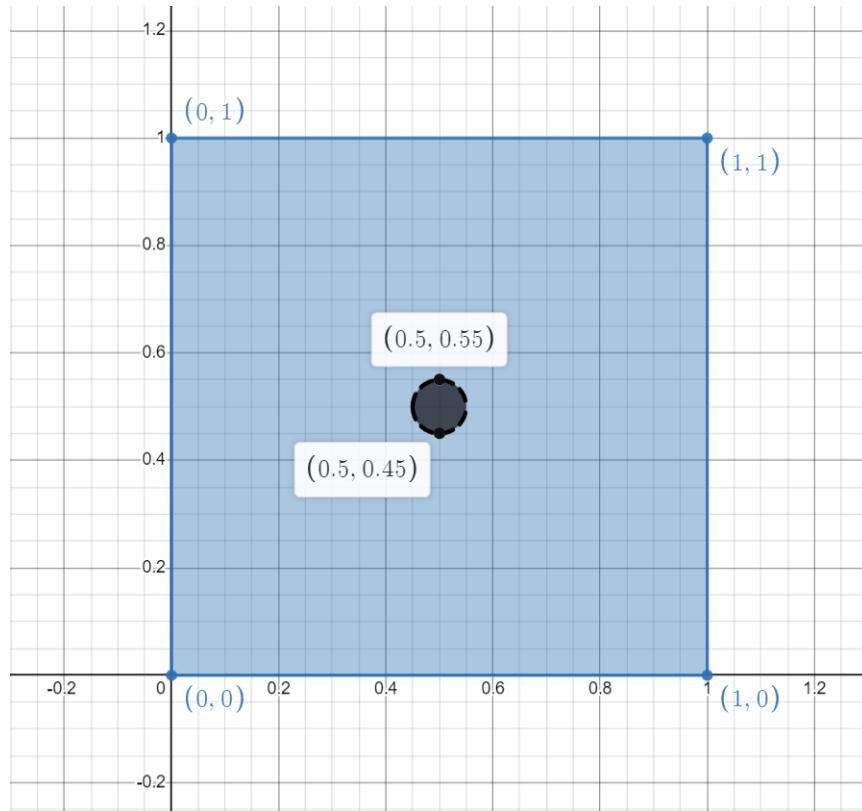


Fig. 13: Figure 1: Geometry of the Rock and Excavation

5.6.1 1.0 Main Steps

1. Initialize the model
2. Import mesh
3. Create groups
4. Assign material properties
5. Assign boundaries

6. Setup output
7. Execute model

5.6.2 2.0 Codes

Please note that, all the setting in the example is based on the developer's computer. You may need to change these setting based on your own conditions.

To start with programming, create a new empty text file (later add the `.of` extension). Begin writing the following commands:

2.1 Initialize the model

1. Create a new run and clean up the old memories.

```
of.new
```

2. Set up the folder name to save your results. Here the folder name is set to be “result”.

```
of.set.result "result"
```

3. Set the number of cores to be used for running this model.

Warning: In this example, 15 cores will be used for running. To prevent the crushing of your computer, please check your computer to fill in the number of cores you want to use from your computer. To check the number of cores your computer has, you can go Task Manager > Performance > CPU > Cores. If nothing is set here, serial computing will be used.

```
of.set.omp 15
```

2.2 Create the geometry and the mesh

1. See Figure 1, we will create a square rock model with a 0.05 m radius hole at the center of the rock.

Note: Circles in OpenFDEM are defined by four values: x coordinate, y coordinate, radius and the number of segments. Users can define a circle by entering `[x_value y_value radius #_of_segements]`.

```
of.geometry.square 'rock' [0 1 0 1]
of.geometry.cut.circle 'hole' 'rock' [0.5 0.5 0.05 100]
```

2. Set the mesh size and the default method to create the mesh

```
of.geometry.mesh.size 'default' 0.03
of.geometry.mesh auto
```

3. Group the elements which will be excavated later.

```
of.group.element 'excavation' range circle.in [0.5 0.5 0.05]
```

2.3 Assign Material Properties

Set the material properties as shown below.

```
of.mat.element 'default' elastic den 2700 E 30e9 v 0.3 damp 1.0  
of.mat.contact 'default' MC fric 0.3
```

2.4 Group the Edges

Group the four boundaries of the model for applying the in-situ stresses and boundary conditions later.

```
of.group.edge 'bottom' range plane.on [0.0 0.0 1.0 0.0]  
of.group.edge 'up' range plane.on [0.0 1.0 1.0 1.0]  
of.group.edge 'left' range plane.on [0.0 0.0 0.0 1.0]  
of.group.edge 'right' range plane.on [1.0 0.0 1.0 1.0]
```

2.5 Assign Boundary Conditions

1. For this example, 30 MPa compressive stress in the xx direction and 5 MPa compressive stresses in the yy direction are assigned to the rock.

```
of.boundary.element.stress [rock] xx -30e6 xy 0.0 yy -5e6
```

2. To quickly balance the model, absorbing boundary conditions are assigned to four boundaries in the model.

```
of.boundary.edge.viscous 'up' normal shear  
of.boundary.edge.viscous 'bottom' normal shear  
of.boundary.edge.viscous 'right' normal shear  
of.boundary.edge.viscous 'left' normal shear
```

2.6 Set the Outputs

1. Set the output interval to be every 2000 steps and output all fields variables and fracture variables. Control the output interval to a reasonable size could shorten the computation time but get a good understanding of the model.

```
of.history.pv.interval 2000  
of.history.pv.field default  
of.history.pv.fracture default
```

2. In this step, model will first achieve the equilibrium without excavation under in-situ stresses state. The result of velocity or kinematic ratio should be less than 1e-5.

```
of.step 50000
```

3. Insert CZM after achieve the equilibrium of in-situ stresses

```
of.mesh.insert 'default'  
of.mat.cohesive 'default' EM ten 1e6 coh 20e6 fric 0.8 GI 2 GII 40
```

4. One step excavation. Remove the elements in the “excavation” group.

```
of.boundary.excavation 'excavation'
```

5. Clear the absorbing boundaries

```
of.boundary.edge.clear 'up'
of.boundary.edge.clear 'bottom'
of.boundary.edge.clear 'right'
of.boundary.edge.clear 'left'
```

6. Add pressures at the external boundaries

```
of.boundary.edge.pressure 'right' normal 30e6
of.boundary.edge.pressure 'up' normal 5e6
of.boundary.edge.pressure 'bottom' normal 5e6
of.boundary.edge.pressure 'left' normal 30e6
```

7. The program will run 500000 steps in total. In other words, it will output 25 files for reference.

```
of.step 500000
```

8. Finalize the model and clear all the temporary memories.

```
of.finalize
```

9. Save the notepad and double click the .of file to run the program.

5.6.3 3.0 Run the Program

When you run the program, you can first check the mesh that was created by Gmsh as shown in Figure 2. If the mesh has a good quality, you can close the window to continue run the program.

Check the boundary conditions here. Absorbing boundary conditions are on for both normal and shear directions. In-situ stresses are applied to the boundaries.

5.6.4 4.0 Results

The stress and displacement results here are for reference.

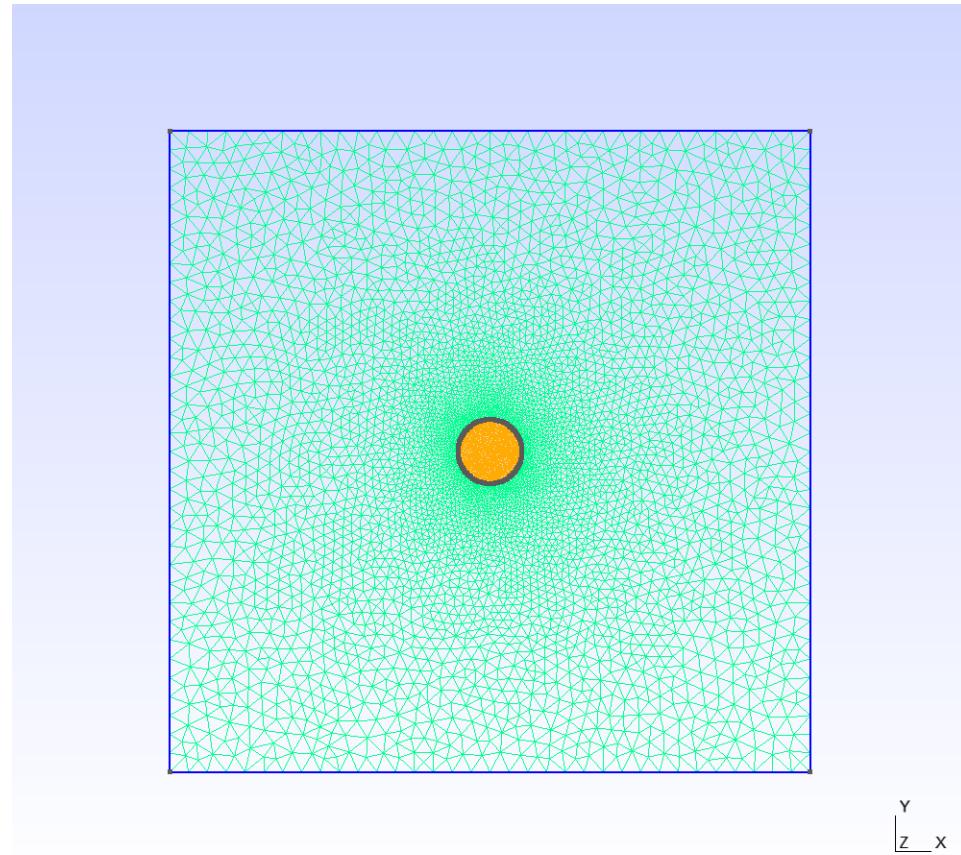


Fig. 14: Figure 2: Mesh created by Gmsh

| Checking model boundary information ... | | | | |
|---|------------------|---------------|------------|---|
| Bou ID | Category objects | Boundary type | Group Tags | Info and Extra |
| 0 | Edge boundary | Viscous | up | normal ON, shear ON |
| 1 | Edge boundary | Viscous | bottom | normal ON, shear ON |
| 2 | Edge boundary | Viscous | right | normal ON, shear ON |
| 3 | Edge boundary | Viscous | left | normal ON, shear ON |
| 4 | Element boundary | Stress | | xx = -30.00 MPa, xy = 0.00 MPa, yy = -5.00 MPa, |

Fig. 15: Figure 3: Check the mesh setting from command window

5.6.5 5.0 Full Script

- Apply_Insitu_Stress.of (click to download from Gitlab)

```
# initialization, this command is to clear the memory in your last run, it is
# not mandatory
# but strongly recommend.
of.new

# Set the path folder of your output results, //result// in the same path of
# your input file
# will be created by default
of.set.result "result"

# Set the number of cores you want to use, the parallelization will be
# automatically turned on
# when you use the following command, otherwise the serialization will be used
# by default
of.set.omp 15

#####
##### create mesh #####
# of.geometry.square 'rock' [0 1 0 1]
# of.geometry.cut.circle 'hole' 'rock' [0.5 0.5 0.05 100]

of.geometry.mesh.size 'default' 0.03
of.geometry.mesh auto

of.group.element 'excavation' range circle.in [0.5 0.5 0.05]

#####
##### assign material parameters #####
# Assign material for matrix.
of.mat.element 'default' elastic den 2700 E 30e9 v 0.3 damp 1.0
# assign materials for contacts
of.mat.contact 'default' MC fric 0.3

#####
##### create groups #####
# OpenFDEM can manually group the nodes, elements, cohesive elements and edges
# by using the
# region of box, circle and plane.
of.group.edge 'bottom' range plane.on [0.0 0.0 1.0 0.0]
of.group.edge 'up' range plane.on [0.0 1.0 1.0 1.0]
of.group.edge 'left' range plane.on [0.0 0.0 0.0 1.0]
of.group.edge 'right' range plane.on [1.0 0.0 1.0 1.0]

#####
##### assign boundaries #####
# Set the in-situ stresses at the boundaries
of.boundary.element.stress [rock] xx -30e6 xy 0.0 yy -5e6
```

(continues on next page)

(continued from previous page)

```

# add absorbing boundary to quickly balance the model
of.boundary.edge.viscous 'up' normal shear
of.boundary.edge.viscous 'bottom' normal shear
of.boundary.edge.viscous 'right' normal shear
of.boundary.edge.viscous 'left' normal shear

#####
# Set the interval of writing ParaView results.
of.history.pv.interval 2000
# Output all results by default.
of.history.pv.field default
of.history.pv.fracture default

#####
# Run the in-situ stress equilibrium step. Energy equilibrium is achieved when
# the maximum
# velocity is less than 1e-5 or the kinematic ratio is less than 1e-5 or the
# mechanical ratio
# is less than 1e-5
of.step 50000

# insert CZM after achieve the equilibrium of in-situ stresses
of.mesh.insert 'default'
of.mat.cohesive 'default' EM ten 1e6 coh 20e6 fric 0.8 GI 2 GII 40
of.boundary.excavation 'excavation'

# Clear the absorbing boundaries
of.boundary.edge.clear 'up'
of.boundary.edge.clear 'bottom'
of.boundary.edge.clear 'right'
of.boundary.edge.clear 'left'

# Add pressures at the external boundaries
of.boundary.edge.pressure 'right' normal 30e6
of.boundary.edge.pressure 'up' normal 5e6
of.boundary.edge.pressure 'bottom' normal 5e6
of.boundary.edge.pressure 'left' normal 30e6

# Run the excavation step
of.step 500000

# Finalize the model and clear the memory
of.finalize

```

5.7 Tutorial 6: Hydro Seepage

Tutorial 6 and 7 will present the numerical method to solve hydraulic fracture on a dam model in both finite element method (FEM) and finite-discrete element method (FDEM).

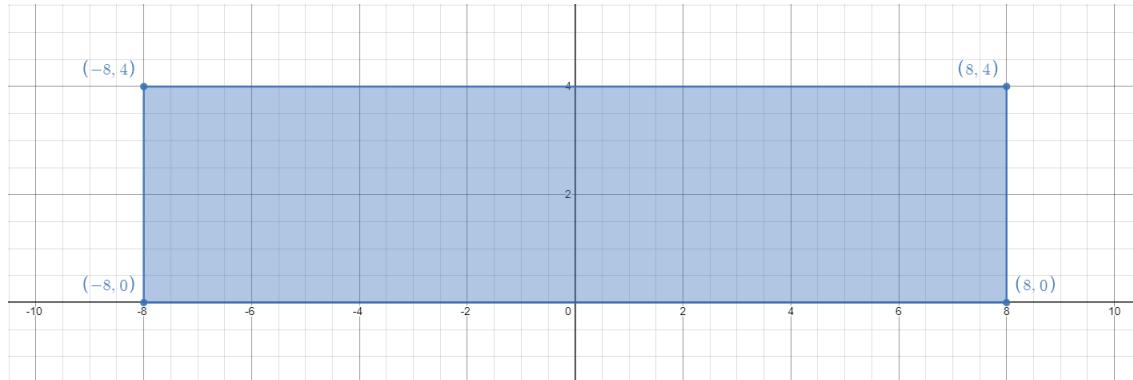


Fig. 16: Figure 1: Geometry of a Dam Model

5.7.1 1.0 Main Steps

1. Initialize the model.
2. Create the geometry and the mesh
3. Assign material properties.
4. Create the group
5. assign boundary conditions.
6. Set the outputs.

5.7.2 2.0 Codes

Please note that, all the setting in the example is based on the developer's computer. You may need to change these setting based on your own conditions.

To start with programming, create a new empty text file (later add the *.of* extension). Begin writing the following commands:

2.1 Initialize the model

1. Create a new run and clean up the old memories.

```
of.new
```

2. Set up the folder name to save your results. Here the folder name is set to be “result”.

```
of.set.result "result"
```

3. Set the number of cores to be used for running this model.

Warning: In this example, 15 cores will be used for running. To prevent the crushing of your computer, please check your computer to fill in the number of cores you want to use from your computer. To check the number of cores your computer has, you can go Task Manager > Performance > CPU > Cores. If nothing is set here, serial computing will be used.

```
of.set.omp 15
```

2.2 Create the geometry and the mesh

1. See Figure 1 for the dam model and draw the corresponding geometry.

```
of.geometry.square 'rock' xmin 0 xmax 8.0 ymin 0.0 ymax 4.0
```

2. Set the mesh size of the geometry.

```
of.geometry.mesh.size 'default' 0.1  
of.geometry.mesh delaunay
```

2.3 Assign Material Properties

1. Set the material properties. For this test, the example uses the elastic constitutive model and assigned material for CZM and contact.

```
of.mat.element 'default' elastic density 2700 E 5e8 v 0.2  
of.mat.contact 'default' MC fric 0.5
```

2. Set the hydraulic properties.

```
of.mat.hydro.matrix 'default' permeability 2e-8 Biot_K 22.0e9 Biot_c 0.1  
prosity 0.2
```

2.4 Create Groups and Assign Boundary Conditions

1. Group the nodes to prepare for the next step.

```
of.group.nodal 'left' range box.in xmin -100.0 xmax 0.1 ymin -100 ymax 100
of.group.nodal 'bottom' range box.in xmin -100 xmax 100 ymin -100 ymax 0.001
of.group.nodal 'up' range box.in xmin -100 xmax 100 ymin 3.9999 ymax 100
of.group.nodal 'right' range box.in xmin 7.999 xmax 100.0 ymin -100 ymax 100
```

2. Assign the nodal boundaries. Fix the bottom of the model.

```
of.boundary.nodal.velocity 'bottom' x 0 y 0
```

3. Assign the water level based on Figure 1. Water at the 0.0 m from the top of the dam on the left side of the dam and 3.0 m deep from the top of the dam on the right side of the dam.

```
of.boundary.hydro.waterlevel 'left' p0 0.0 head 4.0
of.boundary.hydro.waterlevel 'right' p0 0.0 head 1.0
```

2.5 Set the Outputs

1. Set the output interval to be every 6000 steps and output all fields variables and fracture variables. Control the output interval to a reasonable size could shorten the computation time but get a good understanding of the model.

```
of.history.pv.interval 6000
of.history.pv.field default
of.history.pv.fracture default
```

2. In this model, only hydro module is considered in the calculation. ????????

```
of.history.pv.interval 6000
of.history.pv.field default
of.history.pv.fracture default
```

3. Timestep for hydraulic calculation can be fixed to 1e-8 in this example.

```
of.hydro.timestep 1e-5
```

4. The program will run 60000 steps in total. In other words, it will output 10 files for reference.

```
of.step 60000
```

5. Finalize the model and clear all the temporary memories.

```
of.finalize
```

6. Save the notepad and double click the .of file to run the program.

5.7.3 3.0 Run the Program

Hydro module and matrix flow module are on to model the hydro seepage.

```
Checking global modules ...
+*****+
| Modules Turned ON |
+*****+
| Elasticity ON
| Hydro ON
| Fracture flow ON
| CZM ON
| Contact ON
| Release ON
| OpenMPI ON
+*****+
```

Fig. 17: Figure 2: Applied Modules

Hydro material information are included in the modeling.

| Checking model material information ... | | | | |
|---|-------------------------------|---------------------|---------|---|
| ID | Type | Model type | Tags | Properties |
| 0 | Matrix material | Linear elastic | default | E = 0.50 GPa v = 0.20 density = 2700.00 kg/m^3 damp = 1.00 |
| 1 | Cohesive material | Evans-Marathé FPZ | default | pn = 5.56 GPa pt = 4.17 GPa tension = 5.00 MPa cohesion = 10.00 MPa GI = 5.00 J/m^2 GII = 100.00 J/m^2 friction = 0.50 |
| 2 | Contact material (default) | Mohr-Coulomb law | default | kn = 5.56 GPa ks = 4.17 GPa friction = 0.50 |
| 3 | Hydro fracture material | Hrdro fracture flow | default | a0 = 0.10 mm a.min = 0.00 mm alpha = 1.00 power factor = 3.00 |
| 4 | Hydro fluid material | Hydro water | default | rou = 1000.00 kg/m^3 K.water = 20.00 Gpa viscosity = 0.00 Pa.s cohesion = 0.00 Mpa conductivity = 0.60 W/(m.C) specific heat = 4.19 J/(Kg.C) |

Fig. 18: Figure 3: Hydro Materials

Hydro boundary conditions.

| Checking model boundary information ... | | | | |
|---|------------------|---------------|------------|------------------------------|
| Bou ID | Category objects | Boundary type | Group Tags | Info and Extra |
| 0 | Node boundary | Velocity | bottom | x fixed y fixed |
| 1 | Hydro boundary | Water level | left | p0 = 0.00 MPa, head = 4.00 m |
| 2 | Hydro boundary | Water level | right | p0 = 0.00 MPa, head = 1.00 m |

Fig. 19: Figure 4: Hydro Boundary Conditions

5.7.4 4.0 Results

The stress and displacement results here are for reference.

5.7.5 5.0 Full Script

- Hydro_Seepage.of (click to download from Gitlab)

```
# initialization, this command is to clear the memory in your last run, it is
# not mandatory
# but strongly recommend.
of.new

# Set the path folder of your output results, //result// in the same path of
# your input file
# will be created by default
of.set.result "result"

# Set the number of cores you want to use, the parallelization will be
# automatically turned on
# when you use the following command, otherwise the serialization will be used
# by default
of.set.omp 15

##### create mesh #####
#of.geometry.square 'rock' xmin 0 xmax 8.0 ymin 0.0 ymax 4.0

of.geometry.mesh.size 'default' 0.1
of.geometry.mesh delaunay

##### assign material parameters #####
#of.mat.element 'default' elastic density 2700 E 5e8 v 0.2
#of.mat.contact 'default' MC fric 0.5

#set matrix parameter
#of.mat.hydro.matrix 'default' permiability 2e-8 Biot_K 22.0e9 Biot_c 0.1
#prosity 0.2

##### create groups #####
# (continues on next page)
```

(continued from previous page)

```

→#####
# OpenFDEM can manually group the nodes, elements, cohesive elements and edges
→by using the
# region of box, circle and plane.
of.group.nodal 'left' range box.in xmin -100.0 xmax 0.1 ymin -100 ymax 100
of.group.nodal 'bottom' range box.in xmin -100 xmax 100 ymin -100 ymax 0.001
of.group.nodal 'up' range box.in xmin -100 xmax 100 ymin 3.9999 ymax 100
of.group.nodal 'right' range box.in xmin 7.999 xmax 100.0 ymin -100 ymax 100

##### assign boundaries #####
→#####
# boundaries can be assigned after you define the groups
of.boundary.nodal.velocity 'bottom' x 0 y 0
of.boundary.hydro.waterlevel 'left' p0 0.0 head 4.0
of.boundary.hydro.waterlevel 'right' p0 0.0 head 1.0

##### set output #####
→#####
# Set the interval of writing ParaView results.
of.history.pv.interval 6000
of.history.pv.field default
of.history.pv.fracture default

##### execute model #####
→#####
# gravity for water level
of.set.gravity x 0 y -10
# only consider the hydro module
of.hydro.mechanical off
# set the timestep for hydro module
of.hydro.timestep 1e-5
# to excuate the model
of.step 60000
# finalize the model and clear the memory
of.finalize

```

5.8 Tutorial 7: Hydro Fracture Flow

This tutorial will continue the hydro seepage model in tutorial 6 with discrete elements.

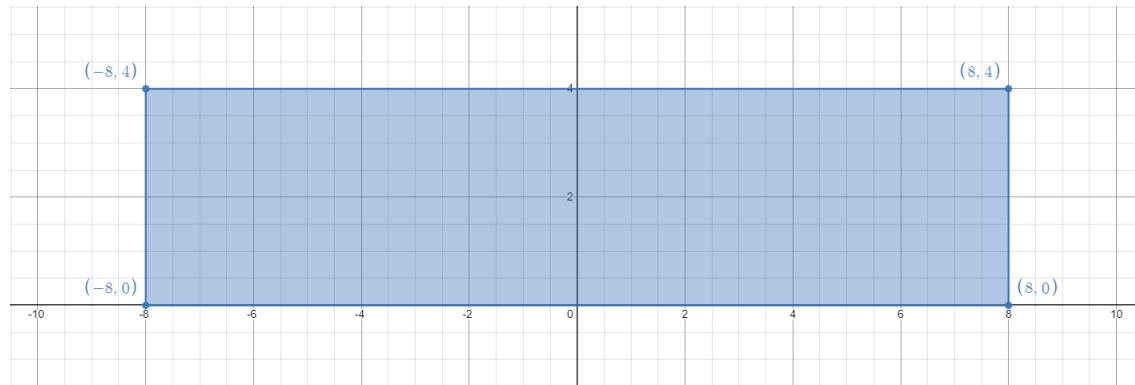


Fig. 20: Figure 1: Geometry of the uniaxial compression strength test

5.8.1 1.0 Code

Based on the code in tutorial 6, fracture flow parameters will be added to model the hydro fracture.

1. To include the discrete elements, joint elements are added to the model after creating the mesh.

```
of.mesh.insert 'default'
```

2. Assign the material properties to CZM.

```
of.mat.contact 'default' MC fric 0.5
```

3. Set fracture flow parameters

```
of.mat.hydro.fluid density 1000.0 K 20.0e9 viscosity 1e-3
of.mat.hydro.fracture 'default' a0 1e-4 amin 1e-4 amax 3e-4 power 3.0 b 1.0
```

5.8.2 2.0 Run the Program

Hydro module, Fracture flow module and CZM module are on to model the hydro fracture flow.

```
Checking global modules ...
+*****+
| Modules Turned ON |
+*****+
| Elasticity ON
| Hydro ON
| Fracture flow ON
| CZM ON
| Contact ON
| Release ON
| OpenMPI ON |
+*****+
```

Fig. 21: Figure 2: Applied Modules

Contact material, hydro fracture material and hydro fluid material are included in the modeling.

| Checking model material information ... | | | | |
|---|----------------------------|---------------------|---------|---|
| ID | Type | Model type | Tags | Properties |
| 0 | Matrix material | Linear elastic | default | E = 0.50 GPa v = 0.20 density = 2700.00 kg/m^3 damp = 1.00 |
| 1 | Cohesive material | Evans-Marathe FPZ | default | pn = 5.56 GPa pt = 4.17 GPa tension = 5.00 MPa cohesion = 10.00 MPa GI = 5.00 J/m^2 GII = 100.00 J/m^2 friction = 0.50 |
| 2 | Contact material (default) | Mohr-Coulomb law | default | kn = 5.56 GPa, ks = 4.17 GPa friction = 0.50 |
| 3 | Hydro fracture material | Hydro fracture flow | default | a0 = 0.10 mm a.min = 0.00 mm alpha = 1.00 power factor = 3.00 |
| 4 | Hydro fluid material | Hydro water | default | rho = 1000.00 kg/m^3 K.water = 20.00 Gpa viscosity = 0.00 Pa.s cohesion = 0.00 Mpa conductivity = 0.60 W/(m.C) specific heat = 4.19 J/(Kg.C) |

Fig. 22: Figure 3: Hydro Fracture Materials

Hydro boundary conditions.

| Checking model boundary information ... | | | | |
|---|------------------|---------------|------------|------------------------------|
| Bou ID | Category objects | Boundary type | Group Tags | Info and Extra |
| 0 | Node boundary | Velocity | bottom | x fixed y fixed |
| 1 | Hydro boundary | Water level | left | p0 = 0.00 MPa, head = 4.00 m |
| 2 | Hydro boundary | Water level | right | p0 = 0.00 MPa, head = 1.00 m |

Fig. 23: Figure 4: Hydro Boundary Conditions

5.8.3 3.0 Results

The stress and displacement results here are for reference.

5.8.4 4.0 Full Script

- Hydro_Fracture_Flow.of (click to download from GitHub gist)

```
# initialization, this command is to clear the memory in your last run, it is
# not mandatory
# but strongly recommend.
of.new

# Set the path folder of your output results, //result// in the same path of
# your input file
# will be created by default
of.set.result "result"

# Set the number of cores you want to use, the parallelization will be
# automatically turned on
# when you use the following command, otherwise the serialization will be used
# by default
of.set.omp 15

##### create mesh #####
#of.geometry.square 'rock' xmin 0 xmax 8.0 ymin 0.0 ymax 4.0

of.geometry.mesh.size 'default' 0.1
of.geometry.mesh delaunay

of.mesh.insert 'default'

##### assign material parameters #####
#of.mat.element 'default' elastic density 2700 E 5e8 v 0.2
#of.mat.cohesive 'default' EM tension 5e6 cohesion 10e6 fric 0.5 GI 5 GII 100
#of.mat.contact 'default' MC fric 0.5

#set fracture flow parameter
of.mat.hydro.fluid density 1000.0 K 20.0e9 viscosity 1e-3
of.mat.hydro.fracture 'default' a0 1e-4 amin 1e-4 amax 3e-4 power 3.0 b 1.0

##### create groups #####
# OpenFDEM can manually group the nodes, elements, cohesive elements and edges
# by using the
# region of box, circle and plane.
of.group.nodal 'left' range box.in xmin -100.0 xmax 0.1 ymin -100 ymax 100
of.group.nodal 'bottom' range box.in xmin -100 xmax 100 ymin -100 ymax 0.001
of.group.nodal 'up' range box.in xmin -100 xmax 100 ymin 3.9999 ymax 100
of.group.nodal 'right' range box.in xmin 7.999 xmax 100.0 ymin -100 ymax 100
```

(continues on next page)

(continued from previous page)

```

#####
##### assign boundaries #####
# boundaries can be assigned after you define the groups
of.boundary.nodal.velocity 'bottom' x 0 y 0
of.boundary.hydro.waterlevel 'left' p0 0.0 head 4.0
of.boundary.hydro.waterlevel 'right' p0 0.0 head 1.0

#####
##### set output #####
# Set the interval of writing ParaView results.
of.history.pv.interval 6000
of.history.pv.field default
of.history.pv.fracture default

#####
##### execute model #####
# gravity for water level
of.set.gravity x 0 y -10
# only consider the hydro module
of.hydro.mechanical off
# set the timestep for hydro module
of.hydro.timestep 1e-5
# to excuate the model
of.step 60000
# finalize the model and clear the memory
of.finalize

```

5.9 Tutorial 8: Thermal Flux

Thermal module will be reviewed in this tutorial.

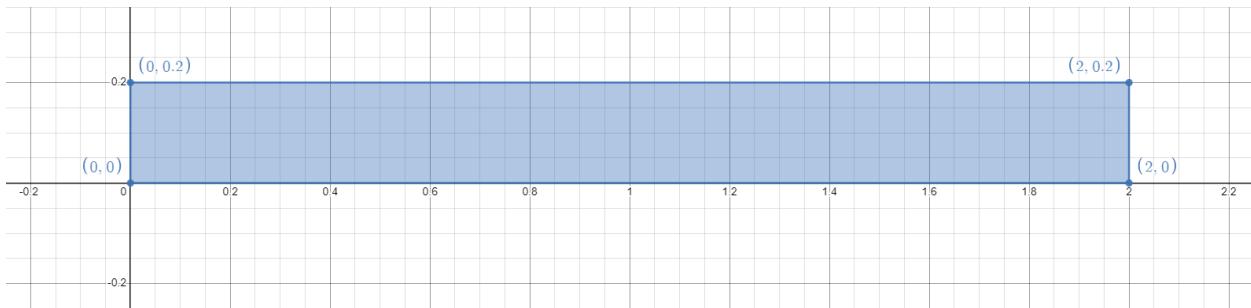


Fig. 24: Figure 1: Geometry of the uniaxial compression strength test

5.9.1 1.0 Main Steps

1. Initialize the model.
2. Create the geometry and the mesh
3. Assign material properties.
4. Create the group and assign boundary conditions.
5. Set the outputs.

5.9.2 2.0 Codes

Please note that, all the setting in the example is based on the developer's computer. You may need to change these setting based on your own conditions.

To start with programming, create a new empty text file (later add the *.of* extension). Begin writing the following commands:

2.1 Initialize the model

1. Create a new run and clean up the old memories.

```
of.new
```

2. Set up the folder name to save your results. Here the folder name is set to be “result”.

```
of.set.result "result"
```

3. Set the number of cores to be used for running this model.

Warning: In this example, 15 cores will be used for running. To prevent the crushing of your computer, please check your computer to fill in the number of cores you want to use from your computer. To check the number of cores your computer has, you can go Task Manager > Performance > CPU > Cores. If nothing is set here, serial computing will be used.

```
of.set.omp 15
```

2.2 Create the geometry and the mesh

1. See Figure 1 for the thermal conduction setup and draw the corresponding geometry.

```
of.geometry.square 'barleft' xmin 0 xmax 1.0 ymin 0 ymax 0.2
of.geometry.square 'barright' xmin 1.0 xmax 2.0 ymin 0 ymax 0.2
```

2. Set the mesh size of the geometry.

```
of.geometry.mesh.size 'default' 0.05
```

3. Set the quadrangle elements

```
of.geometry.recombine 'barright'
```

4. Create the mesh

```
of.geometry.mesh delaunay
```

2.3 Assign Material and Thermal Properties

1. Set the material properties.

```
of.mat.element 'default' elastic density 2700 E 5e8 v 0.2
of.mat.cohesive 'default' EM ten 1e6 coh 3e6 fric 0.3 GI 10 GII 50 heat_exchage_
→20
of.mat.contact 'default' MC fric 0.3 conductivity 200
```

2. Set the thermal properties.

```
of.mat.thermal 'default' conductivity 160 spec-heat 0.2 expansion 1e-6
```

2.4 Create Groups and Assign Boundary Conditions

1. Group the nodes to prepare for the next step.

```
of.group.nodal 'left' range plane.on x0 0.0 y0 0.0 x1 0.0 y1 0.2
of.group.nodal 'right' range plane.on x0 2.0 y0 0.0 x1 2.0 y1 0.2
of.group.nodal 'up' range plane.on x0 0.0 y0 0.2 x1 2.0 y1 0.2
of.group.nodal 'down' range plane.on x0 0.0 y0 0.0 x1 2.0 y1 0.0
```

2. Assign the thermal properties to the model. In this model, the temperature of the left of the example is 50 degree Celsius. The temperature of the right of the example is 0 degree Celsius. Heat will be conducted in this model from left to right.

```
of.group.nodal 'left' range plane.on x0 0.0 y0 0.0 x1 0.0 y1 0.2
of.group.nodal 'right' range plane.on x0 2.0 y0 0.0 x1 2.0 y1 0.2
of.group.nodal 'up' range plane.on x0 0.0 y0 0.2 x1 2.0 y1 0.2
of.group.nodal 'down' range plane.on x0 0.0 y0 0.0 x1 2.0 y1 0.0
```

3. Assign the nodal boundaries. In this model, top, bottom and right boundaries are fixed in both x and y directions.

```
of.boundary.nodal.velocity 'right' x 0.0 y 0.0
of.boundary.nodal.velocity 'up' x 0.0 y 0.0
of.boundary.nodal.velocity 'down' x 0.0 y 0.0
```

2.5 Set the Outputs

1. Set the output interval to be every 500 steps and output all fields variables and fracture variables. Control the output interval to a reasonable size could shorten the computation time but get a good understanding of the model.

```
of.history.pv.interval 500
of.history.pv.field default
of.history.pv.fracture default
```

2. Timestep of thermal calculation can be fixed to 5e-5 in this example.

```
of.thermal.timestep 5e-5
```

3. The program will run 15000 steps in total. In other words, it will output 10 files for reference.

```
of.step 15000
```

4. Finalize the model and clear all the temporary memories.

```
of.finalize
```

5. Save the notepad and double click the .of file to run the program.

5.9.3 3.0 Run the Program

When you run the program, you can first check the mesh that was created by Gmsh as shown in Figure 2. If the mesh has a good quality, you can close the window to continue run the program.

The mesh is using triangle elements.

Material properties you set on step 2.3 will be shown on the screen. You can confirm it while the program just starts to run.

Furthermore, node boundaries are shown at the header of the program.

Global timestep is shown here.

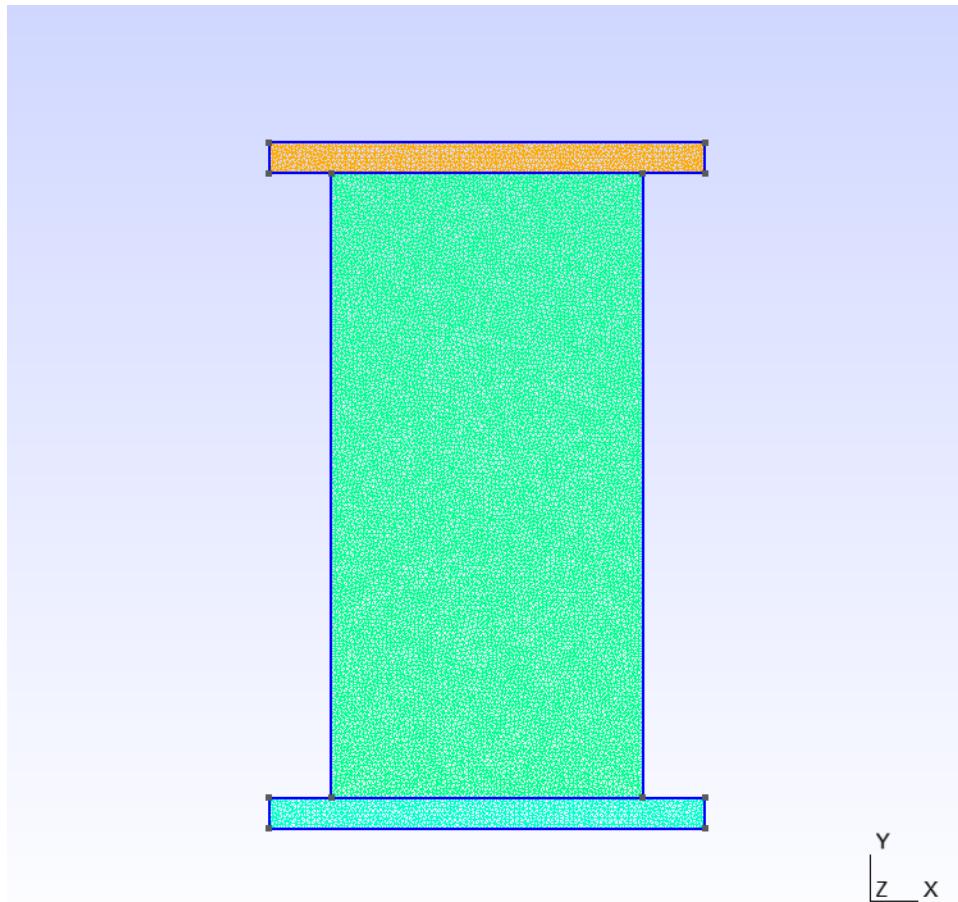


Fig. 25: Figure 2: Mesh created by Gmsh

| group ID | Type | Number of Object | Tags | Extra |
|-----------------|---------|------------------|------|--------------|
| 0 | Node | 782 | up | |
| 1 | Node | 773 | down | |
| 2 | Element | 20622 | rock | Triangle, T3 |
| 3 | Element | 1372 | up | Triangle, T3 |
| 4 | Element | 1354 | down | Triangle, T3 |
| Total nodes | | 63421 | | |
| Total elements | | 23348 | | |
| Total cohesives | | 31121 | | |
| Total entities | | 20624 | | |

Fig. 26: Figure 3: Check the mesh setting from command window

| Checking model material information ... | | | | |
|---|----------------------------|-------------------|---------|--|
| ID | Type | Model type | Tags | Properties |
| 0 | Matrix material | Linear elastic | default | E = 30.00 GPa v = 0.30 density = 2700.00 kg/m ³ damp = 1.00 |
| 1 | Cohesive material | Evans-Marathé FPZ | default | p _n = 500.00 GPa p _t = 230.77 GPa tension = 1.00 MPa cohesion = 3.00 MPa GI = 10.00 J/m ² GII = 50.00 J/m ² friction = 0.30 |
| 2 | Contact material (default) | Mohr-Coulomb law | default | k _n = 500.00 GPa k _s = 230.77 GPa friction = 0.30 |

Fig. 27: Figure 4: Check the material assignment from command window

| Checking model boundary information ... | | | | |
|---|------------------|---------------|------------|-----------------------|
| Bou ID | Category objects | Boundary type | Group Tags | Info and Extra |
| 0 | Node boundary | Velocity | up | x fixed y = -0.05 m/s |
| 1 | Node boundary | Velocity | down | x fixed y = 0.05 m/s |

Fig. 28: Figure 5: Check the boundary information from command window

Start to run the program. Results are logged for every 2000 steps.

5.9.4 4.0 Results

The stress and displacement results here are for reference.

5.9.5 5.0 Full Script

- Thermal_Flux.of (click to download from Gitlab)

```
# initialization, this command is to clear the memory in your last run, it is
# not mandatory
# but strongly recommend.
of.new

# Set the path folder of your output results, //result// in the same path of
# your input file
```

(continues on next page)

```
Info    :critical timestep from MATRIX is: [1.23e-07].
Info    :critical timestep from INTERFACE is: [1.83e-10].
Info    :critical timestep from CONTACT is: [1.41e-08].
Info    :Warning: the user-defined timestep is greater than the critical timestep required.
Info    :global timestep is: [5.00e-09].
```

Fig. 29: Figure 6: Check the timesteps from command window

| Start running ... | | | | | | |
|-------------------|------------------|--------------------|---------------|------------|---------------|-------------|
| step | physical time(s) | unbalance force(N) | kinetic ratio | broken CZM | contact pairs | run time(s) |
| 0 | 0.000e+00 | 0.000e+00 | 0.000e+00 | 0 | 621 | 0.000 |
| 2000 | 1.000e-05 | 2.846e+02 | 2.426e-04 | 0 | 626 | 29.437 |
| 4000 | 2.000e-05 | 3.472e+02 | 4.435e-04 | 0 | 630 | 60.029 |

Fig. 30: Figure 7: Start to run the program

(continued from previous page)

```

# will be created by default
of.set.result "result"

# Set the number of cores you want to use, the parallelization will be
# automatically turned on
# when you use the following command, otherwise the serialization will be used
# by default
of.set.omp 15

#####
# create mesh #####
#
of.geometry.square 'barleft' xmin 0 xmax 1.0 ymin 0 ymax 0.2
of.geometry.square 'barright' xmin 1.0 xmax 2.0 ymin 0 ymax 0.2
of.geometry.mesh.size 'default' 0.05
of.geometry.recombine 'barright'
of.geometry.mesh delaunay

#####
# assign material parameters #####
#
# assign material for matrix
of.mat.element 'default' elastic density 2700 E 5e8 v 0.2
# heat exchange in cohesive elements
of.mat.cohesive 'default' EM ten 1e6 coh 3e6 fric 0.3 GI 10 GII 50 heat_
# exchange 20
# conductivity of thermal in contacts
of.mat.contact 'default' MC fric 0.3 conductivity 200

#set thermal matrix parameter
of.mat.thermal 'default' conductivity 160 spec-heat 0.2 expansion 1e-6

#####
# create groups #####
#
of.group.nodal 'left' range plane.on x0 0.0 y0 0.0 x1 0.0 y1 0.2
of.group.nodal 'right' range plane.on x0 2.0 y0 0.0 x1 2.0 y1 0.2
of.group.nodal 'up' range plane.on x0 0.0 y0 0.2 x1 2.0 y1 0.2
of.group.nodal 'down' range plane.on x0 0.0 y0 0.0 x1 2.0 y1 0.0

#####
# assign boundaries #####
#
# boundaries can be assigned after you define the groups

```

(continues on next page)

(continued from previous page)

```

of.boundary.thermal.temperature 'left' T 50
of.boundary.thermal.temperature 'right' T 0.0
of.boundary.nodal.velocity 'right' x 0.0 y 0.0
of.boundary.nodal.velocity 'up' x 0.0 y 0.0
of.boundary.nodal.velocity 'down' x 0.0 y 0.0

#####
# Set the interval of writing ParaView results.
of.history.pv.interval 500
of.history.pv.field default
of.history.pv.fracture default

#####
# set the timestep for thermal module
of.thermal.timestep 5e-5
# to excuate the model
of.step 15000
# finalize the model and clear the memory
of.finalize

```

5.10 Tutorial 9: Grain-based Model (GBM)

Grain based modeling on UCS test.

5.10.1 1.0 Tutorial Prerequisites

The following files are needed to follow along the tutorial:

- [Job-1.inp](#) (click to download from GitHub gist)

5.10.2 2.0 Main Steps

1. Initialize the model.
2. Create the geometry and the mesh
3. Assign material properties.
4. Create the group and assign boundary conditions.
5. Set the outputs.

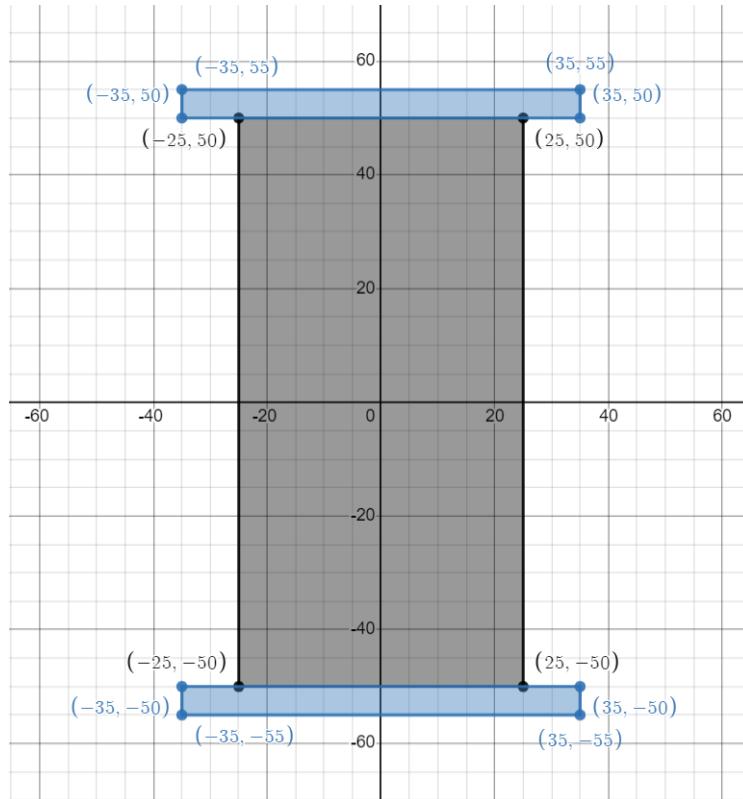


Fig. 31: Figure 1: Geometry of the uniaxial compression strength test

5.10.3 3.0 Codes

Please note that, all the setting in the example is based on the developer's computer. You may need to change these setting based on your own conditions.

To start with programming, create a new empty text file (later add the *.of* extension). Begin writing the following commands:

3.1 Initialize the model

1. Create a new run and clean up the old memories.

```
of.new
```

2. Turn on GBM module

```
of.config.GBM
```

3. Set up the folder name to save your results. Here the folder name is set to be “result”.

```
of.set.result "result"
```

4. Set the number of cores to be used for running this model.

Warning: In this example, 15 cores will be used for running. To prevent the crushing of your computer, please check your computer to fill in the number of cores you want to use from your computer. To check the number of cores your computer has, you can go Task Manager > Performance > CPU > Cores. If nothing is set here, serial computing will be used.

```
of.set.omp 15
```

3.2 Import the geometry and the mesh

1. Import the geometry

```
of.import 'Job-1.inp'
```

2. of.set.gbm 3 'Qtz' 0.3 'Fel' 0.46 'Bio' 0.24

3. Group the elements.

```
of.group.element 'rock' range box.in xmin -0.05 xmax 0.05 ymin -0.05 ymax 0.05
of.group.element 'up' range box.in xmin -1.0 xmax 1.0 ymin 0.05 ymax 1.0
of.group.element 'down' range box.in xmin -1.0 xmax 1.0 ymin -1.0 ymax -0.05
```

4. Insert cohesive elements

```
of.mesh.insert 'rock'
```

5. Group cohesive elements

```
of.group.cohelement.from.gbm 'Qtz-Qtz' 'Qtz' 'Qtz' # Quartz group
of.group.cohelement.from.gbm 'Qtz-Fel' 'Qtz' 'Fel' # Quartz-feldspar
→group
of.group.cohelement.from.gbm 'Fel-Fel' 'Fel' 'Fel' # feldspar-feldspar
→group
of.group.cohelement.from.gbm 'Fel-Bio' 'Fel' 'Bio' # feldspar-biotite
→group
of.group.cohelement.from.gbm 'Bio-Bio' 'Bio' 'Bio' # biotite-biotite
→group
of.group.cohelement.from.gbm 'Bio-Qtz' 'Bio' 'Qtz' # biotite-Quartz group
```

3.3 Assign Material and Thermal Properties

1. Set the material properties of minerals and platens.

```
of.mat.element 'Qtz' elastic den 2700 E 70e9 v 0.1 damp 0.6
of.mat.element 'Fel' elastic den 3500 E 45e9 v 0.2 damp 0.6
of.mat.element 'Bio' elastic den 1800 E 20e9 v 0.3 damp 0.6
of.mat.element 'up' elastic den 7000 E 120e9 v 0.3 damp 0.6
of.mat.element 'down' elastic den 7000 E 120e9 v 0.3 damp 0.6
```

2. Assign materials for CZM.

```
of.mat.cohesive 'Qtz-Qtz' EM ten 15e6 coh 30e6 fric 0.1 GI 5e2 GII 50e3
of.mat.cohesive 'Qtz-Fel' EM ten 5e6 coh 10e6 fric 0.3 GI 1e2 GII 50e3
of.mat.cohesive 'Fel-Fel' EM ten 10e6 coh 20e6 fric 0.2 GI 2e2 GII 50e3
of.mat.cohesive 'Fel-Bio' EM ten 5e6 coh 10e6 fric 0.3 GI 1e2 GII 50e3
of.mat.cohesive 'Bio-Bio' EM ten 8e6 coh 15e6 fric 0.2 GI 1.5e2 GII 50e3
of.mat.cohesive 'Bio-Qtz' EM ten 5e6 coh 10e6 fric 0.3 GI 1e2 GII 50e3
```

3. Assign materials for contact.

```
of.mat.contact 'default' MC fric 0.3
of.mat.contact 'Qtz' 'Qtz' MC fric 0.1
of.mat.contact 'Qtz' 'Fel' MC fric 0.3
of.mat.contact 'Fel' 'Fel' MC fric 0.2
of.mat.contact 'Fel' 'Bio' MC fric 0.3
of.mat.contact 'Bio' 'Bio' MC fric 0.2
of.mat.contact 'Bio' 'Qtz' MC fric 0.3
```

3.4 Create Groups and Assign Boundary Conditions

1. Group the nodals to prepare for the next step.

```
of.group.nodal.from.element 'up' 'up'
of.group.nodal.from.element 'down' 'down'
```

2. Assign the nodal boundaries. In this model, top, bottom and right boundaries are fixed in both x and y directions.

```
of.boundary.nodal.velocity 'up' x 0.0 y -0.05
of.boundary.nodal.velocity 'down' x 0.0 y 0.05
```

3.5 Set the Outputs

1. Set the output interval to be every 500 steps and output all fields variables and fracture variables. Control the output interval to a reasonable size could shorten the computation time but get a good understanding of the model.

```
of.history.pv.interval 50000
of.history.pv.field default
of.history.pv.fracture default
```

2. The program will run 10000000 steps in total. In other words, it will output 10 files for reference.

```
of.step 10000000
```

3. Finalize the model and clear all the temporary memories.

```
of.finalize
```

4. Save the notepad and double click the .of file to run the program.

5.10.4 4.0 Run the Program

When you run the program, you can first check the mesh that was created by Gmsh as shown in Figure 2. If the mesh has a good quality, you can close the window to continue run the program.

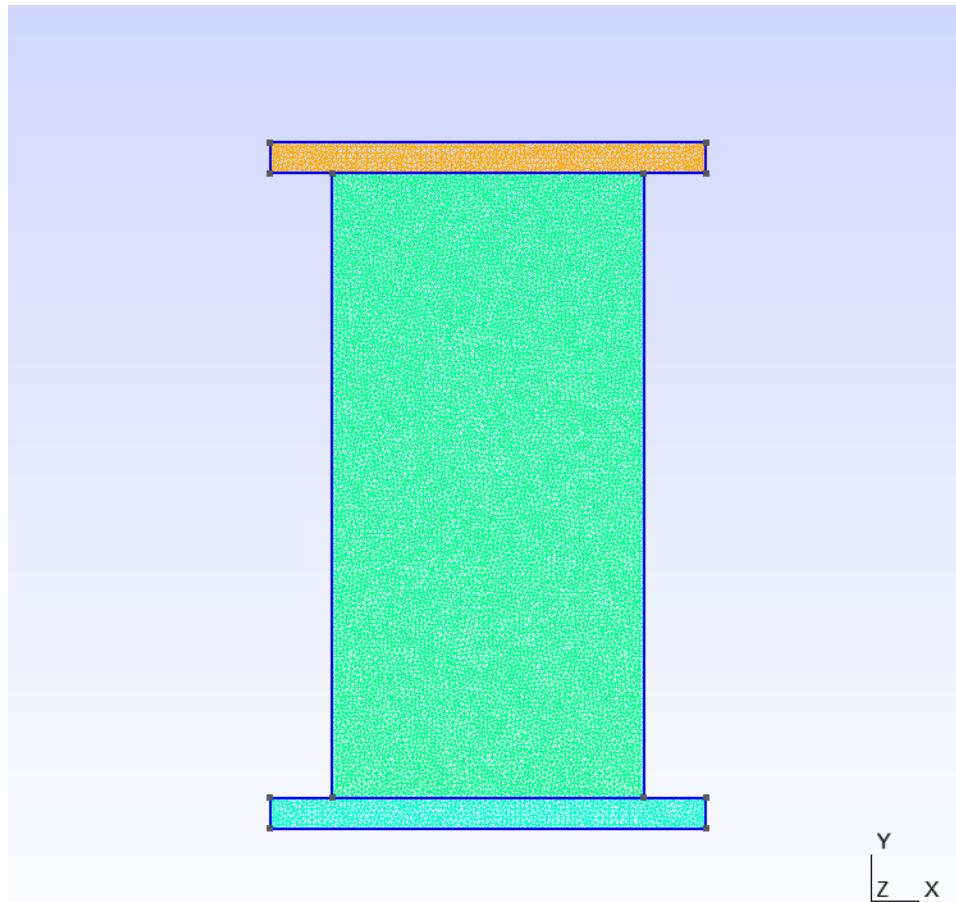


Fig. 32: Figure 2: Mesh created by Gmsh

The mesh is using triangle elements.

Material properties you set on step 2.3 will be shown on the screen. You can confirm it while the program just starts to run.

| group ID | Type | Number of Object | Tags | Extra |
|-----------------|---------|------------------|------|--------------|
| 0 | Node | 782 | up | |
| 1 | Node | 773 | down | |
| 2 | Element | 20622 | rock | Triangle, T3 |
| 3 | Element | 1372 | up | Triangle, T3 |
| 4 | Element | 1354 | down | Triangle, T3 |
| Total nodes | | 63421 | | |
| Total elements | | 23348 | | |
| Total cohesives | | 31121 | | |
| Total entities | | 20624 | | |

Fig. 33: Figure 3: Check the mesh setting from command window

| Checking model material information ... | | | | |
|---|----------------------------|-------------------|---------|--|
| ID | Type | Model type | Tags | Properties |
| 0 | Matrix material | Linear elastic | default | E = 30.00 GPa v = 0.30 density = 2700.00 kg/m^3 damp = 1.00 |
| 1 | Cohesive material | Evans-Marathé FPZ | default | pn = 500.00 GPa pt = 230.77 GPa tension = 1.00 MPa cohesion = 3.00 MPa GI = 10.00 J/m^2 GII = 50.00 J/m^2 friction = 0.30 |
| 2 | Contact material (default) | Mohr-Coulomb law | default | kn = 500.00 GPa ks = 230.77 GPa friction = 0.30 |

Fig. 34: Figure 4: Check the material assignment from command window

Furthermore, node boundaries are shown at the header of the program.

| Checking model boundary information ... | | | | |
|---|------------------|---------------|------------|-----------------------|
| Bou ID | Category objects | Boundary type | Group Tags | Info and Extra |
| 0 | Node boundary | Velocity | up | x fixed y = -0.05 m/s |
| 1 | Node boundary | Velocity | down | x fixed y = 0.05 m/s |

Fig. 35: Figure 5: Check the boundary information from command window

Global timestep is shown here.

Start to run the program. Results are logged for every 2000 steps.

```

Info    :critical timestep from MATRIX is: [1.23e-07].
Info    :critical timestep from INTERFACE is: [1.83e-10].
Info    :critical timestep from CONTACT is: [1.4le-08].
Info    :Warning: the user-defined timestep is greater than the critical timestep required.
Info    :global timestep is: [5.00e-09].

```

Fig. 36: Figure 6: Check the timesteps from command window

| Start running ... | | | | | | |
|-------------------|------------------|--------------------|---------------|------------|---------------|-------------|
| step | physical time(s) | unbalance force(N) | kinetic ratio | broken CZM | contact pairs | run time(s) |
| 0 | 0.000e+00 | 0.000e+00 | 0.000e+00 | 0 | 621 | 0.000 |
| 2000 | 1.000e-05 | 2.846e+02 | 2.426e-04 | 0 | 626 | 29.437 |
| 4000 | 2.000e-05 | 3.472e+02 | 4.435e-04 | 0 | 630 | 60.029 |

Fig. 37: Figure 7: Start to run the program

5.10.5 5.0 Results

The stress and displacement results here are for reference.

5.10.6 6.0 Full Script

- GBM.of (click to download from GitHub gist)

```

# initialization, this command is to clear the memory in your last run, it is
# not mandatory
# but strongly recommend.
of.new
of.config.GBM

# Set the path folder of your output results, //result// in the same path of
# your input file
# will be created by default
of.set.result "result"

# Set the number of cores you want to use, the parallelization will be
# automatically turned on
# when you use the following command, otherwise the serialization will be used
# by default
of.set.omp 15

#####
# import mesh #####
of.import 'Job-1.inp'

of.set.gbm 3 'Qtz' 0.3 'Fel' 0.46 'Bio' 0.24
of.group.element 'rock' range box.in xmin -0.05 xmax 0.05 ymin -0.05 ymax 0.05
of.group.element 'up'   range box.in xmin -1.0 xmax 1.0 ymin 0.05 ymax 1.0
of.group.element 'down' range box.in xmin -1.0 xmax 1.0 ymin -1.0 ymax -0.05

```

(continues on next page)

(continued from previous page)

```

# insert cohesive elements
of.mesh.insert 'rock'

# Group Cohesive Elements
of.group.cohelement.from.gbm 'Qtz-Qtz' 'Qtz' 'Qtz'      # Quartz group
of.group.cohelement.from.gbm 'Qtz-Fel' 'Qtz' 'Fel'        # Quartz-feldspar
  ↳group
of.group.cohelement.from.gbm 'Fel-Fel' 'Fel' 'Fel'        # feldspar-feldspar
  ↳group
of.group.cohelement.from.gbm 'Fel-Bio' 'Fel' 'Bio'        # feldspar-biotite
  ↳group
of.group.cohelement.from.gbm 'Bio-Bio' 'Bio' 'Bio'        # biotite-biotite
  ↳group
of.group.cohelement.from.gbm 'Bio-Qtz' 'Bio' 'Qtz'        # biotite-Quartz
  ↳group

##### assign material parameters #####
→#####
of.mat.element 'Qtz' elastic den 2700 E 70e9 v 0.1 damp 0.6
of.mat.element 'Fel' elastic den 3500 E 45e9 v 0.2 damp 0.6
of.mat.element 'Bio' elastic den 1800 E 20e9 v 0.3 damp 0.6

of.mat.element 'up' elastic den 7000 E 120e9 v 0.3 damp 0.6
of.mat.element 'down' elastic den 7000 E 120e9 v 0.3 damp 0.6

# assign materials for CZM
of.mat.cohesive 'Qtz-Qtz' EM ten 15e6 coh 30e6 fric 0.1 GI 5e2 GII 50e3
of.mat.cohesive 'Qtz-Fel' EM ten 5e6 coh 10e6 fric 0.3 GI 1e2 GII 50e3
of.mat.cohesive 'Fel-Fel' EM ten 10e6 coh 20e6 fric 0.2 GI 2e2 GII 50e3
of.mat.cohesive 'Fel-Bio' EM ten 5e6 coh 10e6 fric 0.3 GI 1e2 GII 50e3
of.mat.cohesive 'Bio-Bio' EM ten 8e6 coh 15e6 fric 0.2 GI 1.5e2 GII 50e3
of.mat.cohesive 'Bio-Qtz' EM ten 5e6 coh 10e6 fric 0.3 GI 1e2 GII 50e3

# assign materials for contacts
of.mat.contact 'default' MC fric 0.3
of.mat.contact 'Qtz' 'Qtz' MC fric 0.1
of.mat.contact 'Qtz' 'Fel' MC fric 0.3
of.mat.contact 'Fel' 'Fel' MC fric 0.2
of.mat.contact 'Fel' 'Bio' MC fric 0.3
of.mat.contact 'Bio' 'Bio' MC fric 0.2
of.mat.contact 'Bio' 'Qtz' MC fric 0.3

##### create groups #####
→#####
of.group.nodal.from.element 'up' 'up'
of.group.nodal.from.element 'down' 'down'

##### assign boundaries #####
→#####
# boundaries can be assigned after you define the groups
of.boundary.nodal.velocity 'up' x 0.0 y -0.05
of.boundary.nodal.velocity 'down' x 0.0 y 0.05

```

(continues on next page)

(continued from previous page)

```
#####
# set output #####
# Set the interval of writing ParaView results.
of.history.pv.interval 50000
of.history.pv.field default
of.history.pv.fracture default

#####
# execute model #####
# to excute the model
of.step 1000000
# finalize the model and clear the memory
of.finalize
```

5.11 Tutorial 10: Phase Field

Uniaxial compression test is one of the most fundamental tests that will be used to get the material properties from the experimental stress-strain curve.

5.11.1 1.0 Main Steps

1. Initialize the model.
2. Create the geometry and the mesh
3. Assign material properties.
4. Create the group and assign boundary conditions.
5. Set the outputs.

5.11.2 2.0 Codes

Please note that, all the setting in the example is based on the developer's computer. You may need to change these setting based on your own conditions.

To start with programming, create a new empty text file (later add the `.of` extension). Begin writing the following commands:

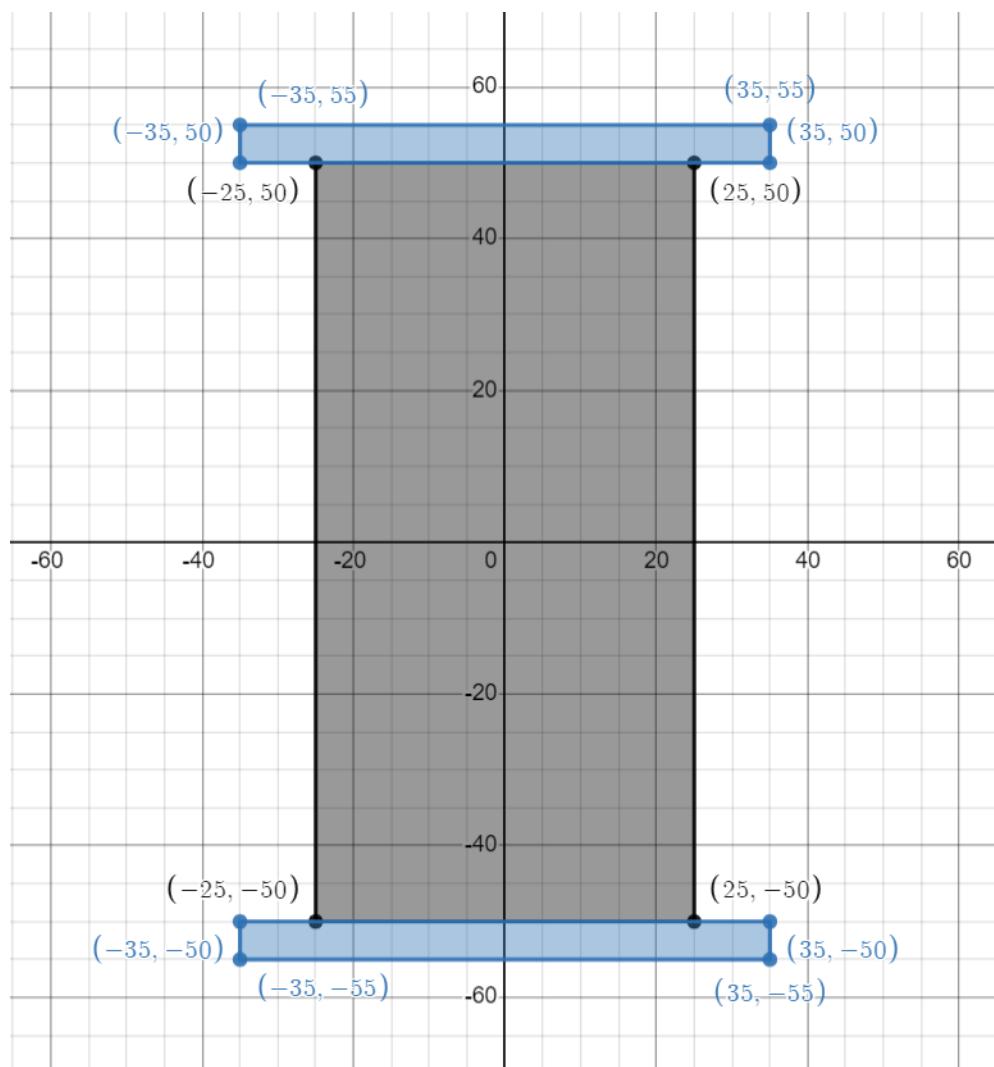


Fig. 38: Figure 1: Geometry of the uniaxial compression strength test

2.1 Initialize the model

1. Create a new run and clean up the old memories.

```
of.new
```

2. Set up the folder name to save your results. Here the folder name is set to be “result”.

```
of.set.result "result"
```

3. Turn off the contact

```
of.set.contact off
```

4. Set the minangle to delete the bad segments for joints and DFNs.

```
of.geometry.minangle 0.0
```

5. Set the number of cores to be used for running this model.

Warning: In this example, 15 cores will be used for running. To prevent the crushing of your computer, please check your computer to fill in the number of cores you want to use from your computer. To check the number of cores your computer has, you can go Task Manager > Performance > CPU > Cores. If nothing is set here, serial computing will be used.

```
of.set.omp 15
```

2.2 Create the geometry and the mesh

1. See Figur 1 to create a polygon with 9 points.

```
of.geometry.polygon 'plate' size 9
[-30e-3 30e-3]
[-30e-3 0.01e-3]
[-1e-3 0.01e-3]
[-0e-3 0.0]
[-1e-3 -0.01e-3]
[-30e-3 -0.01e-3]
[-30e-3 -30e-3]
[30e-3 -30e-3]
[30e-3 30e-3]
```

2. Cut the joints in the plate.

```
of.geometry.cut.joint 'Line' 'plate' [-12e-3 0 30e-3 0.0]
```

3. Group the geometries.

```
of.geometry.nodal.group 'tip_Zone' range box.in [0e-3 1e-3 -0.050e-3 0.050e-3 ]
of.geometry.nodal.group 'cohesive_Zone' range box.in [9e-3 11e-3 -30e-3 30e-3]
```

4. Set different mesh size for different group of geometries.

```
of.geometry.mesh.size 'default' 2.0e-3
of.geometry.mesh.size 'Line' 0.05e-3
of.geometry.mesh.size 'cohesive_Zone' 0.1e-3
of.geometry.mesh.size 'tip_Zone' 0.05e-3
```

5. Create the mesh.

```
of.geometry.mesh delaunay
```

6. Group the elements in a mesh.

```
of.group.element 'Mat2' range plane.below x0 10e-3 y0 -40e-3 x1 10e-3 y1 40e-3
```

2.3 Assign Material and Thermal Properties

1. Set the material properties.

```
of.mat.element 'plate' elastic den 2700 E 30e9 v 0.3 lc 1.0e-4 Gc 200 nita 1
  ↵damp 0.0
of.mat.element 'Mat2' elastic den 2700 E 20e9 v 0.3 lc 1.0e-4 Gc 100 nita 1
  ↵damp 0.0
```

2. Assign materials contacts.

```
of.mat.contact 'default' MC fric 0.3
```

2.4 Create Groups and Assign Boundary Conditions

1. Group the nodes to prepare for the next step.

```
of.group.edge 'up_b' range plane.on x0 -30e-3 y0 30e-3 x1 30e-3 y1 30e-3
of.group.edge 'bottom_b' range plane.on x0 -30e-3 y0 -30e-3 x1 30e-3 y1 -30e-3
```

2. Assign the pressure boundary conditions.

```
of.boundary.edge.pressure 'up_b' normal -1 ntable stress
of.boundary.edge.pressure 'bottom_b' normal -1 ntable stress
```

2.5 Set the Outputs

1. Set the output interval to be every 2000 steps and output all fields variables and fracture variables. Control the output interval to a reasonable size could shorten the computation time but get a good understanding of the model.

```
of.history.pv.interval 2000
of.history.pv.field default
```

2. The program will run 100000 steps in total. In other words, it will output 10 files for reference.

```
of.step 100000
```

3. Finalize the model and clear all the temporary memories.

```
of.finalize
```

4. Save the notepad and double click the .of file to run the program.

5.11.3 3.0 Run the Program

When you run the program, you can first check the mesh that was created by Gmsh as shown in Figure 2. If the mesh has a good quality, you can close the window to continue run the program.

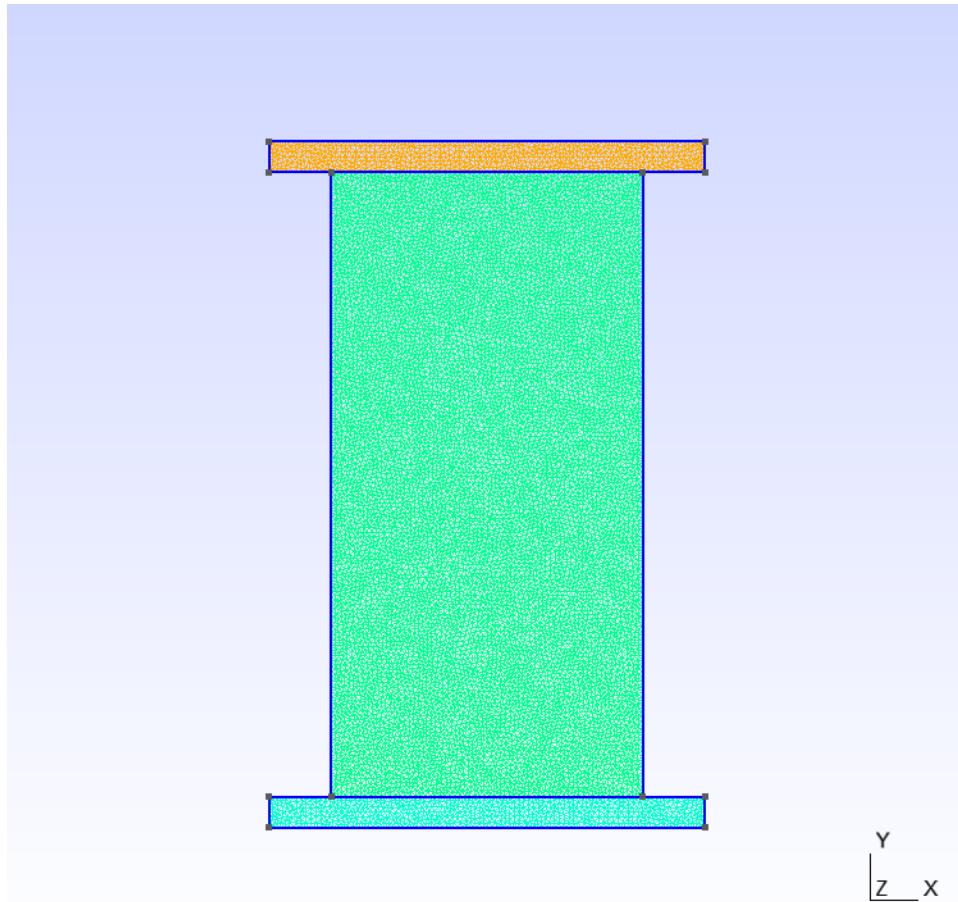


Fig. 39: Figure 2: Mesh created by Gmsh

The mesh is using triangle elements.

| group ID | Type | Number of Object | Tags | Extra |
|-----------------|---------|------------------|------|--------------|
| 0 | Node | 782 | up | |
| 1 | Node | 773 | down | |
| 2 | Element | 20622 | rock | Triangle, T3 |
| 3 | Element | 1372 | up | Triangle, T3 |
| 4 | Element | 1354 | down | Triangle, T3 |
| Total nodes | | 63421 | | |
| Total elements | | 23348 | | |
| Total cohesives | | 31121 | | |
| Total entities | | 20624 | | |

Fig. 40: Figure 3: Check the mesh setting from command window

Material properties you set on step 2.3 will be shown on the screen. You can confirm it while the program just starts to run.

| Checking model material information ... | | | | |
|---|-------------------------------|-------------------|---------|--|
| ID | Type | Model type | Tags | Properties |
| 0 | Matrix material | Linear elastic | default | E = 30.00 GPa v = 0.30 density = 2700.00 kg/m^3 damp = 1.00 |
| 1 | Cohesive material | Evans-Marathé FPZ | default | pn = 500.00 GPa pt = 230.77 GPa tension = 1.00 MPa cohesion = 3.00 MPa GI = 10.00 J/m^2 GII = 50.00 J/m^2 friction = 0.30 |
| 2 | Contact material (default) | Mohr-Coulomb law | default | kn = 500.00 GPa, ks = 230.77 GPa friction = 0.30 |

Fig. 41: Figure 4: Check the material assignment from command window

Furthermore, node boundaries are shown at the header of the program.

| Checking model boundary information ... | | | | |
|---|------------------|---------------|------------|-----------------------|
| Bou ID | Category objects | Boundary type | Group Tags | Info and Extra |
| 0 | Node boundary | Velocity | up | x fixed y = -0.05 m/s |
| 1 | Node boundary | Velocity | down | x fixed y = 0.05 m/s |

Fig. 42: Figure 5: Check the boundary information from command window

Global timestep is shown here.

```

Info    :critical timestep from MATRIX is: [1.23e-07].
Info    :critical timestep from INTERFACE is: [1.83e-10].
Info    :critical timestep from CONTACT is: [1.4le-08].
Info    :Warning: the user-defined timestep is greater than the critical timestep required.
Info    :global timestep is: [5.00e-09].

```

Fig. 43: Figure 6: Check the timesteps from command window

Start to run the program. Results are logged for every 2000 steps.

| Start running ... | | | | | | |
|-------------------|------------------|--------------------|---------------|------------|---------------|-------------|
| step | physical time(s) | unbalance force(N) | kinetic ratio | broken CZM | contact pairs | run time(s) |
| 0 | 0.000e+00 | 0.000e+00 | 0.000e+00 | 0 | 621 | 0.000 |
| 2000 | 1.000e-05 | 2.846e+02 | 2.426e-04 | 0 | 626 | 29.437 |
| 4000 | 2.000e-05 | 3.472e+02 | 4.435e-04 | 0 | 630 | 60.029 |

Fig. 44: Figure 7: Start to run the program

5.11.4 4.0 Results

The stress and displacement results here are for reference.

5.11.5 5.0 Full Script

- Phase_Field.of (click to download from Gitlab)

```

# initialization, this command is to clear the memory in your last run, it is
# not mandatory
# but strongly recommend.
of.new

# Set the path folder of your output results, //result// in the same path of
# your input file
# will be created by default
of.set.result "result"

# turn off the contact
of.set.contact off

# Set the minangle to delete the bad segments for jsets and dfns.
of.geometry.minangle 0.0

# Set the number of cores you want to use, the parallelization will be
# automatically turned on
# when you use the following command, otherwise the serialization will be used

```

(continues on next page)

(continued from previous page)

```

→by default
of.set.omp 15

#####
# create mesh #####
→#####
of.geometry.polygon 'plate' size 9
[-30e-3 30e-3]
[-30e-3 0.01e-3]
[-1e-3 0.01e-3]
[-0e-3 0.0]
[-1e-3 -0.01e-3]
[-30e-3 -0.01e-3]
[-30e-3 -30e-3]
[30e-3 -30e-3]
[30e-3 30e-3]

# Cut the joints in the plate.
of.geometry.cut.joint 'Line' 'plate' [-12e-3 0 30e-3 0.0]

of.geometry.nodal.group 'tip_Zone' range box.in [0e-3 1e-3 -0.050e-3 0.050e-3
→]
of.geometry.nodal.group 'cohesive_Zone' range box.in [9e-3 11e-3 -30e-3 30e-3]

of.geometry.mesh.size 'default' 2.0e-3
of.geometry.mesh.size 'Line' 0.05e-3
of.geometry.mesh.size 'cohesive_Zone' 0.1e-3
of.geometry.mesh.size 'tip_Zone' 0.05e-3

of.geometry.mesh delaunay

of.group.element 'Mat2' range plane.below x0 10e-3 y0 -40e-3 x1 10e-3 y1 40e-3

#####
# assign material parameters #####
→#####
# assign material for matrix
of.mat.element 'plate' elastic den 2700 E 30e9 v 0.3 lc 1.0e-4 Gc 200 nita 1
→damp 0.0
of.mat.element 'Mat2' elastic den 2700 E 20e9 v 0.3 lc 1.0e-4 Gc 100 nita 1
→damp 0.0
# assign materials for contacts
of.mat.contact 'default' MC fric 0.3

#####
# create groups #####
→#####
of.group.edge 'up_b' range plane.on x0 -30e-3 y0 30e-3 x1 30e-3 y1 30e-3
of.group.edge 'bottom_b' range plane.on x0 -30e-3 y0 -30e-3 x1 30e-3 y1 -30e-3

#####
# assign boundaries #####
→#####
# import the data table
of.import.table stress 'load.dat'

```

(continues on next page)

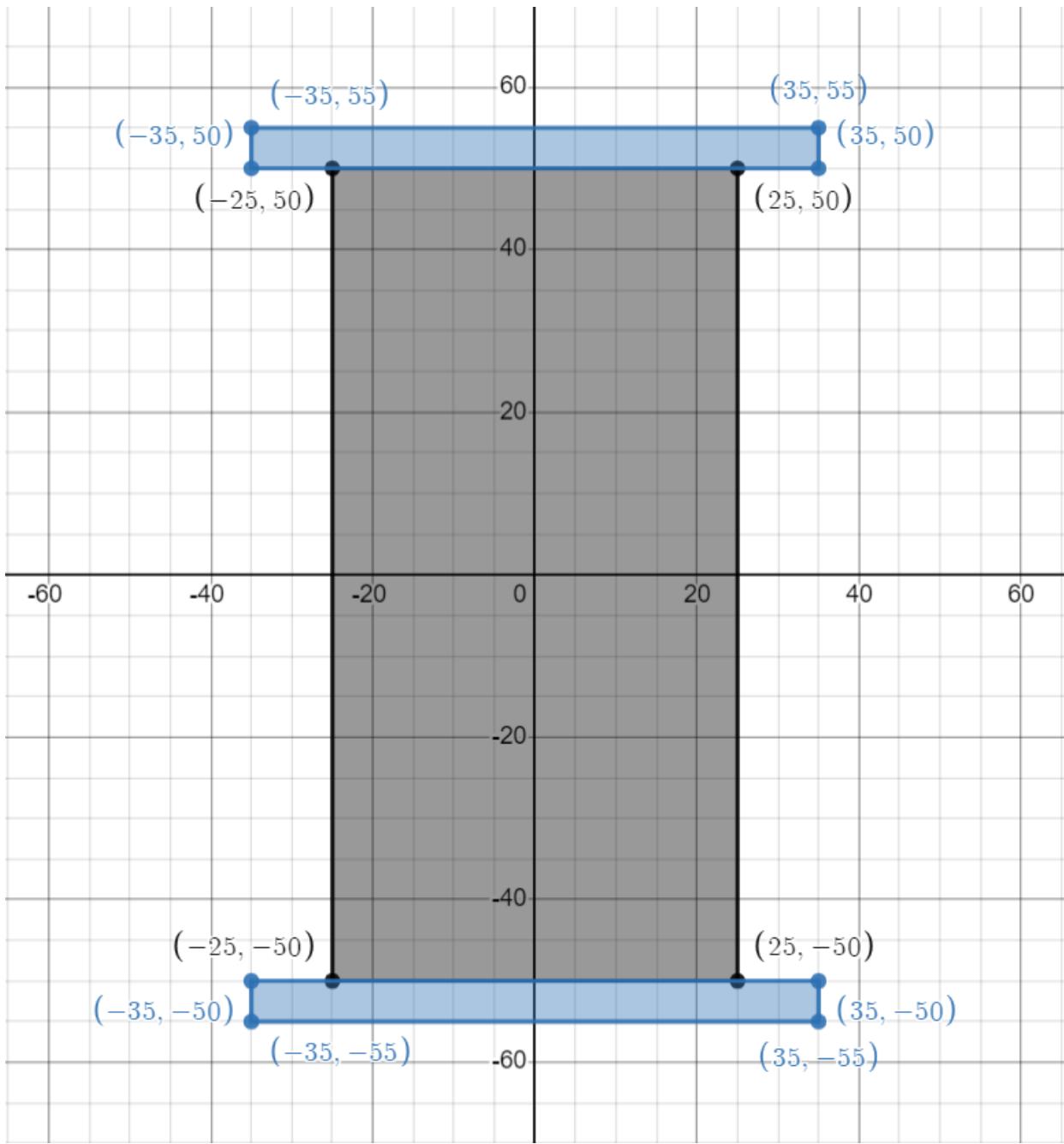
(continued from previous page)

```
# Assign the boundary conditions
of.boundary.edge.pressure 'up_b' normal -1 ntable stress
of.boundary.edge.pressure 'bottom_b' normal -1 ntable stress

#####
# Set the interval of writing ParaView results.
of.history.pv.interval 2000
of.history.pv.field default

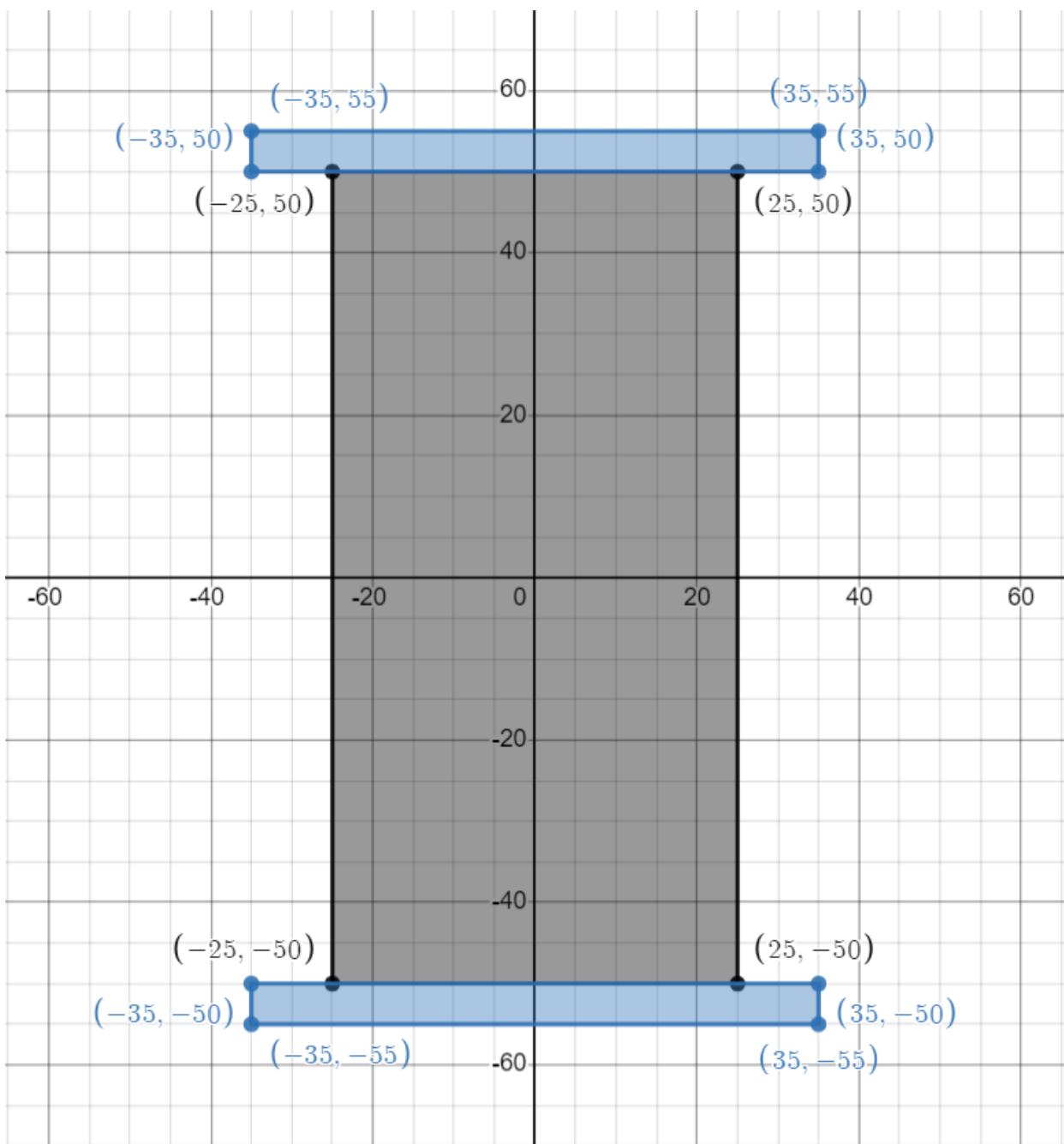
#####
# to excuate the model
of.step 100000
# finalize the model and clear the memory
of.finalize
```

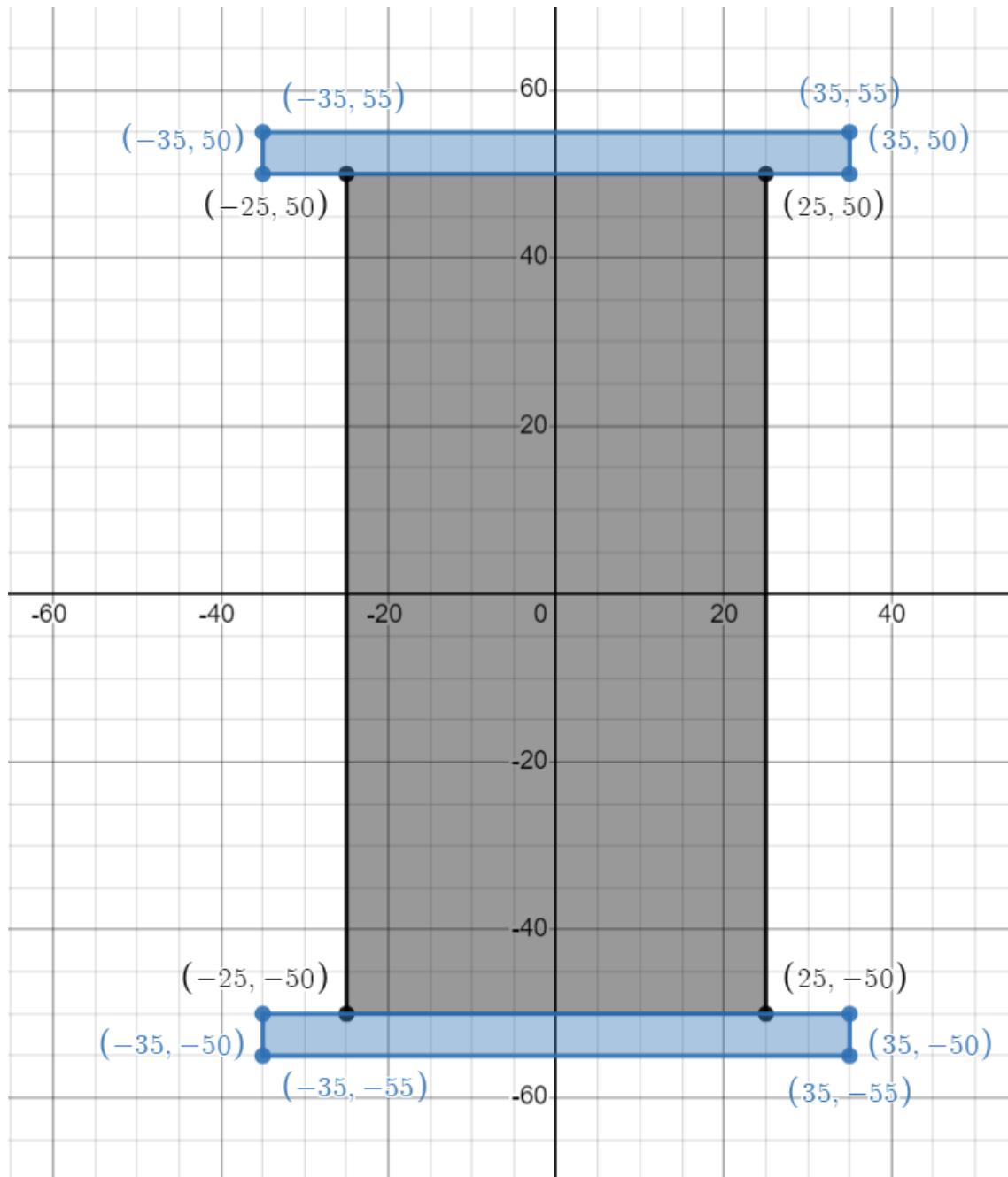
5.12 Create the Geometry



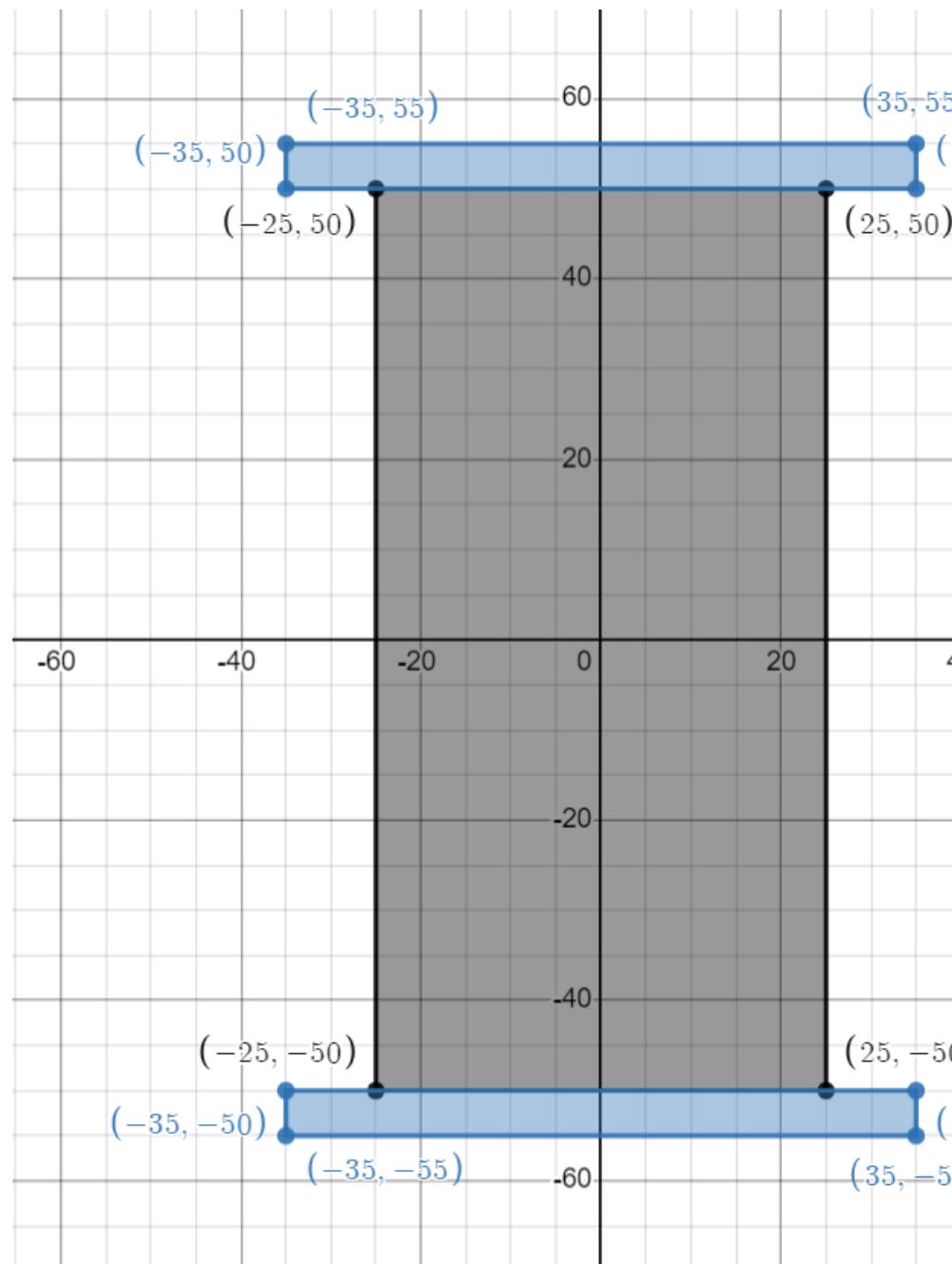
tutorial0_Geometry

5.13 Lab Tests

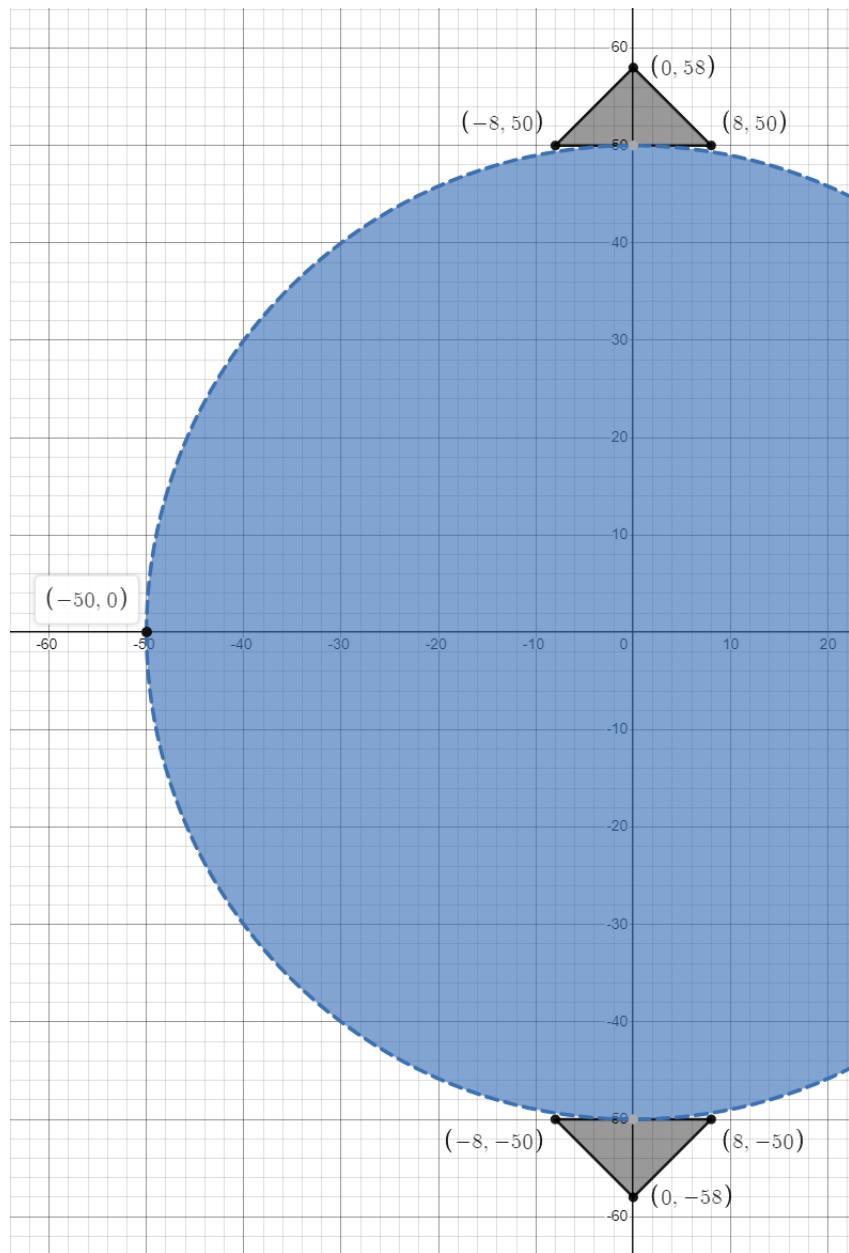




tutorial1_UCS_Test



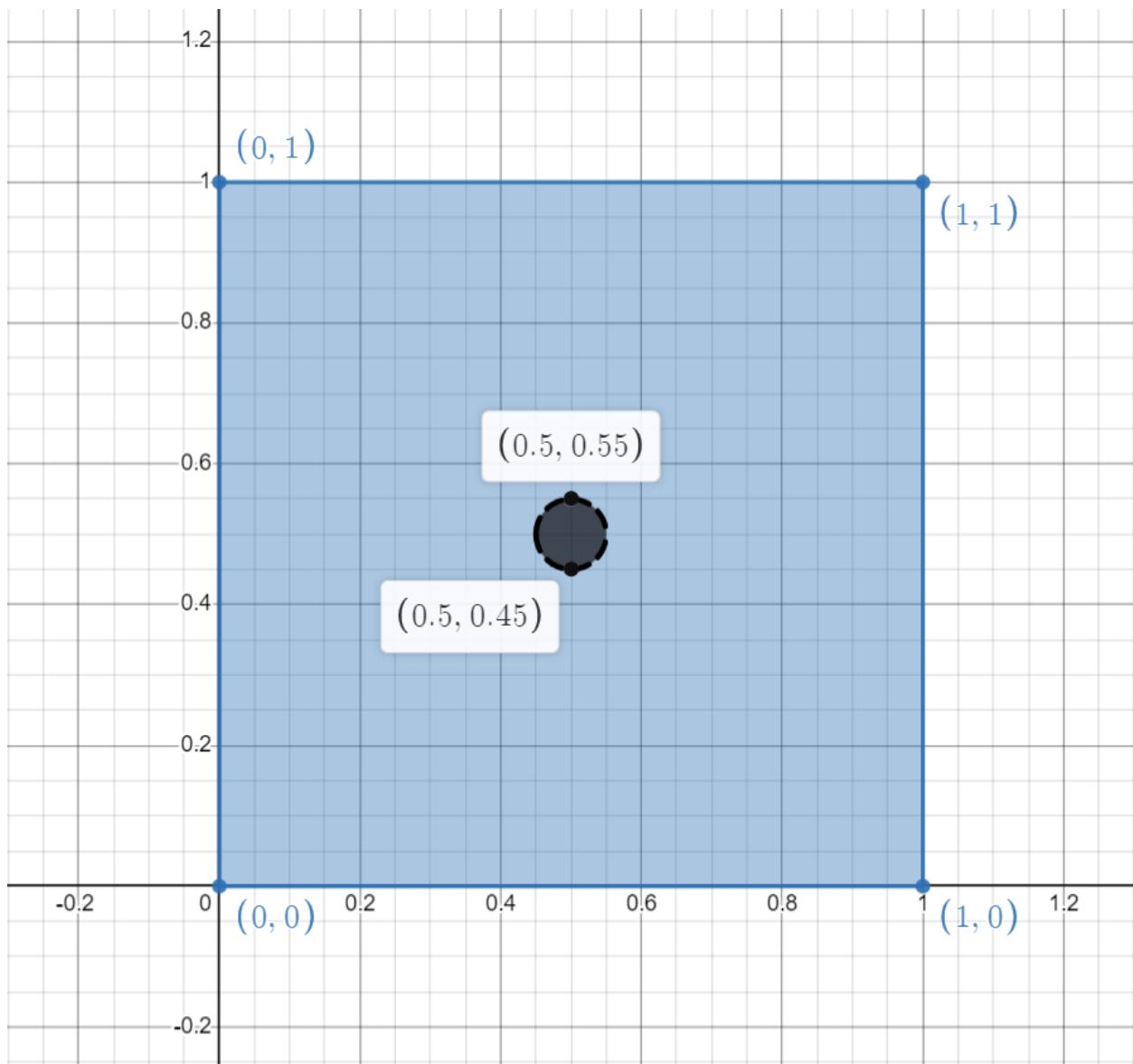
tutorial2_UCS_with_Gmsh



[tutorial3_UCS_Test_with_Quadrangle_Elements](#)

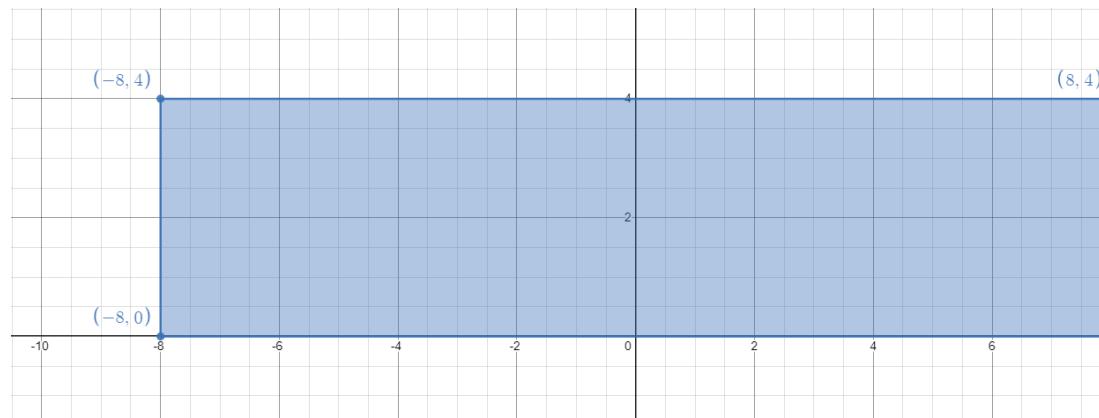
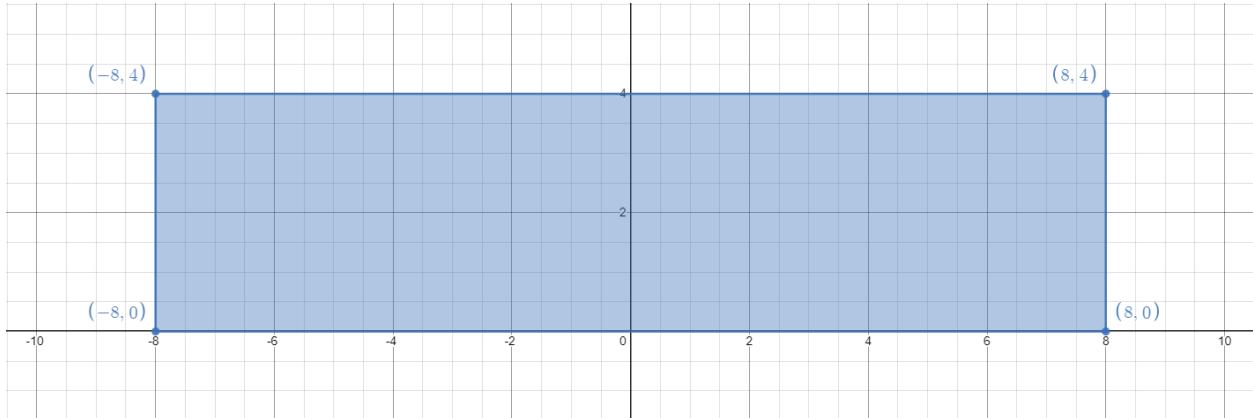
[tutorial4_BD_Test](#)

5.14 Excavation



tutorial5_In_Situ_Stress

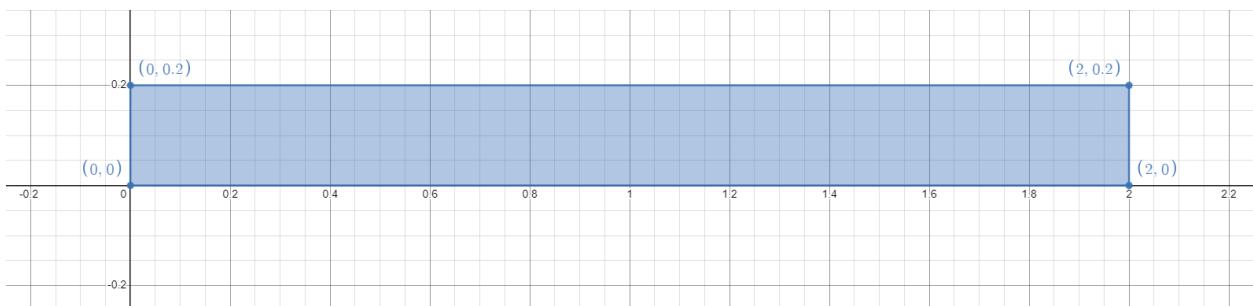
5.15 Hydro Module



tutorial6_Hydro_Seepage

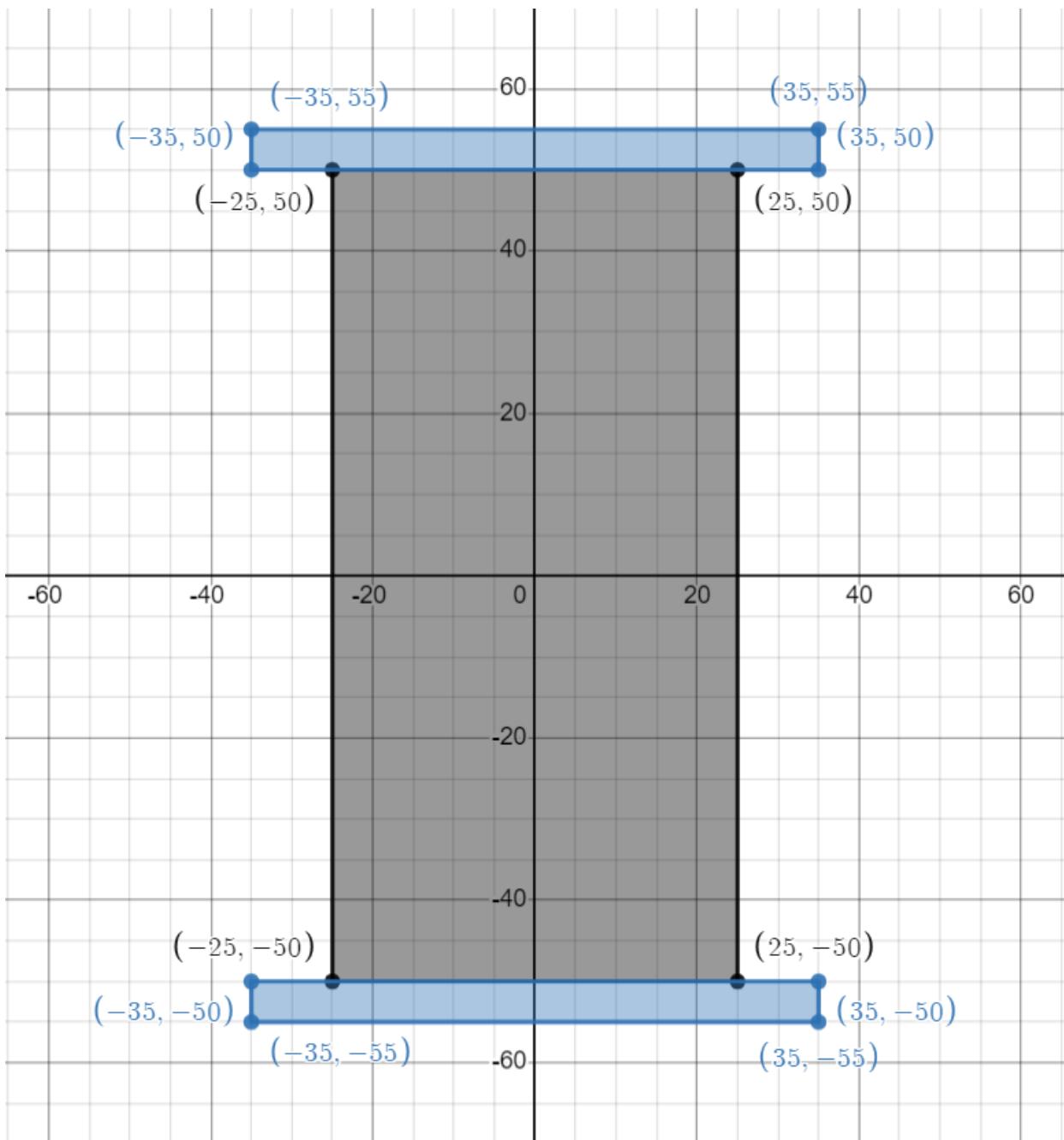
tutorial7_Hydro_Fracture_Flow

5.16 Thermal Module



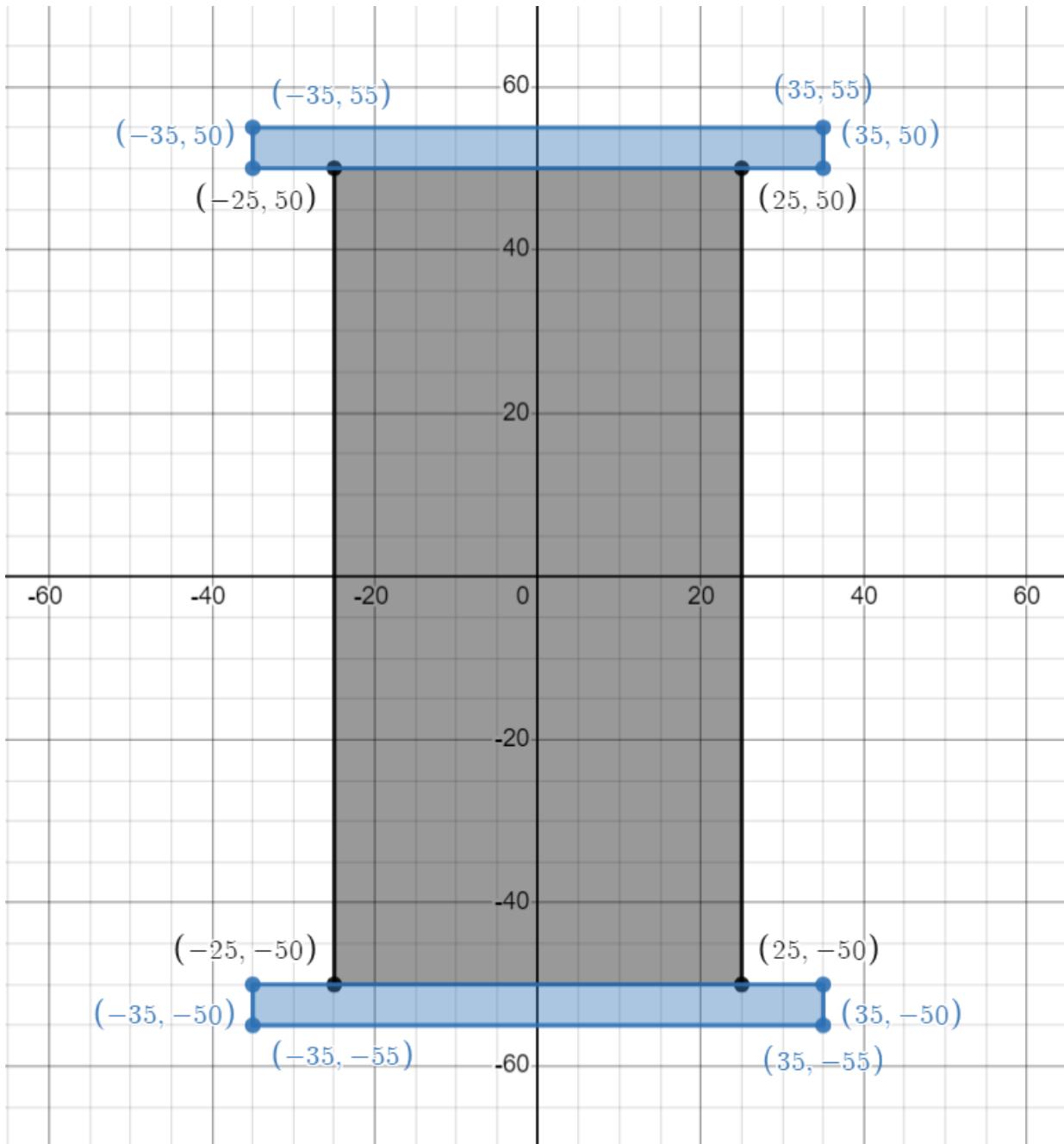
tutorial8_Thermal_Flux

5.17 GBM



tutorial9_GBM

5.18 Phase Field



tutorial10_Phase_Field

USER GUIDE

6.1 General

6.1.1 of.console.interval

of.console.interval *interval* <int>

Set the interval to print the results on the console screen. A high frequency of outputting rate will reduce the computational speed and could be unnecessary.

Parameters:

interval: (default 1000) The interval for results to print on the screen.

Example:

```
# Default
of.console.interval 1000
```

6.1.2 of.damp.global

of.damp.global *ratio* <Real>

Set the global damping value to slow down the loading velocity, which is helpful for balance. **The damping value should not be over than 1.0.**

Parameters:

ratio: (default 0.7) Global damping value for slow down the loading velocity.

Example:

```
# Default
of.damp.global 0.7
```

6.1.3 of.finalize

of.finalize

(Mandatory) This command terminates the console at the end of the run and clear all the old memory from the last run.

Example:

```
# "...Code about the model..."  
# Last line of the code  
of.finalize
```

6.1.4 of.import

of.import file_name <string>

Import the file, the file type can be:

- .fdem, which is self-defined
- .fem, which is for insitu balance,
- .msh2 or .msh, which is standard mesh file, should not be over version 2.0
- .inp, which is standard mesh file,
- .geo, which is standard geometry file
- .stl, which is for profile
- .dxf, which is for the geometry
- .step, which is for the standard geometry

Example:

```
# Example  
of.import "mesh.msh"
```

6.1.5 of.import.table

of.gravity table_tag <string> file_path <string>

Import the table, the file type can be arbitrary.

1. Y-only table

```
<Table Type> time [start time, end time] <N data>  
<data.0>  
<data.1>  
...
```

2. X-Y table

```
<Table Type> <N data>
<data.x0> <data.y0>
<data.x1><data.y1>
...
...
```

Parameters:

table_tag: the tag to this table of data.

file_path: file path of the table.

Example:

```
# Example
of.import.table 'pwave' "inputPwave.dat"
```

6.1.6 of.log.interval**of.log.interval *interval* <int>**

Set the interval to print the results in the .log file. A high frequency of outputting rate will reduce the computational speed and could be unnecessary.

Parameters:

interval: (default 2000) interval to print the results in .log file.

Example:

```
# Default
of.log.interval 2000
```

6.1.7 of.new**of.new**

(Recommend) Clear all old memories from the last run and start a new run. It is used as the first line in the input file. It also prevents the conflict when two programs run at the same time.

Example:

```
# First line of the program
of.new
```

6.1.8 of.restore

of.restore *file_name* <string>

Restore the model data.

Note: Also see **of.save**.

Parameters:

file_name: File name of the data file.

Example:

```
of.restore 'ucs.json'
```

6.1.9 of.save

of.save *file_name* <string>

Save the model. OpenFDEM supports data to be saved in .json, .xml and .bin formats.

Note: Also see **of.restore**.

Parameters:

file_name: File name of the data file.

Example:

```
of.save 'usc.json'
```

6.1.10 of.save.FEM

of.save.FEM *file_name* <string>

(Mandatory for in-situ stress problems) Save the FEM results after in-situ stress equilibrium is reached. This file will write out the mesh information after deformed for the next step excavation. This file avoids rebalancing your model.

Parameters:

file_name: Save the FEM results under this file name.

Example:

```
# Example
of.save.FEM 'insitu.fem'
```

6.1.11 of.set.config

of.config *type* <string>

Choose the plane stress or plane strain for the 2D problems. Plane stress is set by default.

Parameters:

type: (default planestress) 2D problem can be solved based on plane stress or plane strain.

Example:

```
# Default
of.set.config planestress
# Alternative
of.set.config planestrain
```

6.1.12 of.set.contact

of.contact *mode* <bool>

Set the Boolean to consider contact detection and contact force computation or to consider no contacts by turning it off.

Parameters:

mode: turn on or turn off the contact mode.

Example:

```
# Default
of.set.contact ON
# Turn the contact mode OFF
of.set.contact OFF
```

6.1.13 of.set.debug

of.debug *mode* <bool>

Set the Boolean to whether turn on the debug mode. Please noted that debug means writing log information in the .log file, not the compiled debug mode.

Parameters:

mode: turn on or turn off the debug mode.

Example:

```
# Default
of.set.debug ON
# Debug mode OFF
of.set.debug OFF
```

6.1.14 of.set.GBM

of.set.GBM *group_count* <int> *group1_name* <string> *ratio1* <double> *group2_name* <string> *ratio2* <double>...

Assign the number of groups for minerals in GBM module and assign the group tags and area ratio for different minerals. The groups are issued to elements. The accumulation of all ratios should equal to 1.0.

Parameters:

group_count: Number of groups will be assigned.

group_name: The name of mineral groups.

ratio: The ratio of mineral in the model. The sum of mineral ratio should be 1.0.

Example:

```
# Example
of.set.GBM 3 'Quartz' 0.4 'Feldspar' 0.3 'Mica' 0.3
```

6.1.15 of.set.gravity

of.gravity x <double> y <double>

Add gravity to the model. Gravity is a body force, which is issued by x and y components. Negative is down forward by default.

Parameters:

x: gravitational acceleration in the x directions.

y: gravitational acceleration in the y directions.

Example:

```
# Default
of.set.gravity x 0.0 y 0.0
# Method 1
of.set.gravity x 0.0 y -9.8
# Method 2
of.set.gravity y -9.8
```

6.1.16 of.set.large

of.set.large *switch* <bool>

Set the flag of large deformation for the kinematics.

Example:

```
of.set.large on
of.set.large off
```

6.1.17 of.set.massscale

of.set.massscale *mass_scale* <Real>

Set the mass scale value to lump the nodal mass and give a larger timestep. That will be useful to balance your model when kinetics is not important. Be careful to use if you are beginner.

Parameters:

mass_scale: (default 1.0)

Example:

```
# Default
of.set.massscale 1.0
```

6.1.18 of.set.module

of.set.module *type* <string> *switch* <bool>

Turn on or off for special modules.

Parameters:

type: Modules currently available are ‘hydro’, ‘thermal’, ‘blast’ and ‘dynamic’. For some special modules which are not listed here, please contact the developer for the access.

Example:

```
# Hydro modules
of.set.config hydro on
of.set.config hydro off

# Thermal modules
of.set.config thermal on
of.set.config thermal off

# Blast modules
of.set.config blast on
```

(continues on next page)

(continued from previous page)

```
of.set.config blast off  
  
# Dynamic modules  
of.set.config dynamic on  
of.set.config dynamic off
```

6.1.19 of.set.omp

of.omp *number_of_cores* <int>

Assign the count of cores you want to use for CPU parallelization. The parallelization will be turned on only when the computer has more than 2 cores.

Note: Go Task Manager > Performance > CPU > Cores to check the number of cores your computer has. If nothing is set here, serial computing with 1 core will be used.

Parameter:

number_of_cores: (default 1) The number of cores will be used for computation.

Example:

```
# Default  
of.set.omp 1  
# Example  
of.set.omp 16
```

6.1.20 of.set.result

of.set.result *path* <string>

Set the path to store the result files.

Parameters:

path: (default/result) the folder path to save the results.

Example:

```
# Default
of.set.result "result"
# Example
of.set.result "C:/user/document/openfdem/example/result"
```

6.1.21 of.set.rgbm

of.set.rGBM *group_count* <int> *group1_name* <string> *greyvalue1* <int> *group2_name* <string> *greyvalue2* <double>...

Assign the number of groups for minerals to create the realistic GBM model, and assign the group tags, grey levels for different minerals. The groups are issued to elements. The grey levels ranging from 0 -255 are used for separating different minerals.

Parameters:

group_count: Number of groups will be assigned.

group_name: The name of mineral groups.

greyvalue: They grey level of mineral ranges from 0-255.

Example:

```
# Example
of.set.rGBM 3 'Quartz' 76 'Feldspar' 153 'Mica' 255
```

6.1.22 of.solve

of.solve *mechanical_ratio* <double>

(**Mandatory**) Assign the mechanical ratio as the iteration threshold to exit the running. The double value ζ is the ratio of the total maximum unbalance force to the total force. Generally, the ratio less than 1.0e-5 can be considered as the equilibrium state. This command is important to determine the equilibrium state in in-situ stress equilibrium or static problems.

$$\zeta = \frac{\sum_{i=1}^n \sqrt{\bar{f}_i}}{\sum_{i=1}^n \sum_{j=1}^n \sqrt{f_{i,j}}}, \bar{f}_i \text{ is the unbalanced force on node } i. j \text{ are types of the nodal forces. } f_{i,j} \text{ is the } j \text{ th type force on node } i.$$

Note: Also see the alternative of the mandatory option **of.step**

Parameter:

mechanical ratio: (default 0.0001) the maximum ratio between the total maximum unbalance force to the total force before termination.

Example:

```
# Default  
of.solve 0.0001
```

6.1.23 of.step

of.step *number_of_steps* <int>

(**Mandatory Option**) Assign the number of steps for the current run. The run will be terminated after the steps reach the assigned count, but the console will not be terminated until it meets *of.finalize*.

Note: Also see the alternative of the mandatory option **of.solve**

parameter:

number_of_steps: the number of steps program will run before termination.

Example:

```
of.step 30000
```

6.1.24 of.timestep

of.timestep *type* <string> *timestep* <Real>

Set the user defined timestep. The type can be “fix” or “auto”. OpenFDEM will compute the default timestep which can result in stable results. Be careful to set the timestep yourself only when you are rather enough understand what’s explicit dynamic modelling.

OpenFDEM will print out the default timestep and give a warning to you if your defined timestep is larger than the default value.

Parameters:

type: (default ‘auto’) ‘auto’ provides a timestep size evaluated by OpenFDEM, but users can use ‘fix’ to set the user defined timestep.

timestep: (optional) fixed time step

Example:

```
# Default
of.timestep auto
# Fixed time step
of.timestep fix 1.0e-9
```

6.2 Blast

6.2.1 of.blast.position

of.blast.position *borehole_tag* <string> x <double> y <double> radius <double>

Create a blast borehole by its center and radius.

Parameters:

borehole_tag: tag of the group
x: x coordinate of the center of the borehole
y: y coordinate of the center of the borehole
radius: the radius of the borehole.

Example:

```
of.blast.position 'borehole1' x 0.0 y 0.0 radius 1.0
```

6.3 Boundary

6.3.1 of.boundary.nodal.force

of.boundary.nodal.force *nodal_tag* <string> x <double> y <double> r <double> xtable <string_tag> ytable <string_tag> rtable <string_tag>

Create nodal force boundaries on nodal groups. r is for the moment. The force can be time dependent if you use the table. The x, y, r values will be the factor if you use the table. That is

$$f[i] = x \times \text{table}[i]$$

Parameters:

nodal_tag: tag of the group
x: (default 0.0) Nodal forces in the x direction or the amplitude of x table.
y: (default 0.0) Nodal forces in the y direction or the amplitude of y table.
r: (default 0.0) Nodal moments or the amplitude of r table.
xtable: (default -1) time dependent nodal force in x direction.
ytable: (default -1) time dependent nodal force in y direction.
rtable: (default -1) time dependent moment.

Example:

```
of.boundary.nodal.force 'bottom' x -1 xtable 'pwave'
```

6.3.2 of.boundary.nodal.velocity

```
of.boundary.nodal.velocity *nodal_tag* <string> x <double> y <double> r <double> xtable <string_tag>
ytable <string_tag> rtable <string_tag>
```

Create the nodal velocity boundary on the nodal groups. r is for the angular velocity, in rad/s. The velocity can be time dependent if you use the table. The x, y, r values will be the factor if you use the table. That is

$$v[i] = x \times \text{table}[i]$$

Parameters:

nodal_tag: tag of the group
x: (default 0.0) Velocity in the x direction or the amplitude of x table.
y: (default 0.0) Velocity in the y direction or the amplitude of y table.
r: (default 0.0) Angular velocity
xtable: (default -1) time dependent velocity in x direction.
ytable: (default -1) time dependent velocity in y direction.
rtable: (default -1) time dependent angular velocity.

Example:

```
# fix in x direction, y direction and rotation
of.boundary.nodal.velocity 'up' x 0.0 y 0.0 r 0.0
# assign the table which is named as pwave
of.boundary.nodal.velocity 'up' x 1.0 xtable 'pwave'
```

6.3.3 of.boundary.nodal.inivelocity

of.boundary.nodal.inivelocity *nodal_tag* <string> x <double> y <double> r <double> xtable <string_tag> ytable <string_tag> rtable <string_tag>

Create the nodal initial velocity boundaries on the nodal groups. r is for the angular velocity, in rad/s. The velocity will be applied only at the initial.

Parameters:

- nodal_tag:** tag of the group
- x:** (default free) Initial velocity in the x direction.
- y:** (default free) Initial elocity in the y direction.
- r:** (default free) Initial angular velocity

Example:

```
# an initial impact velocity of the ball is 100m/s
of.boundary.nodal.inivelocity 'ball' y -100
```

6.3.4 of.boundary.nodal.acceleration

of.boundary.nodal.acceleration *nodal_tag* <string> x <double> y <double> r <double> xtable <string_tag> ytable <string_tag> rtable <string_tag>

Create the nodal acceleration boundary on the nodal groups. r is for the angular acceleration, in rad/s^2 . The acceleration can be time dependent if you use the table. The x, y, r values will be the factor if you use the table. That is $a[i] = x \times table[i]$

Parameters:

- nodal_tag:** tag of the group
- x:** (default 0.0) Acceleration in the x direction or the amplitude of x table.
- y:** (default 0.0) Acceleration in the y direction or the amplitude of y table.
- r:** (default 0.0) Angular acceleration.
- xtable:** (default -1) time dependent acceleration in x direction.

ytable: (default -1) time dependent acceleration in y direction.

rtable: (default -1) time dependent angular acceleration.

Example:

```
# ball will be give an acceleration of 100m/s^2  
of.boundary.nodal.acceleration 'ball' y -100
```

6.3.5 of.boundary.nodal.clear

of.boundary.nodal.clear *nodal1_tag* <string> *nodal2_tag* <string> *nodal3_tag* <string> ...

Delete the nodal boundaries defined previously. Directly set the nodal groups.

Parameters:

nodal_tag: tag of the nodal group

Example:

```
of.boundary.nodal.clear 'leftedge' 'rightedge' 'up' 'down'
```

6.3.6 of.boundary.edge.force

of.boundary.edge.force *edge_tag* <string> normal <double> shear <double> ntable <string_tag> stable <string_tag>

Create the edge force boundary on the edge groups. The force can be time dependent if you use the table. The normal shear values will be the factor if you use the table. That is $f[i] = x \times table[i]$

The force will be automatically applied on the two nodes of the edge and consider the edge direction. Normal direction is the local normal direction, and towards to inside. Shear is along the edge, and counterclockwise.

Parameters:

nodal_tag: tag of the group

normal: (default 0.0) Edge forces in the local normal direction or the amplitude of n table.

shear: (default 0.0) Edge forces in the local shear direction or the amplitude of s table.

ntable: (default -1) time dependent force in the local normal direction.

stable: (default -1) time dependent force in the local shear direction.

Example:

```
of.boundary.edge.force 'bottom' normal 1000
```

6.3.7 of.boundary.edge.velocity

of.boundary.edge.velocity *edge_tag* <string> normal <double> shear <double> ntable <string_tag> stable <string_tag>

Create the edge velocity boundary on the edge groups. The velocity can be time dependent if you use the table. The normal shear values will be the factor if you use the table. That is $v[i] = x \times \text{table}[i]$

The edge will be automatically applied on the two nodes of the edge and consider the edge direction. Normal direction is the local normal direction, and towards to inside. Shear is along the edge, and counterclockwise

Parameters:

nodal_tag: tag of the group

normal: (default 0.0) Edge velocity in the local normal direction or the amplitude of n table.

shear: (default 0.0) Edge velocity in the local shear direction or the amplitude of s table.

ntable: (default -1) time dependent velocity in the local normal direction.

stable: (default -1) time dependent velocity in the local shear direction.

Example:

```
# pin the up
of.boundary.edge.velocity 'up' normal 0.0 shear 0.0
# assign the table which is named as pwave
of.boundary.edge.velocity 'up' normal 1.0 ntable 'pwave'
```

6.3.8 of.boundary.edge.inivelocity

of.boundary.edge.inivelocity *edge_tag* <string> normal <double> shear <double>

Create the edge initial velocity boundaries on the edge groups. The velocity will be applied only at the initial.

Parameters:

nodal_tag: tag of the group

normal: (default free) Edge velocity in the local normal direction or the amplitude of n table.

shear: (default free) Edge velocity in the local shear direction or the amplitude of s table.

Example:

```
# ball will be given an initial impact velocity of 100m/s  
of.boundary.edge.inivelocity 'ball' shear -100
```

6.3.9 of.boundary.edge.acceleration

of.boundary.edge.acceleration *edge_tag* <string> normal <double> shear <double> ntable <string_tag> stable <string_tag>

Create the edge acceleration boundary on the edge groups. The acceleration can be time dependent if you use the table. The normal shear values will be the factor if you use the table. That is $a[i] = x \times table[i]$

Parameters:

nodal_tag: tag of the group

normal: (default 0.0) Acceleration in the normal direction or the amplitude of n table.

shear: (default 0.0) Acceleration in the shear direction or the amplitude of s table.

ntable: (default -1) time dependent acceleration in the normal direction.

stable: (default -1) time dependent acceleration in the shear direction.

Example:

```
# ball will be given an acceleration of 100m/s^2  
of.boundary.edge.acceleration 'ball' shear -100
```

6.3.10 of.boundary.edge.clear

of.boundary.edge.clear *edge1_tag* <string> *edge2_tag* <string> *edge3_tag* <string> ...

Delete the edge boundaries defined previously. Directly set the edge groups.

Parameters:

edge_tag: tag of the edge group

Example:

```
of.boundary.edge.clear 'leftedge' 'rightedge' 'up' 'down'
```

6.3.11 of.boundary.element.stress

of.boundary.element.stress *element_tag* <string> xx <double> xy <double> yy <double>

Create the in-situ stress on the assigned element groups.

Parameters:

element_tag: tag of the group

xx: (default 0.0) In-situ stress in the xx direction

xy: (default 0.0) In-situ stress in the xy direction

yy: (default 0.0) In-situ stress in the yy direction

Example:

```
# the rock is under 10 MPa in x direction and 30 MPa is y direction. Negative is compression.
of.boundary.element.stress 'rock' xx -10e6 yy -30e6
```

6.3.12 of.boundary.element.stress.xgrad

of.boundary.element.stress.xgrad *element_tag* <string> xx <double> xy <double> yy <double>

Create the in-situ stress gradient in x direction on the assigned element groups.

Parameters:

element_tag: tag of the group

xx: (default 0.0) In-situ stress in the xx direction

xy: (default 0.0) In-situ stress in the xy direction

yy: (default 0.0) In-situ stress in the yy direction

Example:

```
# the rock is under 10 MPa in x direction and 30 MPa is y direction. Negative is ↵
compression.
of.boundary.element.stress.xgrad 'rock' xx -10e6 yy -30e6
```

6.3.13 of.boundary.element.stress.ygrad

of.boundary.element.stress.ygrad *element_tag* <string> xx <double> xy <double> yy <double>

Create the in-situ stress gradient in y direction on the assigned element groups.

Parameters:

element_tag: tag of the group

xx: (default 0.0) In-situ stress in the xx direction

xy: (default 0.0) In-situ stress in the xy direction

yy: (default 0.0) In-situ stress in the yy direction

Example:

```
# the rock is under 10 MPa in x direction and 30 MPa is y direction. Negative is ↵
compression.
of.boundary.element.stress.ygrad 'rock' xx -10e6 yy -30e6
```

6.3.14 of.boundary.element.clear

of.boundary.element.clear *element_tag1* <string> *element_tag2* <string> *element_tag3* <string> ...

Delete the element boundaries defined before. Directly set the element groups

Parameters:

element_tag: tag of the group to be cleared.

Example:

```
of.element.clear 'rock' 'soil'
```

6.3.15 of.boundary.blast

of.blast.position *borehole_tag* <string> p0 <double> table <string>

Create the blast borehole boundaries from table.

Parameters:

borehole_tag: tag of the group

p0: initial position.

table: time dependent position.

Example:

```
of.boundary.blast 'borehole1' p0 1.0 table 'pwave'
```

6.4 DFN**6.4.1 of.DFN.connectivity**

of.DFN.connectivity *dfn_tag* <string> *N* <int> *node0* <int> *node1* <int> *node2* <int> ...

Create DFN groups by knowing its node ids.

Parameters:

dfn_tag: tag of the DFN group

N: size of the group

node: node id

Example:

```
# 3 DFN ids in this DFN group
of.DFN.connectivity 'dfn1' 3 0 14 27 26 48 56
```

6.4.2 of.DFN.group

of.DFN.group *dfn_tag* <string> *N* <int> *dfn_id1* <int> *dfn_id2* <int> ...

Create DFN groups by knowing its DFN ids.

Parameters:

dfn_tag: tag of the DFN group

N: size of the group

dfn_id: DFN id

Example:

```
# 3 DFN ids in this DFN group
of.DFN.group 'dfn1' 3 14 27 15
```

6.4.3 of.DFN.cohesive

of.DFN.cohesive *dfn_tag1* <string> *dfn_tag2* <string> *dfn_tag3* <string>

Set the DFN as initial cohesive by groups.

Parameters:

dfn_tag: tag of the group

Example:

```
of.DFN.cohesive 'dfn1' 'dfn2' 'dfn3'
```

6.4.4 of.DFN.broken

of.DFN.broken *dfn_tag1* <string> *dfn_tag2* <string> *dfn_tag3* <string>

Set the DFN as initial broken fractures by groups.

Parameters:

dfn_tag: tag of the DFN group

Example:

```
of.DFN.broken 'dfn1' 'dfn2' 'dfn3'
```

6.5 FEM

6.5.1 of.import.FEM.nodal.coord

```
of.import.FEM.nodal.coord *N* <int> *x0* <double> *y0* <double> *x1* <double> *y1* <double> ...
```

Import the nodal results after deformation by knowing node count and initial coordinates.

Parameters:

N: size of the group

x: x coordinates

y: x coordinates

Example:

```
of.import.FEM.nodal.coord 2 0.322 0.544 0.233 0.543
```

6.5.2 of.import.FEM.nodal.coord0

```
of.import.FEM.nodal.coord0 *N* <int> *x0* <double> *y0* <double> *x1* <double> *y1* <double> ...
```

Import the nodal results before deformation by knowing node count and initial coordinates.

Parameters:

N: size of the group

x: x coordinates

y: x coordinates

Example:

```
of.import.FEM.nodal.coord0 2 0.322 0.544 0.233 0.543
```

6.5.3 of.import.FEM.nodal.velocity

```
of.import.FEM.nodal.velocity *N* <int> *x0* <double> *y0* <double> *x1* <double> *y1* <double> ...
```

Import the nodal velocities by knowing node count and initial coordinates.

Parameters:

N: size of the group

x: x coordinates

y: y coordinates

Example:

```
of.import.FEM.nodal.velocity 2 0.322 0.544 0.233 0.543
```

6.5.4 of.import.FEM.nodal.groups

```
of.import.FEM.nodal.groups *group_tag* <string> *N* <int> *node_id1* <int> *node_id2* <int> *node_id3* <int> ...
```

Import the nodal groups.

Parameters:

N: size of the group

node_id: node ids

Example:

```
of.import.FEM.nodal.groups 'rock' 5 1 3 9 15 32
```

6.5.5 of.import.FEM.edge.groups

```
of.import.FEM.edge.groups *group_tag* <string> *N* <int> *node_id1* <int> *node_id2* <int> *node_id3* <int> ...
```

Import the edge groups.

Parameters:

N: size of the group

node_id: node ids

Example:

```
of.import.FEM.edge.groups 'rock' 5 1 3 9 15 32
```

6.5.6 of.import.FEM.element.groups

```
of.import.FEM.element.groups *element_tag* <string> *size* <int> *edge_id1* <int> *edge_id2* <int> *edge_id3* <int> ...
```

Import the element groups.

Parameters:

element_tag: tag of elements.

size: size of the groups.

edge_id: edge ids.

Example:

```
of.import.FEM.element.groups 'rock' 5 1 3 9 15 32
```

6.5.7 of.import.FEM.element.connectivity

```
of.import.FEM.element.connectivity *N* <int> *type* <int> *node_count* <int> *edge_id1* <int> *edge_id2* <int> *edge_id3* <int> ...
```

Import the element connectivity.

Parameters:

N: size of the group.

type: type of the element connectivity

| Type Number | Element Type | Number of Edges | Dimensions |
|-------------|--------------|-----------------|------------|
| 1 | LINE2 | 2 | 1 |
| 2 | TRIANGLE3 | 3 | 2 |
| 3 | QUADRANGLE4 | 4 | 2 |
| 4 | TRIANGLE6 | 3 | 2 |
| 5 | QUADRANGLE8 | 4 | 2 |
| 6 | QUADRANGLE9 | 4 | 2 |
| 7 | POLYGON | node_num | 2 |

node_count: number of nodes.

node_id: node ids.

Example:

```
of.import.FEM.element.connectivity 1 2 3 0 10 6
```

6.6 Geometry

6.6.1 of.geometry.square

```
of.geometry.square *geometry_tag* <string> *region* <double, double, double, double>
```

Create a geometry by assigning xmin, xmax, ymin and ymax of the box.

Parameters:

geometry_tag: tag of the geometry group

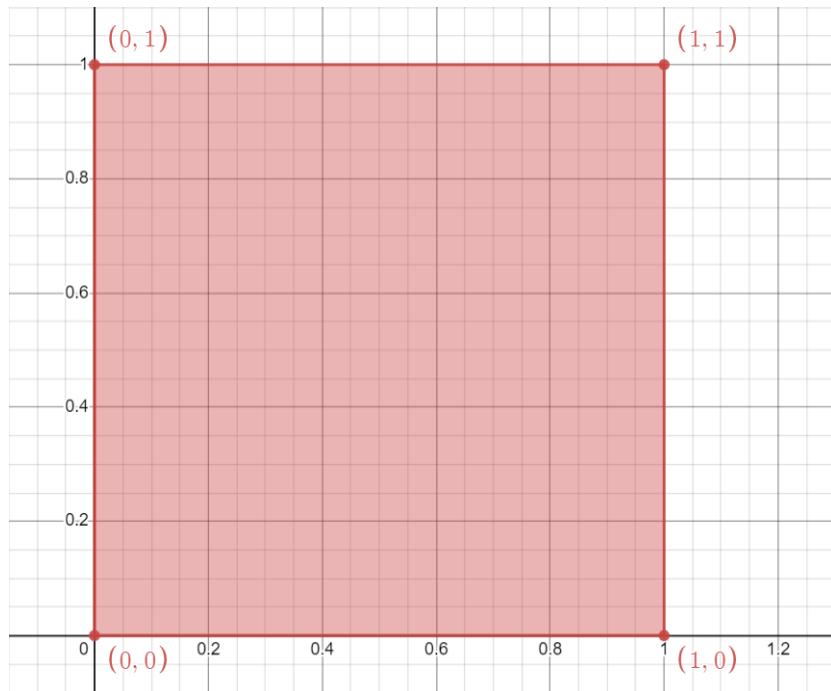
region: There are three methods to create a square:

1. [xmin, xmax, ymin, ymax]

2. x [xmin, xmax] y [ymin, ymax]
3. xmin value xmax value ymin value ymax value

Example:

```
# Method 1
of.geometry.square [0 1 0 1]
# Method 2
of.geometry.square [rock] x [0 1] y [0 1]
# Method 3
of.geometry.square [rock] xmin 0 xmax 1 ymin 0 ymax 1
```

**6.6.2 of.geometry.cut.square**

of.geometry.cut.square *new_geometry_tag* <string> *object_geometry_tag* <string> *region* <double, double, double>

Cut a geometry in existing geometry using the square method.

Note: There are three rules to follow:

- rule 1, the object should already exist.
- rule 2, the region is for the new geometry.
- rule 3, it just cut the object into several parts, but the fragments still have the object tag.

Parameters:

new_geometry_tag: tag of the geometry group cut from the original geometry.

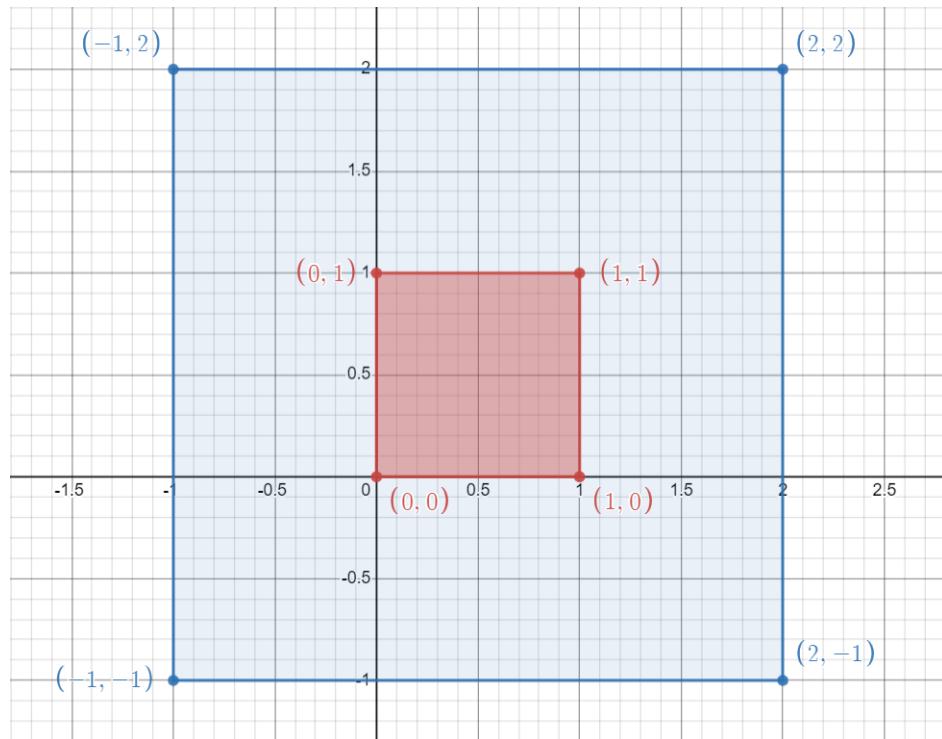
object_geometry_tag: tag of the original geometry group

region: There are three methods to define a square:

1. [xmin, xmax, ymin, ymax]
2. x [xmin, xmax] y [ymin, ymax]
3. xmin value xmax value ymin value ymax value

Example:

```
of.geometry.cut.square [tunnel] [rock] x [0 1] y [0 1]
```



6.6.3 of.geometry.remove.square

of.geometry.remove.square ***new_geometry_tag*** <string> ***object_geometry_tag*** <string> ***region*** <double, double, double, double>

Remove a geometry in existing geometry using the square method.

Note: There are three rules to follow:

- rule 1, the object should already exist.
- rule 2, the region is for the new geometry.
- rule 3, the new geometry will be removed. The tag is still reserved to mark the intersected edges between two objects.

Parameters:

new_geometry_tag: tag of the geometry group cut from the original geometry.

object_geometry_tag: tag of the original geometry group

region: There are three methods to define a square:

1. [xmin. xmax. ymin, ymax]
2. x [xmin, xmax] y [ymin, ymax]
3. xmin value xmax value ymin value ymax value

Example:

```
of.geometry.remove.square [tunnel] [rock] x [0 1] y [0 1]
```

6.6.4 of.geometry.circle

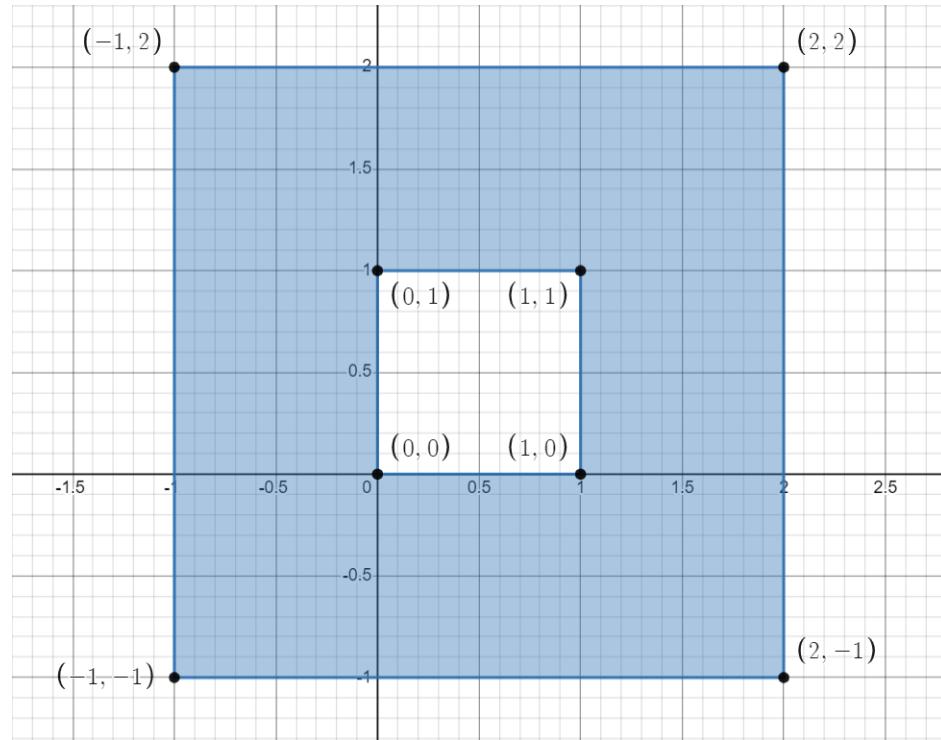
of.geometry.circle ***geometry_tag*** <string> ***region*** <double, double, double, int>

Create a geometry by assigning the x, y, radius and segments of the circle.

Parameters:

geometry_tag: tag of the geometry group

region: (default nseg 20) There are three methods to create a circle: 1. [x, y, rad, nseg] 2. center [x, y] radius nseg 3. x value y value rad value nseg value

**Example:**

```
of.geometry.circle [rock] center [0,0] rad 1.0 segments 40
```

6.6.5 of.geometry.cut.circle

of.geometry.cut.circle *new_geometry_tag* <string> *object_geometry_tag* <string> *region* <double, double, double, int>

Remove a geometry in existing geometry using the square method.

Note: There are three rules to follow:

- rule 1, the object should already exist.
- rule 2, the region is for the new geometry.
- rule 3, it only cuts the object into several parts, but the fragments still have the object tag nseg, which is 20 by default.

Parameters:

new_geometry_tag: tag of the geometry group cut from the original geometry.

object_geometry_tag: tag of the original geometry group

region: (default nseg 20) There are three methods to create a circle:

1. [x, y, rad, nseg]
2. center [x, y] radius nseg
3. x value y value rad value nseg value

Example:

```
of.geometry.cut.circle [tunnel] [rock] center [0,0] rad 1.0 segments 40
```

6.6.6 of.geometry.remove.circle

of.geometry.remove.circle ***new_geometry_tag*** <string> ***object_geometry_tag*** <string> ***region*** <double, double, double, int>

Remove the geometry in the object by assigning x, y, radius and segments of the circle.

Note: There are three rules to follow:

- rule 1, the object should already exist.
- rule 2, the region is for the new geometry.
- rule 3, the new object will be removed but the curves will be reserved to refine the mesh size.

Parameters:

new_geometry_tag: tag of the geometry group cut from the original geometry.

object_geometry_tag: tag of the original geometry group

region: (default nseg 20) There are three methods to create a circle:

1. [x, y, rad, nseg]
2. center [x, y] radius nseg
3. x value y value rad value nseg value

Example:

```
of.geometry.remove.circle [tunnel] [rock] center [0,0] rad 1.0 segments 40
```

6.6.7 of.geometry.ellipse

of.geometry.ellipse ***geometry_tag*** <string> ***region*** <double, double, double, double, double, int>

Create a geometry by assigning x, y, maxradius, minradius, theta and segments of the ellipse.

Parameters:

geometry_tag: tag of the geometry group

region: (default nseg 20) There are three methods to define a ellipse: 1. [x y minradius maxradius theta nseg] 2. center [x, y] radius [min max] theta nseg 3. x value, y value, rad value. minradius value, maxradius value, theta value, nseg value

Example:

```
of.geometry.ellipse [rock] center [0,0] rad [1.0, 20] theta 0 segments 40
```

6.6.8 of.geometry.cut.ellipse

of.geometry.cut.ellipse *new_geometry_tag* <string> *object_geometry_tag* <string> *region* <double, double, double, double, int>

Cut a geometry in the object by assigning x, y, maxradius, minradius, theta and segments of the ellipse.

Parameters:

new_geometry_tag: tag of the geometry group cut from the original geometry.

object_geometry_tag: tag of the original geometry group

region: (default nseg 20) There are three methods to define a ellipse:

1. [x y minradius maxradius theta nseg]
2. center [x, y] radius [min max] theta nseg
3. x value, y value, rad value. minradius value, maxradius value, theta value, nseg value

Example:

```
of.geometry.cut.ellipse [tunnel] [rock] center [0,0] rad [1.0, 20] theta 0 segments 40
```

6.6.9 of.geometry.remove.ellipse

of.geometry.remove.ellipse *new_geometry_tag* <string> *object_geometry_tag* <string> *region* <double, double, double, double, int>

Remove the geometry in the object by assigning the x, y, maxradius, minradius, theta and segments of the ellipse.

Parameters:

new_geometry_tag: tag of the geometry group cut from the original geometry.

object_geometry_tag: tag of the original geometry group

region: (default nseg 20) There are three methods to define a ellipse:

1. [x y minradius maxradius theta nseg]
2. center [x, y] radius [min max] theta nseg
3. x value, y value, rad value. minradius value, maxradius value, theta value, nseg value

Example:

```
of.geometry.remove.ellipse [tunnel] [rock] center [0,0] rad [1.0, 20] theta 0 segments 40
```

6.6.10 of.geometry.polygon

of.geometry.polygon *geometry_tag* <string> *region* <int, double, double, double, ...>

Create a geometry by assign the N <x> <y> <x> <y>... of the polygon. The polygon will be closed.

Parameters:

geometry_tag: tag of the geometry group

region: Define the points on the polygon: N, [x0,y0] [x1,y1], [x2,y2], [x3, y3]...

Example:

```
of.geometry.polygon [rock] 4 [-1 -1] [1,-1] [1,1] [-1,1]
```

6.6.11 of.geometry.cut.polygon

of.geometry.cut.polygon *new_geometry_tag* <string> *object_geometry_tag* <string> *region* <int, double, double, double, ...>

Cut a geometry in the object by assigning the N <x> <y> <x> <y>... of the polygon.

Parameters:

new_geometry_tag: tag of the geometry group cut from the original geometry.

object_geometry_tag: tag of the original geometry group

region: Define the points on the polygon: N, [x0,y0] [x1,y1], [x2,y2], [x3, y3]...

Example:

```
of.geometry.cut.polygon [tunnel] [rock] 4 [-1 -1] [1,-1] [1,1] [-1,1]
```

6.6.12 of.geometry.remove.polygon

of. geometry.remove.polygon *new_geometry_tag* <string> *object_geometry_tag* <string> *region* <int, double, double, double...>

Remove a geometry in the object by assigning the N <x> <y> <x> <y>... of the polygon.

Parameters:

new_geometry_tag: tag of the geometry group cut from the original geometry.

object_geometry_tag: tag of the original geometry group

region: Define the points on the polygon: N, [x0,y0] [x1,y1], [x2,y2], [x3, y3]...

Example:

```
of.geometry.remove.polygon [tunnel] [rock] 4 [-1 -1] [1,-1] [1,1] [-1,1]
```

6.6.13 of.geometry.table

of. geometry.table *geometry_tag* <string> *table_tage* <string>

Create a geometry by assign the N <x> <y> <x> <y>... of the polygon from a table, all the nodes in the table will be closed.

Parameters:

geometry_tag: tag of the geometry group.

region: Define the points on the polygon: N, [x0,y0] [x1,y1], [x2,y2], [x3, y3]... The polygon will be closed.

Example:

```
of.geometry.table [rock] 'moutain'
```

6.6.14 of.geometry.cut.table

of. geometry.cut.table *new_geometry_tag* <string> *object_geometry_tag* <string> *table_tage* <string>

Cut a geometry by assign the N <x> <y> <x> <y>... of the polygon from a table, the polygon will not be closed.

Parameters:

new_geometry_tag: tag of the geometry group cut from the original geometry.

object_geometry_tag: tag of the original geometry group

region: Define the points on the polygon: N, [x0,y0] [x1,y1], [x2,y2], [x3, y3]... The polygon will be closed.

Example:

```
of.geometry.cut.table [tunnel] [rock] 'moutain'
```

6.6.15 of.geometry.remove.table

of. geometry.remove.table *new_geometry_tag* <string> *object_geometry_tag* <string> *table_tage* <string>

remove a geometry by assign the N <x> <y> <x> <y>... of the polygon from a table, all the nodes in the table will be closed.

Parameters:

new_geometry_tag: tag of the geometry group cut from the original geometry.

object_geometry_tag: tag of the original geometry group

region: Define the points on the polygon: N, [x0,y0] [x1,y1], [x2,y2], [x3, y3]... The polygon will be closed.

Example:

```
of.geometry.remove.table [tunnel] [rock] 'mountain'
```

6.6.16 of.geometry.domain

of.geometry.domain region <double, double, double, double>

Create a geometry domain by assigning xmin, xmax, ymin and ymax of the box.

Parameters:

region: geometry domain for creating the joints. There are three methods to define the domain: 1. [xmin, xmax, ymin, ymax] 2. x [xmin, xmax] y [ymin, ymax] 3. xmin value xmax value ymin value ymax value

Example:

```
of.geometry.domain x [0 1] y [0 1]
```

6.6.17 of.geometry.cut.joint

of.geometry.cut.joint *new_geometry_tag* <string> *object_geometry_tag* <string> *region* <double, double, double...>

Cut a joint in an object by assign the <x> <y> <x> <y>...

Parameters:

new_geometry_tag: tag of the geometry group cut from the original geometry.

object_geometry_tag: tag of the original geometry group

region: Define the points on the joints: [x0,y0] [x1,y1], [x2,y2], [x3, y3]...

Example:

```
of.geometry.cut.joint 'joint' 'tock' [0,1] [2,3]
```

6.6.18 of.geometry.cut.jset

```
of. geometry.cut.jset *geometry_tag* <string> *object_geometry_tag* <string> *dip_method* <double, double> *space_method* <double, double> *trace_method* <double, double> *gap_method* <double, double> *start_point* <double, double>
```

Cut a joint in an object by assign the <x> <y> <x> <y>...

Parameters:

geometry_tag: tag of the joint set group.

object_geometry_tag: tag of the original geometry group

dip_method: There are three methods to define dip: 1. n_dip [mean, dev] 2. u_dip [min, max] 3. dip constantAngle;

space_method: There are two methods to define space: 1. space constantSpace 2. n_space [mean, dev]

trace_method: There are two methods to define trace: 1. trace constanttrace 2. n_trace [mean, dev]

gap_method: There are two methods to define gap: 1. gap constantgap 2. n_gap [mean, dev]

start_point: to define the start points by <x><y>

Note:

- rule 1, if you want to create persistent joint sets, only use dip, space and start point. If you want to create nonpersistent joint sets, use dip, space, trace, gap and start point together.
- rule 2, for the dip angle ranging [0-90]. Start point should be located on the left-up corner. When the range is [90-180], it should be located at the down-left corner.
- rule 3, you should set the domain first before you create jsets and dfns.
- rule 4, recommend you preset the minangle and minszie to delete bad segments in case of causing bad mesh.
- rule 5, start point is not mandatory, but it will help you locate the joint sets at the right places if you use that.
- rule 6, the start points must be located within (not on) the domain.

Example:

```
of.geometry.cut.jset [jset1] [rock] n_dip [60, 10] n_space [3,1] start [-10 10]
```

6.6.19 of.geometry.cut.DFN

of.geometry.cut.DFN ***new_geometry_tag*** <string> ***object_geometry_tag*** <string> ***dip_method*** <double, double> ***length_method*** <double, double> ***threshold_method*** <int>

The joint may be deleted if you assign wrong minangle or wrong minsize Cut DFN sets by assigning the dip, length, threshold method of the joints.

Note: There are three rules to follow:

- rule 1, you should set the domain firstly when you create jsets and dfns.
 - rule 2, recommend you preset the minangle and minsize to delete bad segments in case of causing bad mesh.
 - rule 3, change the iteration if the generation is not converged.
-

Parameters:

new_geometry_tag: tag of the geometry group cut from the original geometry.

object_geometry_tag: tag of the original geometry group

dip_method: There are three methods to define dip:

1. n_dip [mean, dev]
2. u_dip [min, max]
3. dip constantAngle;

length_method: There are four methods to define length:

1. length constantlength
2. n_length [mean, dev]
3. f_length [fisher1,fisher2]
4. p_length [poer1, power2]

threshold_method: There are three methods to define threshold:

1. Count
2. p10
3. p21

Example:

```
of.geometry.cut.dfn [dfn1] [rock] u_dip [20, 60] n_length [4,2] count 200
```

6.6.20 of.geometry.cut.arc

of.geometry.cut.arc ***new_geometry_tag*** <string> ***object_geometry_tag*** <string> ***region*** <double, double, double, double, int>

Create an arc geometry in object geometry by assign the center, start angle, end angle, radius and number of segments.

Parameters:

new_geometry_tag: tag of the geometry group cut from the original geometry.

object_geometry_tag: tag of the original geometry group

region: (default nseg 20) There are three methods to create a circle:

1. [x, y, rad, nseg]
2. center [x, y] radius nseg
3. x value y value rad value nseg value

Example:

```
of.geometry.cut.arc [tunnel_edge] [rock] center [0,0] radius 1.0 theta [45, 90] segments ↵
← 20
```

6.6.21 of.geometry.minsize

of.geometry.minsize ***minsize*** <Real>

Set the minsize to delete the bad segments for jsets and dfns.

Parameters:

minsize: Bad segments under this threshold will be deleted.

Example:

```
of.geometry.minsize 1e-3
```

6.6.22 of.geometry.minangle

of.geometry.minangle *minangle* <Real>

Set the minangle to delete the bad segments for jsets and dfns.

Parameters:

minangle: Bad segments under this threshold will be deleted.

Example:

```
of.geometry.minangle 20
```

6.6.23 of.geometry.iteration

of.geometry.iteration *iteration* <int>

Set the iteration to converge the generation of jsets and dfns.

Parameters:

iteration: (default 1000) number of iteration to run to converge the generation of jsets and dfns.

Example:

```
of.geometry.iteration 100
```

6.6.24 of.geometry.group

of.geometry.group *geometry_tag* <string> *method* <string> *region* <double, double, ...> ...

Set group of the points in gmsh to set the new mesh size, the method includes.

Parameters:

geometry_tag: tag of the geometry group

method:

-box.in/box.on/box.out

-circle.in/circle.on/circle.out

-plane.on/plane.above/plane.below

region:

The region for box can be set as

- x [xmin xmax] y [ymin ymax] or
- [xmin xmax ymin ymax]

The region for circle can be set as

- center [x,y] radius or
- [x,y,radius]

The region for plane can be set as

- p1 [x0,y0] p2 [x1,y1] or
- [x0,y0,x1,y1]

Note:

- rule 1, the group of geometry is used for point only,
- rule 2, the previous lines or surfaces also are inherited to groups
- rule 3, the group information is updated after the entity fragmentation

Example:

```
of.geometry.group [refine] range box.in x [0,1] y [0,1]
```

6.6.25 of.geometry.mesh.size

of.geometry.mesh.size *group_tag* <string> *mesh_size* <Real>

Set the mesh size to the tags. If use default means the global mesh size, you can also set a different value for the zone you want to refine.

Parameters:

geometry_tag: tag of the geometry group

mesh_size: the mesh size for global or local area.

Example:

```
# Global Mesh Size
of.geometry.mesh.size [default] 0.03
# Mesh Size at the User Defined Area
of.geometry.mesh.size [refineLine] 0.001
```

6.6.26 of.geometry.recombine

of.geometry.recombine *surface_tag* <string>

Generate quadrangle element.

Parameters:

surface_tag: generate quadrangle element for the given tag group.

Example:

```
of.geometry.recombine [rock]
```

6.6.27 of.geometry.mesh

of.geometry.mesh *mesh_algorithm* <string> *order* <int>

Set the method to create mesh.

Parameters:

mesh_algorithm: (default delaunay) delaunay, meshadapt, frontal-delaunay

order: (default 1) The order can be set to 2 or 3 to generate higher order elements.

Example:

```
# Default
of.geometry.mesh delaunay
# Alternative choice
of.geometry.mesh meshadapt order 2
```

6.6.28 of.geometry.mesh.write

of.geometry.mesh.write *mesh_file_path* <string>

Export the mesh file. You can also directly export in the Gmsh interface.

Parameters:

mesh_file_path: The path to save the mesh.

Example:

```
of.geometry.mesh.write [rock.msh]
```

6.7 Group

6.7.1 of.group.nodal

of.group.nodal *group_tag* <string> *method* <string> *region* <double, double, ...> ...

Parameters:

group_tag: tag of the group.

method:

-box.in/box.on/box.out

-circle.in/circle.on/circle.out

-plane.on/plane.above/plane.below

region:

The region for box can be set as

- x [xmin xmax] y [ymin ymax] or
- [xmin xmax ymin ymax]

The region for circle can be set as

- center [x,y] radius or
- [x,y,radius]

The region for plane can be set as

- p1 [x0,y0] p2 [x1,y1] or
- [x0,y0,x1,y1]

Note:

- rule 1: it is illegal to use the same tags for one type of entity.
- rule 2: the node group also be inherited from your input mesh file.
- rule 3: the node groups can be done before the mesh processing. The group information will be transferred to the new inserted nodes automatically.

Example:

```
of.group.nodal [rock] range box.in x [0,1] y [0,1]
```

6.7.2 of.group.edge

of.group.edge *group_tag* <string> *method* <string> *region* <double, double,...> ...

Parameters:

group_tag: tag of the group.

method:

-box.in/box.on/box.out

-circle.in/circle.on/circle.out

-plane.on/plane.above/plane.below

region:

The region for box can be set as

- x [xmin xmax] y [ymin ymax] or
- [xmin xmax ymin ymax]

The region for circle can be set as

- center [x,y] radius or
- [x,y,radius]

The region for plane can be set as

- p1 [x0,y0] p2 [x1,y1] or
- [x0,y0,x1,y1]

Note:

- rule 1: it is illegal to use the same tags for one type of entity.
 - rule 2: the node group also be inherited from your input mesh file.
 - rule 3: the node groups can be done before the mesh processing. The group information will be transferred to the new inserted nodes automatically.
-

Example:

```
of.group.edge [rock] range box.in x [0,1] y [0,1]
```

6.7.3 of.group.element

of.group.element *group_tag* <string> *method* <string> *region* <double, double, ...> ...

Parameters:

group_tag: tag of the group.

method:

- box.in/box.on/box.out

- circle.in/circle.on/circle.out

- plane.on/plane.above/plane.below

region:

The region for box can be set as

- x [xmin xmax] y [ymin ymax] or
- [xmin xmax ymin ymax]

The region for circle can be set as

- center [x,y] radius or
- [x,y,radius]

The region for plane can be set as

- p1 [x0,y0] p2 [x1,y1] or
- [x0,y0,x1,y1]

Note:

- rule 1: it is illegal to use the same tags for one type of entity.
- rule 2: the node group also be inherited from your input mesh file.
- rule 3: the node groups can be done before the mesh processing. The group information will be transferred to the new inserted nodes automatically.

Example:

```
of.group.element [rock] range box.in x [0,1] y [0,1]
```

6.7.4 of.group.cohelement

of.group.cohelement *group_tag* <string> *method* <string> *region* <double, double, ...> ...

Parameters:

group_tag: tag of the group.

method:

-box.in/box.on/box.out

-circle.in/circle.on/circle.out

-plane.on/plane.above/plane.below

region:

The region for box can be set as

- x [xmin xmax] y [ymin ymax] or
- [xmin xmax ymin ymax]

The region for circle can be set as

- center [x,y] radius or
- [x,y,radius]

The region for plane can be set as

- p1 [x0,y0] p2 [x1,y1] or
- [x0,y0,x1,y1]

Note:

- rule 1: it is illegal to use the same tags for one type of entity.
 - rule 2: the node group also be inherited from your input mesh file.
 - rule 3: the node groups can be done before the mesh processing. The group information will be transferred to the new inserted nodes automatically.
-

Example:

```
of.group.cohelement [rock] range box.in x [0,1] y [0,1]
```

6.7.5 of.group.nodal.from.element

of.group.nodal.from.element *group_tag* <string> *element_group_tag* <string>

Inherit the nodal group from their parent elements. All the nodals of the targeted elements will be grouped as the new nodal groups.

Parameters:

group_tag: tag of the group.

element_group_tag: element group tag.

Example:

```
of.group.nodal.from.element [rocknodal] [rockelement]
```

6.7.6 of.group.edge.from.element

of.group.edge.from.element *group_tag* <string> *element_group_tag* <string>

Inherit the edge group from their parent elements. All the edges of the targeted elements will be grouped as the new edge groups.

Parameters:

group_tag: tag of the group.

element_group_tag: element group tag.

Example:

```
of.group.edge.from.element [rockedge] [rockelement]
```

6.7.7 of.group.edge.from.cohelement

of.group.edge.from.cohelement *group_tag* <string> *cohelement_group_tag* <string>

Inherit the edge group from their parent cohelements. All the edges of the targeted cohelements will be group as the new edge groups.

Parameters:

group_tag: tag of the group.

cohelement_group_tag: cohelement group tag.

Example:

```
of.group.edge.from.cohelement [rockedge] [rockcohelement]
```

6.7.8 of.group.edge.from.dfn

of.group.edge.from.dfn *group_tag* <string> *dfn_group_tag* <string>

Inherit the edge group from their parent dfns. All the edges of the targeted dfns will be group as the new edge groups.

Parameters:

group_tag: tag of the group.

dfn_group_tag: dfn group tag.

Example:

```
of.group.edge.from.dfn [rockedge] [dfn2]
```

6.7.9 of.group.cohelement.from.dfn

of.group.cohelement.from.dfn *group_tag* <string> *dfn_group_tag* <string>

Inherit the cohelement group from their parent dfns. All the cohelements of the targeted dfns will be group as the new cohelement groups.

Parameters:

group_tag: tag of the group.

dfn_group_tag: dfn group tag.

Example:

```
of.group.cohelement.from.dfn [rockcohelement] [dfn2]
```

6.7.10 of.group.cohelement.from.gbm

of.group.cohelement.from.gbm *group_tag* <string> *element_group_tag1* <string> *element_group_tag2* <string>

Create the cohesive element groups if they are located between the target element tags.

Parameters:

group_tag: tag of the group.

element_group_tag: element group tag.

Example:

```
of.group.cohelement.from.gbm [Quartz-mica] [Quartz] [Mica]
```

6.7.11 of.group.nodal.bool.union

```
of.group.nodal.bool.union *group_tag* <string> tag1 <string> tag2 <string> ...
```

Get the bool union of the second - last nodal groups and store them into the first tags.

Parameters:

group_tag: tag of the group.

tag: nodal groups to be operated.

Example:

```
of.group.nodal.bool.union [rock] [rock_up] [rock_down] [rock_left] [rock_right]
```

6.7.12 of.group.nodal.bool.intersect

```
of.group.nodal.bool.intersect *group_tag* <string> tag1 <string> tag2 <string> ...
```

Get the bool intersect of the second - last nodal groups and store them into the first tags.

Parameters:

group_tag: tag of the group.

tag: nodal groups to be operated.

Example:

```
of.group.nodal.bool.intersect [rock] [rock_up] [rock_down] [rock_left] [rock_right]
```

6.7.13 of.group.nodal.bool.subtract

of.group.nodal.bool.subtract *group_tag* <string> tag1 <string> tag2 <string> ...

Get the bool subtract of the second - last nodal groups and store them into the first tags.

Parameters:

group_tag: tag of the group.

tag: nodal groups to be operated.

Example:

```
of.group.nodal.bool.subtract [rock] [rock_up] [rock_down] [rock_left] [rock_right]
```

6.7.14 of.group.edge.bool.union

of.group.edge.bool.union *group_tag* <string> tag1 <string> tag2 <string> ...

Get the bool union of the second - last edge groups and store them into the first tags.

Parameters:

group_tag: tag of the group.

tag: edge groups to be operated.

Example:

```
of.group.edge.bool.union [rock] [rock_up] [rock_down] [rock_left] [rock_right]
```

6.7.15 of.group.edge.bool.intersect

of.group.edge.bool.intersect *group_tag* <string> tag1 <string> tag2 <string> ...

Get the bool intersect of the second - last edge groups and store them into the first tags.

Parameters:

group_tag: tag of the group.

tag: edge groups to be operated.

Example:

```
of.group.edge.bool.intersect [rock] [rock_up] [rock_down] [rock_left] [rock_right]
```

6.7.16 of.group.edge.bool.subtract

of.group.edge.bool.subtract *group_tag* <string> tag1 <string> tag2 <string> ...

Get the bool subtract of the second - last edge groups and store them into the first tags.

Parameters:

group_tag: tag of the group.

tag: edge groups to be operated.

Example:

```
of.group.edge.bool.subtract [rock] [rock_up] [rock_down] [rock_left] [rock_right]
```

6.7.17 of.group.element.bool.union

of.group.element.bool.union *group_tag* <string> tag1 <string> tag2 <string> ...

Get the bool union of the second - last element groups and store them into the first tags.

Parameters:

group_tag: tag of the group.

tag: element groups to be operated.

Example:

```
of.group.element.bool.union [rock] [rock_up] [rock_down] [rock_left] [rock_right]
```

6.7.18 of.group.element.bool.intersect

```
of.group.element.bool.intersect *group_tag* <string> tag1 <string> tag2 <string> ...
```

Get the bool intersect of the second - last element groups and store them into the first tags.

Parameters:

group_tag: tag of the group.

tag: element groups to be operated.

Example:

```
of.group.nodal.bool.intersect [rock] [rock_up] [rock_down] [rock_left] [rock_right]
```

6.7.19 of.group.element.bool.subtract

```
of.group.element.bool.subtract *group_tag* <string> tag1 <string> tag2 <string> ...
```

Get the bool subtract of the second - last element groups and store them into the first tags.

Parameters:

group_tag: tag of the group.

tag: element groups to be operated.

Example:

```
of.group.element.bool.subtract [rock] [rock_up] [rock_down] [rock_left] [rock_right]
```

6.7.20 of.group.cohelement.bool.union

of.group.cohelement.bool.union *group_tag* <string> tag1 <string> tag2 <string> ...

Get the bool union of the second - last cohesive element groups and store them into the first tags.

Parameters:

group_tag: tag of the group.

tag: cohesive element groups to be operated.

Example:

```
of.group.cohelement.bool.union [rock] [rock_up] [rock_down] [rock_left] [rock_right]
```

6.7.21 of.group.cohelement.bool.intersect

of.group.cohelement.bool.intersect *group_tag* <string> tag1 <string> tag2 <string> ...

Get the bool intersect of the second - last cohesive element groups and store them into the first tags.

Parameters:

group_tag: tag of the group.

tag: cohesive element groups to be operated.

Example:

```
of.group.cohelement.bool.intersect [rock] [rock_up] [rock_down] [rock_left] [rock_right]
```

6.7.22 of.group.cohelement.bool.subtract

of.group.cohelement.bool.subtract *group_tag* <string> tag1 <string> tag2 <string> ...

Get the bool subtract of the second - last cohesive element groups and store them into the first tags.

Parameters:

group_tag: tag of the group.
tag: cohesive element groups to be operated.

Example:

```
of.group.coheelement.bool.subtract [rock] [rock_up] [rock_down] [rock_left] [rock_right]
```

6.8 Mesh

6.8.1 of.mesh.insert

of.mesh.insert *element_tag* <string>

Insert cohesive in the target element groups. OpenFDEM support partially insert CZMs.

Parameters:

element_tag: tag of the elements.

Example:

```
of.mesh.insert [default]
of.mesh.insert [rock]
```

6.8.2 of.mesh.insert.dfn

of.mesh.insert.dfn *dfn_tag* <string>

Insert cohesive only on a dfn plane, the dfn should cross the block. OpenFDEM supports inserting cohesive elements on interface.

Parameters:

dfn_tag: tag of the DFNs.

Example:

```
of.mesh.insert.dfn [DFN1]
```

6.8.3 of.mesh.voronoi

of.blast.position *element_tag* <string>

Change the delaunay element to Voronoi element. All the group information will be cleared after this mesh processing. You should regroup all the entities. This command only supports entire mesh processing.

Parameters:

element: tag of the elements.

Example:

```
of.mesh.voronoi [rock]
```

6.8.4 of.mesh.split

of.mesh.split *element_tag* <string>

Insert contacts within the element tags and no cohesive elements will be created. The target element group will be changed to discrete bodies.

Parameters:

element_tag: tag of the elements.

Example:

```
of.mesh.split [rock]
```

6.8.5 of.mesh.bary

of.mesh.bary *element_tag* <string>

Refine the target element groups using bary separation scheme.

Parameters:

element_tag: tag of the elements.

Example:

```
of.mesh.bary [rock]
```

6.8.6 of.mesh.RG

of.mesh.RG *element_tag* <string>

Refine the target element groups using RG separation scheme.

Parameters:

element_tag: tag of the elements.

Example:

```
of.mesh.RG [rock]
```

6.8.7 of.mesh.RGB

of.mesh.RGB *element_tag* <string>

Refine the target element groups using RGB separation scheme.

Parameters:

element_tag: tag of the elements.

Example:

```
of.mesh.RGB [rock]
```

6.8.8 of.mesh.NVB

of.mesh.NVM *element_tag* <string>

Refine the target element groups using NVB separation scheme.

Parameters:

element_tag: tag of the elements.

Example:

```
of.mesh.NVB [rock]
```

6.8.9 of.mesh.baryGB

of.mesh.baryGB *element_tag* <string>

Refine the target element groups using baryGB separation scheme.

Parameters:

element_tag: tag of the elements.

Example:

```
of.mesh.baryGB [rock]
```

6.9 Material

6.9.1 of.mat.element

**of.mat.element *element_tag* <string> *material_type* <string> parameter_keyword1 <string>
parameter_value1 <double> ...**

Assign the material parameters to the target element groups. The details of materials in OpenFDEM can be found in the table below.

| Material type | Constitutive keyword | Parameter keywords | Units |
|----------------|----------------------|---|-------|
| Hooken elastic | <i>elastic</i> | <i>density</i> - density of the mat <i>v</i> - Poisson's ratio <i>E</i> - young modulus <i>K</i> - bulk modulus (E-v or K-G you can choose one) <i>G</i> - shear modulus <i>table</i> - time dependent for all kinds of modules, including E, K, G, lambda <i>damp</i> - stiffness damp ratio, 0.0 by default The following parameters for phasefield: <i>Gc</i> - energy release rate <i>lc</i> - characteristic length <i>nita</i> - phase viscous factor | |
| Neo-hooken | <i>neo</i> | <i>density</i> - density of the mat <i>v</i> - Poisson's ratio <i>E</i> - young modulus <i>K</i> - bulk modulus (E-v or K-G you can choose one) | |

continues on next page

Table 1 – continued from previous page

| Material type | Constitutive keyword | Parameter keywords | Units |
|------------------------|----------------------|--|-------|
| Excavation mat | excavation | G - shear modulus <i>table</i> - time dependent for all kinds of modules, including E, K, G, lambda <i>damp</i> - stiffness damp ratio, 0.0 by default <i>density</i> - density of the mat <i>v</i> - Poisson's ratio <i>E</i> - young modulus <i>K</i> - bulk modulus (E-v or K-G you can choose one) <i>G</i> - shear modulus <i>table</i> - time dependent for all kinds of modules, including E, K, G, lambda <i>damp</i> - stiffness damp ratio, 0.0 by default | |
| Rigid bodies | <i>Rigid</i> | <i>density</i> - density of the mat | |
| Mohr-Coulomb Plastic | MC | <i>density</i> - density of the mat <i>v</i> - Poisson's ratio <i>E</i> - young modulus <i>K</i> - bulk modulus (E-v or K-G you can choose one) <i>G</i> - shear modulus <i>tension</i> - tension strength, cut-off Mohor-Coulomb model <i>cohesion</i> - cohesive strength, c <i>dilation</i> - dilation angle, non associated flow, equal to friction by default <i>friction</i> - friction angle <i>table</i> - time dependent for all kinds of modules, including E, K, G, lambda <i>damp</i> - stiffness damp ratio, 0.0 by default | |
| Drucker-Prager plastic | DP | <i>density</i> - density of the mat <i>v</i> - Poisson's ratio <i>E</i> - young modulus <i>K</i> - bulk modulus (E-v or K-G you can choose one) <i>G</i> - shear modulus <i>tension</i> - tension strength, cut-off Mohor-Coulomb model <i>kshear</i> - cohesive strength, c <i>qdilation</i> - dilation angle, non associated flow, equal to friction by default <i>qfriction</i> - friction angle <i>damp</i> - stiffness damp ratio, 0.0 by default | |
| Transverse Mat | Transverse | <i>density</i> - density of the mat <i>v_xy</i> - Poisson's ratio in x-y direction <i>v_yx</i> - Poisson's ratio in y-x direction <i>E_x</i> - young modulus in x <i>E_y</i> - young modulus in y <i>G</i> - shear modulus <i>damp</i> - stiffness damp ratio, 0.0 by default | |
| Explosive EOS | JWL | <i>density</i> - density of the explosive <i>E0</i> - chemical energy of the explosive <i>detonation</i> - detonation of the explosive <i>A</i> - parameter A <i>B</i> - parameter B <i>R1</i> - parameter R1 <i>R2</i> - parameter R2 <i>omega</i> - parameter R1 <i>x</i> - firestarter position, x <i>y</i> - firestarter position, y <i>t0</i> - fire starting time | |

continues on next page

Table 1 – continued from previous page

| Material type | Constitutive keyword | Parameter keywords | Units |
|------------------------|----------------------|--|-------|
| Mazars concrete damage | <i>Mazars</i> | <i>density</i> - density of the mat <i>v</i> - Poisson's ratio <i>E</i> - young modulus <i>At</i> - tension parameter <i>Ac</i> - compression parameter <i>Bt</i> - tension factor <i>Bc</i> - compression factor <i>e0</i> - <i>damp</i> - stiffness damp ratio, 0.0 by default | |
| Johns on-Holmquist-II | <i>JH2</i> | <i>density</i> - density <i>v</i> - Poisson's ratio <i>E</i> - young modulus <i>A</i> - parameter A <i>B</i> - parameter B <i>C</i> - parameter C <i>M</i> - parameter M <i>N</i> - parameter N <i>D1</i> - parameter D1 <i>D2</i> - parameter D2 <i>T</i> - parameter T <i>HEL</i> - parameter HEL <i>PHEL</i> - parameter PHEL <i>SHEL</i> - parameter SHEL <i>K1</i> - parameter K1 <i>K2</i> - parameter K2 <i>K3</i> - parameter K3 <i>damp</i> - stiffness damp ratio, 0.0 by default | |
| Burger creep | <i>Burgers</i> | To be added in the tutorial, if you want to use this model, contact the developer | |
| Power creep law | <i>Power</i> | To be added in the tutorial, if you want to use this model, contact the developer | |
| Johns on-Holmquist-II | <i>JH2</i> | To be added in the tutorial, if you want to use this model, contact the developer | |
| Yang blast model | <i>Yang</i> | To be added in the tutorial, if you want to use this model, contact the developer | |

Example:

```
of.mat.element 'default' elastic den 2700 E 30e9 v 0.3 damp 2.0
```

6.9.2 of.mat.cohesive

```
of.mat.cohesive *element_tag* <string> *material_type* <string> parameter_keyword1 <string>
parameter_value1 <double> ...
```

Assign the material parameters to the target cohesive element groups. The details of materials in OpenFDEM can be found in the table below.

| Material type | Constitutive keyword | Parameter keywords | Units |
|---|----------------------|--|-------|
| Evans_Marathe cohesive law | <i>EM</i> | <i>tension</i> - tension strength <i>cohesion</i> - cohesion strength <i>friction</i> - friction pn - normal stiffness, will be assigned by default pt - tangential stiffness, will be assigned by default GI - energy release rate in normal GII - energy release rate in tangential beta-I - viscous of CZM in normal beta-II - viscous of CZM in tangential We also have a table to each variables if you want the <i>ttable</i> - table for tension <i>ctable</i> - table for cohesion <i>ftable</i> - table for friction <i>Gitable</i> - table for GI <i>Gitable</i> - table for GII | |
| Strain rate Evans_Marathe cohesive law | <i>EM_dyn</i> | <i>tension</i> - tension strength <i>cohesion</i> - cohesion strength <i>friction</i> - friction pn - normal stiffness, will be assigned by default pt - tangential stiffness, will be assigned by default GI - energy release rate in normal GII - energy release rate in tangential beta-I - viscous of CZM in normal beta-II - viscous of CZM in tangential We also have a table to each variables if you want the <i>ttable</i> - table for tension <i>ctable</i> - table for cohesion <i>ftable</i> - table for friction <i>Gitable</i> - table for GI <i>Gitable</i> - table for GII <i>n_rate</i> - critical strain rate in normal <i>n_pow</i> - dynamic increase factor in normal <i>s_rate</i> - critical strain rate in tangential <i>s_pow</i> - dynamic increase factor in tangential | |
| Ortiz and Pandolfi exponential Cohesive Law | <i>OP</i> | <i>tension</i> - tension strength <i>cohesion</i> - cohesion strength | |
| Linear Cohesive Law | <i>LINEAR</i> | <i>tension</i> - tension strength | |

continues on next page

Table 2 – continued from previous page

| Material type | Constitutive keyword | Parameter keywords | Units |
|--|----------------------|---|--|
| | | <i>cohesion</i> - cohesion strength | |
| | | <i>friction</i> - friction | |
| | | <i>pn</i> - normal stiffness, will be assigned by default | |
| | | <i>pt</i> - tangential stiffness, will be assigned by default | |
| | | <i>GI</i> - energy release rate in normal | |
| | | <i>GII</i> - energy release rate in tangential | |
| | | We also have a table to each variables if you want the | |
| | | <i>ttable</i> - table for tension | |
| | | <i>ctable</i> - table for cohesion | |
| | | <i>ftable</i> - table for friction | |
| | | <i>GItable</i> - table for GI | |
| | | <i>GIItable</i> - table for GII | |
| Heterogeneous Evans_Marathe cohesive law | <i>EM_het</i> | | To be added in the tutorial, if you want to use this model |
| Anistropic Evans_Marathe cohesive law | <i>EM_ani</i> | | To be added in the tutorial, if you want to use this model |

Example:

```
of.mat.cohesive 'default' EM ten 1e6 coh 3e6 fric 0.3 GI 10 GII 50
```

6.9.3 of.mat.contact

of.mat.contact *element_tag* <string> *material_type* <string> parameter_keyword1 <string>
parameter_value1 <double> ...

Assign the material parameters to the target element group pairs. The details of materials in OpenFDEM can be found in the table below.

| Material type | Constitutive keyword | Parameter keywords | Units |
|--|----------------------|---|-------|
| Mohr-Coulomb Slip | <i>MC</i> | <i>friction</i> - friction <i>kn</i> - normal stiffness, will be assigned by default | Pa |
| | | $pn = \lambda(K_1 + K_2)/2$ | |
| | | <i>ks</i> - tangential stiffness, will be assigned by default | Pa |
| | | $pn = \lambda(G_1 + G_2)/2$ | |
| Rate dependent Coulomb Slip | <i>rMC</i> | To be added in the tutorial, if you want to use this model, contact the developer | |
| Giovanni Grasselli rough shear contact | <i>roughshear</i> | <i>friction</i> - friction | |
| | | <i>kn</i> - normal stiffness, will be assigned by default | Pa |
| | | $pn = \lambda(K_1 + K_2)/2$ | |
| | | <i>ks</i> - tangential stiffness, will be assigned by default | Pa |
| | | $pn = \lambda(G_1 + G_2)/2$ | |
| | | theta -parameter | |
| | | C -parameter | |
| | | A0 - parameter | |
| | | B -parameter | |
| | | <i>tension</i> - tension strength | |
| Dynamic Mohr-Coulomb Slip | <i>dMC</i> | <i>s_friction</i> - static friction | |
| | | <i>d_friction</i> - dynamic friction | |
| | | <i>slip_rate</i> - slip rate | /s |
| | | <i>kn</i> - normal stiffness, will be assigned by default | Pa |
| | | $pn = \lambda(K_1 + K_2)/2$ | |
| | | <i>ks</i> - tangential stiffness, will be assigned by default | Pa |
| | | $pn = \lambda(G_1 + G_2)/2$ | |
| Hertz contact model | <i>HERTZ</i> | To be added in the tutorial, if you want to use this model, contact the developer | |
| Friction | <i>FRICTION</i> | To be added in the tutorial, if you want to use this model, contact the developer | |

Example:

```
of.mat.contact 'default' MC 0.3
of.mat.contact 'up' 'rock' MC 0.0
```

6.9.4 of.mat.hydro.matrix

To be added in the tutorial soon. If you want to use this model, please contact the developer.

6.9.5 of.mat.hydro.fracture

To be added in the tutorial soon. If you want to use this model, please contact the developer.

6.9.6 of.mat.mpm.fluid

To be added in the tutorial soon. If you want to use this model, please contact the developer.

6.9.7 of.mat.mpm.solid

To be added in the tutorial soon. If you want to use this model, please contact the developer.

6.9.8 of.mat.hydro.fluid

To be added in the tutorial soon. If you want to use this model, please contact the developer.

6.9.9 of.mat.hydro.gas

To be added in the tutorial soon. If you want to use this model, please contact the developer.

6.9.10 of.mat.nonlocal

To be added in the tutorial soon. If you want to use this model, please contact the developer.

6.9.11 of.mat.thermal

To be added in the tutorial soon. If you want to use this model, please contact the developer.

6.10 History

6.10.1 of.history.pv.interval

of.history.pv.interval interval <int>

Set the result writing intervals.

Parameters:

global: (default 1000) output intervals

Example:

```
of.history.pv.interval 20000
```

6.10.2 of.history.pv.dynamic.interval

of.history.pv.dynamic.interval global <int> threshold <int> reduce <int>

Set the dynamic result writing intervals by the increasing broken CZM.

Parameters:

global: (default 1000) output intervals.

threshold: threshold for the minimum interval.

reduce: step of reducing.

Example:

```
of.history.pv.dynamic.interval global 20000 threshold 100 reduce 200
of.history.pv.dynamic.interval [20000 100 200]
```

6.10.3 of.history.pv.field

of.history.pv.field *tag1* <string> *tag2* <string> *tag3* <string> ...

Set the items you want to write in ParaView results (See the list below).

Field keywords:

- default (Output all)
- velocity
- force
- displacement

- fluid_pressure – only for hydro
- nodal_group
- element_group
- gbm_group
- mass
- stress
- strain
- strain_rate
- principal_stress
- mat_id
- fragment
- temperature – only for thermal
- thermal-flux – only for thermal
- t0 – only for thermal

Example:

```
of.history.pv.field 'default'  
of.history.pv.field 'velocity' 'force' 'displacement'
```

6.10.4 of.history.pv.fracture

of.history.pv.fracture *tag1* <string> *tag2* <string> *tag3* <string> ...

Set the items you want to write in ParaView results (See the list below).

Field keywords:

- default (Output all)
- mode
- time
- win_time
- win_kinetic
- kinetic
- magnitude
- energy

Example:

```
of.history.pv.fracture default
of.history.pv.fracture 'mode' 'time' 'magnitude'
```

6.10.5 of.history.pv.cohesive**of.history.pv.cohesive *tag1* <string> *tag2* <string> *tag3* <string> ...**

Set the items you want to write in ParaView results (See the list below).

Field keywords:

- default (Output all)
- velocity
- force
- displacement
- shear_strength
- dfn
- mat_id
- group

Example:

```
of.history.pv.cohesive 'default'
of.history.pv.cohesive 'velocity' 'force' 'displacement'
```

6.10.6 of.history.pv.damage**of.history.pv.damage *tag1* <string> *tag2* <string> *tag3* <string> ...**

Set the items you want to write in ParaView results (See the list below).

Field keywords:

- default (Output all)
- mode
- sliding
- opening
- area
- time
- length

Example:

```
of.history.pv.damage default  
of.history.pv.damage 'mode' 'time' 'length'
```

6.10.7 of.history.pv.ae

of.history.pv.ae *tag1* <string> *tag2* <string> *tag3* <string> ...

Set the items you want to write in ParaView results (See the list below). It can be used when AE mode is turned on.

Field keywords:

- default (Output all)
- mode
- time
- win_time
- win_kinetic
- kinetic
- magnitude
- energy

Example:

```
of.history.pv.ae default  
of.history.pv.ae 'mode' 'time' 'magnitude'
```

The input commands are started from the topmost class `openfdem`. which can also be shortened as `of..` In this tutorial, `of.` will be used for simplification.

6.11 General

| Function | Description |
|----------|---|
| of.new | Clear all old memory and start a new run. |

| Function | Description |
|---------------------|---|
| of.console.interval | Set the interval to show the results printed on the console screen. |
| of.damp.global | Set the global damping value to slow down the loading velocity. |
| of.finalize | Clear all the memory after the last run. |
| of.import | Import the file. |
| of.import.table | Import the table. |
| of.log.interval | Set the interval to print the results in .log file. |
| of.new | Clear all old memory and start a new run. |
| of.restore | Restore the model and rerun the data. |
| of.save | Save the model and make it possible to be restored. |
| of.save.FEM | Save the FEM results after in-situ stress equilibrium is reached. |
| of.set.config | Choose the plane stress or plane strain for the 2D problems. |
| of.set.contact | Set the model with contact detection and contact forces computation or not. |
| of.set.debug | Turn on the debug mode. |
| of.set.GBM | Assign the number of groups for minerals in GBM model. |
| of.set.gravity | Add the gravity in the model. |
| of.set.large | Set the flag of large deformation for the kinematics. |
| of.set.massscale | Set the mass scale value to lump the nodal mass and give a larger timestep. |
| of.set.module | Set the flags of different modules. |
| of.set.omp | Assign the number of cores used for CPU parallelization. |
| of.set.result | Set the path to store the result files. |
| of.set.rgbm | Assign the number of groups for minerals to create the realistic GBM model. |
| of.solve | Assign the mechanical ratio as the iteration threshold to exit the running. |
| of.step | Assign the cycle steps for the current run. |
| of.timestep | Set the user-defined timestep. |

6.12 Blast

| Function | Description |
|-------------------|---|
| of.blast.position | Create a blast borehole by its center and radius. |

6.13 Boundary Conditions

| Function | Description |
|----------------------------------|---|
| of.boundary.nodal.force | Create nodal force boundaries on nodal groups. |
| of.boundary.nodal.velocity | Create the nodal velocity boundary on the nodal groups. |
| of.boundary.nodal.inivelocity | Create the nodal initial velocity boundaries on the nodal groups. |
| of.boundary.nodal.acceleration | Create the nodal acceleration boundary on the nodal groups. |
| of.boundary.nodal.clear | Delete the nodal boundaries defined before. Directly set the nodal groups. |
| of.boundary.edge.force | Create the edge force boundary on the edge groups. |
| of.boundary.edge.velocity | Create the edge velocity boundary on the edge groups. |
| of.boundary.edge.inivelocity | Create the edge initial velocity boundaries on the edge groups. |
| of.boundary.edge.acceleration | Create the edge acceleration boundary on the edge groups. |
| of.boundary.edge.clear | Delete the edge boundaries defined before. |
| of.boundary.element.stress | Create the in-situ stress on the assigned element groups. |
| of.boundary.element.stress.xgrad | Create the in-situ stress gradient in x direction on the assigned element groups. |
| of.boundary.element.stress.ygrad | Create the in-situ stress gradient in y direction on the assigned element groups |
| of.boundary.element.clear | Delete the element boundaries defined before. Directly set the element groups. |
| of.boundary.blast | Create the blast borehole boundaries from table. |

6.14 DFN

| Function | Description |
|---------------------|--|
| of.DFN.connectivity | Create DFN by knowing its node id. |
| of.DFN.group | Create DFN groups by knowing the dfn id. |
| of.DFN.cohesive | Set the DFN as initial cohesive by groups. |
| of.DFN.broken | Set the DFN as initial broken fractures by groups. |

6.15 FEM

| Function | Description |
|------------------------------------|---|
| of.import.FEM.nodal.coord | Import the nodal by knowing node count and initial coordinates. |
| of.import.FEM.nodal.coord0 | Import the nodal by knowing node count and initial coordinates. |
| of.import.FEM.nodal.velocity | Import the nodal velocities by knowing node count and initial velocities. |
| of.import.FEM.nodal.groups | Import the nodal groups. |
| of.import.FEM.edge.groups | Import the edge groups. |
| of.import.FEM.element.groups | Import the element groups. |
| of.import.FEM.element.connectivity | Import the element connectivity. |

6.16 Geometry

| Function | Description |
|----------------------------|--|
| of.geometry.square | Create a rectangular object. |
| of.geometry.cut.square | Cut the rectangle from the existing geometry. |
| of.geometry.remove.square | Remove the rectangle from the existing geometry. |
| of.geometry.circle | Create a circle object. |
| of.geometry.cut.circle | Cut the circle from the existing geometry. |
| of.geometry.remove.circle | Remove the circle from the existing geometry. |
| of.geometry.ellipse | Create an ellipse object. |
| of.geometry.cut.ellipse | Cut the ellipse from the existing geometry. |
| of.geometry.remove.ellipse | Remove the ellipse from the existing geometry. |
| of.geometry.polygon | Create a polygon object. |
| of.geometry.cut.polygon | Cut the polygon from the existing geometry. |
| of.geometry.remove.polygon | Remove the polygon from the existing geometry. |
| of.geometry.table | Create an object based on the data in the table. |
| of.geometry.cut.table | Cut the object from the existing geometry. |
| of.geometry.remove.table | Remove the object from the existing geometry. |
| of.geometry.domain | Range of the geometry. |
| of.geometry.cut.joint | Cut a joint in an object. |
| of.geometry.cut.jset | Cut joint sets by assigning the dip, space, trace, and gap of the joints. |
| of.geometry.cut.DFN | Cut DFN sets by assigning the dip, length, and threshold method of the joints. |
| of.geometry.cut.arc | Create an arc. |
| of.geometry.minsize | Set the minsize to delete the bad segments for jsets and dfns. |
| of.geometry.minangle | Set the minangle to delete the bad segments for jsets and dfns. |
| of.geometry.iteration | Set the iteration to converge the generation of jsets and dfns. |
| of.geometry.group | Set group of the points in gmsh to set the new mesh size. |
| of.geometry.mesh.size | Set the mesh size to the tags. |
| of.geometry.recombine | Generate quadrangle element. |
| of.geometry.mesh | Set the method to create mesh. |
| of.geometry.mesh.write | Export the mesh file. |

6.17 Group

| Function | Description |
|------------------------------------|---|
| of.group.nodal | Set the group of nodals in mesh. |
| of.group.edge | Set the group of edges in mesh. |
| of.group.element | Set the group of elements in mesh. |
| of.group.cohelement | Set the group of cohesive elements in mesh. |
| of.group.nodal.from.element | Inherit the nodal group from their parent elements. |
| of.group.edge.from.element | Inherit the edge group from their parent elements. |
| of.group.edge.from.cohelement | Inherit the edge group from their parent cohelements. |
| of.group.edge.from.dfn | Inherit the edge group from DFN tags. |
| of.group.cohelement.from.dfn | Inherit the cohelement group from DFN tags. |
| of.group.cohelement.from.gbr | Create the cohesive element groups if they are located between the target element tags. |
| of.group.nodal.bool.union | Get the bool union of the second – last nodal groups and store them into the first tags. |
| of.group.nodal.bool.intersect | Get the bool intersect of the second – last nodal groups and store them into the first tags. |
| of.group.nodal.bool.subtract | Get the bool subtract of the second to other-last nodal groups and store them into the first tags. |
| of.group.edge.bool.union | Get the bool union of the second – last edge groups and store them into the first tags. |
| of.group.edge.bool.intersect | Get the bool intersect of the second – last edge groups and store them into the first tags. |
| of.group.edge.bool.subtract | Get the bool subtract of the second to other-last edge groups and store them into the first tags. |
| of.group.element.bool.union | Get the bool union of the second – last element groups and store them into the first tags. |
| of.group.element.bool.intersect | Get the bool intersect of the second – last element groups and store them into the first tags. |
| of.group.element.bool.subtract | Get the bool subtract of the second to other-last element groups and store them into the first tags. |
| of.group.cohelement.bool.union | Get the bool union of the second – last cohesive element groups and store them into the first tags. |
| of.group.cohelement.bool.intersect | Get the bool intersect of the second – last cohelement groups and store them into the first tags. |
| of.group.cohelement.bool.subtract | Get the bool subtract of the second to other-last cohelement groups and store them into the first tags. |

6.18 Mesh

| Function | Description |
|--------------------|---|
| of.mesh.insert | Insert cohesive in the target element groups. |
| of.mesh.insert.dfn | Insert cohesive only on a dfn plane, the dfn should cross the block. |
| of.mesh.voronoi | Change the delaunay element to Voronoi element. |
| of.mesh.split | Insert contacts within the element tags and no cohesive elements will be created. |
| of.mesh.bary | Refine the target element groups using bary separation scheme. |
| of.mesh.RG | Refine the target element groups using RG separation scheme. |
| of.mesh.RGB | Refine the target element groups using RGB separation scheme. |
| of.mesh.NVB | Refine the target element groups using NVB separation scheme. |
| of.mesh.baryGB | Refine the target element groups using baryGB separation scheme. |

6.19 Material

| Function | Description |
|-----------------------|---|
| of.mat.element | Assign the material parameters to the target element groups. |
| of.mat.cohesive | Assign the material parameters to the target cohesive element groups. |
| of.mat.contact | Assign the material parameters to the target element group pairs. |
| of.mat.hydro.matrix | Contact the developer for more information. |
| of.mat.hydro.fracture | Contact the developer for more information. |
| of.mat.mpm.fluid | Contact the developer for more information. |
| of.mat.mpm.solid | Contact the developer for more information. |
| of.mat.hydro.fluid | Contact the developer for more information. |
| of.mat.hydro.gas | Contact the developer for more information. |
| of.mat.nonlocal | Contact the developer for more information. |
| of.mat.thermal | Contact the developer for more information. |

6.20 History

| Function | Description |
|--------------------------------|--|
| of.history.pv.interval | Set the result writing intervals. |
| of.history.pv.dynamic.interval | Set the dynamic result writing intervals by the increasing broken CZM. |
| of.history.pv.field | Export field results. |
| of.history.pv.fracture | Export fracture results. |
| of.history.pv.cohesive | Export cohesive results. |
| of.history.pv.damage | Export damage results. |
| of.history.pv.ae | Export ae results. |

CHAPTER
SEVEN

COMMON CLASS

```
struct geo_curve_type
#include <openfdem_geometry.h>
```

Public Members

```
int n_line
```

```
int *line
```

```
struct geo_group_type
#include <openfdem_geometry.h>
```

Public Members

```
char *tag
```

```
int size
```

```
int line_size
```

```
int new_dfn_size
```

```
int embed_id
```

```
int *nodes
```

```
int *lines
```

```
int *new_dfn
```

```
struct geo_line_type
#include <openfdem_geometry.h>
```

Public Members

```
char *group  
  
int embed  
  
int embed_id  
  
int node_1  
  
int node_2  
  
struct geo_node_type  
#include <openfdem_geometry.h>
```

Public Members

Real x

Real y

Real z

Real mesh_size

```
struct geo_surface_type  
#include <openfdem_geometry.h>
```

Public Members

char *tag

int embed_id

int n_curve

int embed

int *curve

```
struct Geometry_structure  
#include <openfdem_geometry.h>
```

Public Members

Real **x0**

Real **x1**

Real **y0**

Real **y1**

int **node_num**

int **line_num**

int **curve_num**

int **surface_num**

int **group_num**

double **mesh_size**

double **min_size**

geo_node_type ***node**

geo_line_type ***line**

geo_curve_type ***curve**

geo_surface_type ***surface**

geo_group_type ***group**

file **openfdem_geometry.h**

#include “..*common/openfdem_struct.h*”

TypeDefs

```
typedef struct Geometry_structure *Geometry  
  
file push_geometry.h  
#include “../io/parser/parser_geometry.h”
```

Functions

```
void push_geometry_square(General general, Geometry geometry, int embed_flag)  
void push_geometry_polygon(General general, Geometry geometry, int embed_flag)  
void push_geometry_table(General general, Geometry geometry, int embed_flag)  
void push_geometry_circle(General general, Geometry geometry, int embed_flag)  
void push_geometry_ellipse(General general, Geometry geometry, int embed_flag)  
void push_geometry_arc(General general, Geometry geometry)  
void push_geometry_joint(General general, Geometry geometry)  
void push_geometry_jset(General general, Geometry geometry)  
void push_geometry_DFN(General general, Geometry geometry)  
void get_geo_node_group_square(Geometry geometry, char *tag, double x_lef, double x_rig, double  
y_bot, double y_top)  
void get_geo_node_group_square_on(Geometry geometry, char *tag, double x_lef, double x_rig, double  
y_bot, double y_top)  
void get_geo_node_group_plane_left(Geometry geometry, char *tag, double x1, double y1, double x2,  
double y2)  
void get_geo_node_group_plane_right(Geometry geometry, char *tag, double x1, double y1, double x2,  
double y2)  
void get_geo_node_group_plane(Geometry geometry, char *tag, double x1, double y1, double x2, double  
y2)  
void get_geo_node_group_circle_inner(Geometry geometry, char *tag, double x0, double y0, double  
radius)  
void get_geo_node_group_circle_outer(Geometry geometry, char *tag, double x0, double y0, double  
radius)  
void get_geo_node_group_circle(Geometry geometry, char *tag, double x0, double y0, double radius)  
void push_mesh_size(Geometry geometry, General general, char *keyword)  
void gmsh_api(Openfdem openfdem, Geometry geometry, char **argv, char *keyword, int write_flag)  
void geometry_free(Geometry geometry)
```

```
void push_geometry_rdfns(General general, Geometry geometry)
```

```
dir /Users/ekaterinaossetchkina/CProjects/openfdem_solver/openfdem src/src/geometry
```

```
dir /Users/ekaterinaossetchkina/CProjects/openfdem_solver/openfdem src/src
```

CHAPTER
EIGHT

GEOMETRY CLASS

```
struct energy_type
#include <openfdem_general.h>
```

Public Members

Real **external_work**

Real **internal_energy**

Real **gravity_energy**

Real **kinetic_energy**

Real **damping_energy**

Real **absorbing_energy**

Real **contact_energy**

Real **contact_nenergy**

Real **element_energy**

Real **cohesive_energy**

```
struct General_structure
#include <openfdem_general.h>
```

Public Members

```
long TotalStep  
  
config_type_enum config_type  
  
config_module_enum config_module  
  
short config_GBM  
  
int config_large  
  
long NowStep  
  
char RunTime[256]  
  
char InputFileName[256]  
  
FILE *InputFile  
  
FILE *CheckFile  
  
FILE *FilePointer[10]  
  
long FileID  
  
long debug  
  
char save_path[300]  
  
short save_file_flag  
  
of_vector gravity  
  
double TimeStep  
  
double CurrentTime  
  
Real global_damp  
  
long ShowTimeInterval
```

```
short save_flag  
  
char save_filename[300]  
  
short thread  
  
energy_type energy  
  
Real kinetic_ratio  
  
solver_base solver  
  
struct of_fracture  
#include <openfdem_struct.h>
```

Public Members

Real I

Real II

Real III

```
struct of_principal  
#include <openfdem_struct.h>
```

Public Members

Real p1

Real p2

Real p3

```
struct of_tensor  
#include <openfdem_struct.h>
```

Public Members

Real **xx**

Real **xy**

Real **yy**

Real **yx**

Real **zz**

struct **of_vector**

#include <openfdem_struct.h>

Public Members

Real **x**

Real **y**

struct **of_vectorn**

#include <openfdem_struct.h>

Public Members

Real **normal**

Real **shear**

struct **solver_base**

#include <openfdem_general.h>

Public Members

UInt **last_node_n**

UInt **last_ele_n**

UInt **last_cohele_n**

UInt **run_times**

file openfdem_common.h
#include <stdlib.h> #include <stdio.h> #include <string.h> #include <limits.h> #include <signal.h> #include
“openfdem_message.h” #include “openfdem_enum.h” #include “openfdem_struct.h”

Defines

FileNull

Char2DNull

Double1DNull

Double2DNull

Double3DNull

Int1DNull

Int2DNull

Edge2Length(l, x1, y1, x2, y2)

V2Darea(s, l1, l2, l3)

Functions

void **get_filename_no_path**(char *return_string, char *input_string)
void **get_path**(char *return_string, char *input_string)
void **parse_string**(FILE *InputFile, char *string)
void **get_filename_no_suffix**(char *return_string, char *input_string)
void **get_suffix**(char *return_string, char *input_string)
void **strip_ext**(char *fname)
double ***malloc_1D_double_array**(long m1)
double ***initialize_1D_double_array**(long m1, double dini)
double ****malloc_2D_double_array**(long m2, long m1)
double ****initialize_2D_double_array**(long m2, long m1, double dini)
double *****malloc_3D_double_array**(long m3, long m2, long m1)

```
double ***initialize_3D_double_array(long m3, long m2, long m1, double dini)
long *malloc_1D_int_array(long m1)
long *initialize_1D_int_array(Int m1, Int iini)
UInt *initialize_1D_UInt_array(UInt m1, UInt iini)
long **malloc_2D_int_array(long m2, long m1)
long **initialize_2D_int_array(long m2, long m1, long iini)
void parse_1D_double_array(FILE *fileptr, double initial_double, long max_size, long actual_size, double **d1array)

void parse_2D_double_array(FILE *fileptr, double initial_double, long max_size1, long max_size2, long read_way, long actual_size1, long actual_size2, double ***d2array)

void parse_3D_double_array(FILE *fileptr, double dinit, long n1, long n2, long n3, double ****d3array)
void parse_1D_int_array(FILE *fileptr, long initial_int, long max_size, long actual_size, long **i1array)
void parse_2D_int_array(FILE *fileptr, long initial_int, long max_size1, long max_size2, long read_way,
                       long actual_size1, long actual_size2, long ***i2array)

void get_vector_unit(double *e1x, double *e1y, double rx, double ry)
void get_vector_rotated(double *e1x, double *e1y, double e2x, double e2y)
void get_vector_global_to_local(double *u, double *v, double rx, double ry, double e1x, double e1y,
                                double e2x, double e2y)

int openfdem_run_usr_break(double dctime, long ncstep, long mcstep)

long get_line_current_position(FILE *fileptr)
void create_result_store_folder(char *Result_store_folder)
void NULL_vector(of_vector *d1array)
void parse_vector(FILE *fileptr, double dinit, of_vector *d1array)

void parse_1D_vector(FILE *fileptr, double dinit, long max_size, long actual_size, of_vector **d1array)
of_tensor *initialize_1D_tensor(unsigned int size, double double_ini)
of_tensor *parse_1D_tensor(FILE *fileptr, UInt size, Real double_ini)

UInt *realloc_UInt_array(UInt *array, UInt size)
int *realloc_Int_array(int *array, int size)
```

file **openfdem_enum.h**

Enums

enum **config_type_enum**

Values:

enumerator **CONFIG_NONE**

enumerator **PLANESTRESS**

enumerator **PLANESTRAIN**

enum **mesh_enum**

Values:

enumerator **INSERT**

enumerator **SPLIT**

enumerator **REMESH3**

enumerator **REMESH4**

enumerator **REMESH6**

enumerator **IMPLICIT**

enum **boundary_fluid_enum**

Values:

enumerator **POREPRESSURE**

enumerator **WATER_LEVEL**

enumerator **PRESSURE**

enumerator **FLOW**

enumerator **IMPERMABLE**

enum **boundary_thermal_enum**

Values:

enumerator **INI_TEMPERATURE**

enumerator **TEMPERATURE**

enumerator **CONVECTION**

enumerator **FLUX**

enumerator **RADIATION**

enumerator **SOURCE**

enum **boundary_node_enum**

Values:

enumerator **FORCE**

enumerator **FORCE_X**

enumerator **FORCE_Y**

enumerator **VEL**

enumerator **VEL_X**

enumerator **VEL_Y**

enumerator **INIVEL**

enumerator **INIVEL_X**

enumerator **INIVEL_Y**

enumerator **VISCOUS**

enumerator **VISCOUS_X**

enumerator **VISCOUS_Y**

enumerator **ACCEL**

enumerator **ACCEL_X**

enumerator **ACCEL_Y**

enumerator **FORCE_LOCAL**

enumerator **FORCE_LOCAL_X**

enumerator **FORCE_LOCAL_Y**

enumerator **VEL_LOCAL**

enumerator **VEL_LOCAL_X**

enumerator **VEL_LOCAL_Y**

enumerator **INIVEL_LOCAL**

enumerator **INIVEL_LOCAL_X**

enumerator **INIVEL_LOCAL_Y**

enumerator **ACCEL_LOCAL**

enumerator **ACCEL_LOCAL_X**

enumerator **ACCEL_LOCAL_Y**

enumerator **NORMAL**

enumerator **SHEAR**

enum **cohelement_enum**

Values:

enumerator **INTACT**

enumerator **DAMAGE**

enumerator **BROKEN**

enum **DFN_enum**

Values:

enumerator **NDFN**

enumerator **DFN_COHESIVE**

enumerator **DFN_BROKEN**

enum **boundary_element_enum**

Values:

enumerator **STRESS**

enumerator **GRADIENT_X**

enumerator **GRADIENT_Y**

enum **config_module_enum**

Values:

enumerator **MECHANICAL**

enumerator **HYDRO**

enumerator **THERMAL**

enumerator **BLAST**

enumerator **BLAST_HYDRO**

enumerator **DYNAMIC**

enum **element_cons**

Values:

enumerator **ELASTIC**

enumerator **RIGID**

enumerator **SOFTEN**

enumerator **TRANSVERSE**

enumerator **MC**

enumerator **DP**

enumerator OTHERS

enum **cohelement_cons**

Values:

enumerator COH_EM

enumerator COH_EM_FRIC

enumerator COH_LINEAR

enumerator COH_OP

enumerator COH_BB

enumerator COH_EM_HET

enumerator COH_EM_ANI

enumerator COH_EM_DYNAMIC

enumerator COH_OTHERS

enum **contact_cons**

Values:

enumerator CON_MC

enumerator CON_DYNAMIC_MC

enumerator CON_HERTZ

enumerator CON_BB

enumerator CON_FRICTION

enumerator CON_OTHERS

enum **contact_detection**

Values:

enumerator LIG

enumerator **NBS**

enumerator **mNBS**

enumerator **CELL**

enum **contact_force**

Values:

enumerator **DEFAULT_MUNJIZA**

enumerator **LIG_F**

enum **random_method**

Values:

enumerator **RANDOM_NULL**

enumerator **RANDOM_CONSTANT**

enumerator **RANDOM_GAUSSSE**

enumerator **RANDOM_UNIFORM**

enumerator **RANDOM_EXPONENTIAL**

enumerator **RANDOM_LOG_NORMAL**

enumerator **RANDOM_POWER**

enumerator **RANDOM_FISHER**

enum **his_nodal_t**

Values:

enumerator **HIS_DIS**

enumerator **HIS_VEL**

enumerator **HIS_FOR**

enumerator **HIS_FLUID_P**

enumerator **HIS_FRAC_P**

enumerator **HIS_MATRIX_P**

enumerator **HIS_TEMP**

enum **his_ele_t**
Values:

- enumerator **HIS_STRESS**
- enumerator **HIS_STRAIN**
- enumerator **HIS_STRAIN_R**
- enumerator **HIS_PLASTIC**

enum **his_cohele_t**
Values:

- enumerator **HIS_COH_DIS**
- enumerator **HIS_COH_FOR**
- enumerator **HIS_COH_VEL**
- enumerator **HIS_COH_SHEAR**

enum **his_contact_t**
Values:

- enumerator **HIS_CON_DIS**
- enumerator **HIS_CON_FOR**
- enumerator **HIS_CON_VEL**

enum **his_general_t**
Values:

- enumerator **HIS_TIME**
- enumerator **HIS_CYCLE**

```
enumerator HIS_N_FRACTURE  
  
enumerator HIS_N_DAMAGE  
  
enumerator HIS_N_AE  
  
enumerator HIS_ENERGY_KINETIC  
  
enumerator HIS_ENERGY_ELEMENT_STRAIN  
  
enumerator HIS_ENERGY_FRACTURE  
  
enumerator HIS_ENERGY_AE  
  
enumerator HIS_ENERGY_DAMP  
  
enumerator HIS_ENERGY_SLIP  
  
enumerator HIS_ENERGY_PLASTIC  
  
enumerator HIS_UNBALANCE_FORCE  
  
enumerator HIST_ENERGY_RATIO
```

file openfdem_general.h
 #include "openfdem_common.h"

Typedefs

```
typedef struct General_structure *General
```

file openfdem_io_binary.h
 #include "../solve/openfdem.h"

Functions

```
void openfdem_save_binary(char *filename, struct OpenFDEM_structure openfdem)  
    filename openfdem_math.c  
    author Xiaofeng Li xqli@whrsm.ac.cn  
    date creation: Fri Jun 18 2010 date last modification: Tue Sep 29 2020  
    function: common math
```

8.1 LICENSE

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

```
void openfdem_restore_binary(char *filename, struct OpenFDEM_structure openfdem)
```

file openfdem_math.h

```
#include <math.h>#include "openfdem_struct.h"
```

Defines

Openfdem_pos_epsilon

filename *openfdem_math.h*

author Xiaofeng Li xfli@whrsm.ac.cn

date creation: Fri Jun 18 2010 date last modification: Tue Sep 29 2020

function: common math

8.2 LICENSE

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

Openfdem_neg_epsilon

Openfdem_max_value

Openfdem_PI

Openfdem_e

Openfdem_double_abs(x)

Openfdem_max(x, y)

Openfdem_min(x, y)

Openfdem_sign(x)

Matrix2Inverse(m, minv, det)

Vector2DotProduct(s, x1, y1, x2, y2)

Vector2Normalize(s, x1, y1)

Vector2Cro(s, x1, y1, x2, y2)

Functions

Real **random_normal**(Real mu, Real sigma)

file **openfdem_message.h**

```
#include <stdlib.h>#include <stdio.h>#include <string.h>#include <limits.h>#include <signal.h>
```

Defines

Parse_int(file, x)

Parse_double(file, x)

parse_keyword(file, x)

Dump_int(file, x, ndigit)

Dump_double(file, x, ndigit)

dump_string(file, x)

Dump_statement(file, x, ndigit)

Functions

void **dump_message**(char *c1)

void **dump_message_keyword**(char *c1, char *c2)

void **check_parse_status**(FILE *checkfile, char *keyword)

void **dump_message_mat**(char *c1, int c2)

void **dump_message_group**(int n1, char *c1, char *c2)

void **dump_message_type2**(int c2, char *c1)

Variables

```
static char *DoubleString[20] = {"%le", "%+1.0le", "%+2.0le", "%+3.0le", "%+4.0le", "%+5.0le",
"%+6.0le", "%+7.0le", "%+8.1le", "%+9.2le", "%+10.3le", "%+11.4le", "%+12.5le", "%+13.6le", "%+14.7le",
"%+15.8le", "%+16.9le", "%+17.10le", "%+18.11le", "%+19.12le"}
```

filename *openfdem_message.h*

author Xiaofeng Li xfli@whrsm.ac.cn

date creation: Fri Jun 18 2010 date last modification: Tue Sep 29 2020

function: message

8.3 LICENSE

This library is free software; you can redistribute it and/or modify it under the terms of the GNU Lesser General Public License as published by the Free Software Foundation; either version 2.1 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more details.

```
static char *IntString[20] = {"%ld", "%lld", "%2ld", "%3ld", "%4ld", "%5ld", "%6ld", "%7ld", "%8ld",
"%9ld", "%10ld", "%11ld", "%12ld", "%13ld", "%14ld", "%15ld", "%16ld", "%17ld", "%18ld", "%19ld"}
```

```
static char *CharString[30] = {"%s", "%1s", "%2s", "%3s", "%4s", "%5s", "%6s", "%7s", "%8s", "%9s",
"%10s", "%11s", "%12s", "%13s", "%14s", "%15s", "%16s", "%17s", "%18s", "%19s", "%20s", "%21s",
"%22s", "%23s", "%24s", "%25s", "%26s", "%27s", "%28s", "%29s"}
```

file openfdem_struct.h

Defines

UInt

Int

Real

file openfdem_time.h

```
#include "../solve/openfdem.h"
```

Functions

```
void Openfdem_show_time(General general, Nodal nodal, Contact contact)
    filename openfdem_time.h
    author Xiaofeng Li xfl@whrsm.ac.cn
    date creation: Fri Jun 18 2010 date last modification: Tue Sep 29 2020
    function: time
```

8.4 LICENSE

This library is free software; you can redistribute it and/or
modify it under the terms of the GNU Lesser General Public
License as published by the Free Software Foundation; either
version 2.1 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful,
but WITHOUT ANY WARRANTY; without even the implied warranty of
MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU
Lesser General Public License for more details.

```
void openfdem_date_time(char *date_time)
void openfdem_stamp_time(void)
double openfdem_time_count(void)
```

file openfdem_tools.h

```
#include <stdlib.h>#include <stdio.h>#include <string.h>#include <limits.h>#include <signal.h>#include
“openfdem_struct.h”#include “./solid/solid_mechanics/cohesive/openfdem_solid_cohesive.h”#include
“./solid/solid_mechanics/cohesive/seismic/openfdem_seismic.h”#include “./solid/openfdem_nodal.h”#include
“./common/openfdem_general.h”
```

Functions

```
fracture_type *realloc_fracture_structure(fracture_type *array, UInt size)
damage_type *realloc_damage_structure(damage_type *array, UInt size)
ae_type *realloc_ae_structure(ae_type *array, UInt size)
void parse_node(FILE *fileptr, long max_size, long actual_size, node_type **d1aray, General general)
```

CHAPTER
NINE

SOLID CLASS

struct **ae_type**

#include <openfdem_sesmic.h> this structure is responsible for...

ae_type(5, 6, 7, magnitude=9)

Param time

Param win_time

Param win_kinetic

Public Members

Real **time**

Real **win_time**

Real **win_kinetic**

Real **kinetic**

Real **magnitue**

UInt **coele**

union **any_cohmat_type**

#include <openfdem_solid_cohesive_materials.h>

Public Members

solidcoh_EM_t **EM**

solidcoh_linear_t **linear**

solidcohhet_EM_t **EM_het**

solidcohani_EM_t **EM_ani**

solidcohdyn_EM_t **EM_dyn**

solidcoh_OP_t **OP**

union **any_conmat_type**

#include <openfdem_solid_contact_materials.h>

Public Members

solidcon_MC_t **MC**

solidcon_dynamic_MC_t **D_MC**

union **any_mat_type**

#include <openfdem_solid_materials.h>

Public Members

solid_elastic_t **elastic**

solid_soften_t **soften**

solid_rigid_t **rigid**

solid_transverse_t **transverse**

solid_MC_t **MC**

struct **Coh_element_structure**

#include <openfdem_solid_cohesive.h>

Public Members

```
cohelement_type *cohele  
  
UInt cohe_num  
  
UInt cohe_ini_num  
  
UInt broken_num  
  
UInt damage_num  
  
Real *shear_strength  
  
of_vectorn *force  
  
of_vectorn *dis  
  
of_vectorn *vel  
  
fracture_type *fracture  
  
damage_type *damage  
  
Real *het_value  
  
of_vectorn *DIF_0  
  
of_vectorn *DIF_1  
  
of_vectorn *DIF_2  
  
Int *flag  
  
struct cohelement_type  
#include <openfdem_solid_cohesive.h>
```

Public Members

Real **x**

Real **y**

UInt **matid**

UInt **n_node**

cohelement_enum state

Int **stater**

DFN_enum dfn

UInt ***nodes**

UInt ***ele**

UInt ***edge**

of_fracture damage

Real **energy**

Real **area**

UInt **old_map**

struct **Contact_structure**

#include <openfdem_solid_contact.h>

Public Members

UInt **n_contact**

short ***potential_contact_element**

contact_type ***con**

Int **detection_count**

```
Int contact_count

contact_detection detection

contact_force force_model

Real contact_trigger

Real contact_buffer

Real fn

Real fs

struct contact_type
#include <openfdem_solid_contact.h>
```

Public Members

```
UInt con_n

Int head

Int *hold

Real *sliding_distance[6]

short *sliding_state[6]

Int *target

short *type

short *slip_flag
```

```
struct damage_type
#include <openfdem_solid_cohesive.h>
```

Public Members

Real **time**

Real **length**

Real **type**

UInt **coele**

```
struct DFN_structure
#include <openfdem_DFN.h>
```

Public Members

UInt **n_dfn**

dfn_type ***dfn**

```
struct dfn_type
#include <openfdem_DFN.h>
```

Public Members

char ***tag**

UInt **n_1**

UInt **n_2**

DFN_enum **type**

Int **cohele**

```
struct Element_structure
#include <openfdem_solid.h>
```

Public Members

```
element_type *ele  
  
UInt e_num  
  
UInt e_ini_num  
  
Real e_min_size  
  
Real e_mean_size  
  
Real **soften_state  
  
of_tensor *stress  
  
of_tensor *strain  
  
of_tensor *strain_rate  
  
struct element_type  
#include <openfdem_solid.h>
```

Public Members

```
UInt matid  
  
Real x  
  
Real y  
  
UInt n_node  
  
Real area  
  
UInt *nodes  
  
Int *couple  
  
Real radius  
  
Real energy
```

```
UInt n_group  
  
UInt *group  
  
Real minedge  
  
short excavation  
  
short uncontacted  
  
struct fracture_type  
#include <openfdem_solid_cohesive.h>
```

Public Members

```
Real time  
  
Real length  
  
Real energy  
  
Real type  
  
UInt coele
```

```
struct mat_cohelement_t  
#include <openfdem_mat.h>
```

Public Members

```
char *tag  
  
cohelement_cons cons  
  
any_cohmat_type *mat  
  
struct mat_contact_t  
#include <openfdem_mat.h>
```

Public Members

Int **emat1**

Int **emat2**

contact_cons **cons**

any_conmat_type ***mat**

struct **mat_element_t**

#include <openfdem_mat.h>

Public Members

char ***tag**

element_cons **cons**

any_mat_type ***mat**

Real **damp**

struct **mat_fluid_cohesive_t**

#include <openfdem_mat.h>

Public Members

Real **a_0**

Real **a_min**

Real **para_exp**

Real **para_b**

struct **mat_fluid_element_t**

#include <openfdem_mat.h>

Public Members

Real **permiability**

Real **Biot_modulus**

Real **Biot_c**

```
struct mat_fluid_t  
#include <openfdem_mat.h>
```

Public Members

Real **den**

Real **bulk**

Real **viscosity**

Real **cohesion**

```
struct mat_gas_t  
#include <openfdem_mat.h>
```

Public Members

Real **initial_den**

Real **initial_bulk**

Real **permiability**

Real **constant_B**

Real **alpha**

```
struct mat_thermal_element_t  
#include <openfdem_mat.h>
```

Public MembersReal **conductivity**Real **specific_heat**Real **expansion**

```
struct Nodal_structure
#include <openfdem_nodal.h>
```

Public MembersUInt **dimension**UInt **n_num**UInt **n_ini_num***node_type* ***node***of_vector* ***total_force***of_vector* ***nodal_vel***of_vector* ***dis_excavate**Int ****net_connect**Real **element_maximuum_unbalance_force**Real **cohesive_maximuum_unbalance_force**Real **contact_maximuum_unbalance_force**Real **max_velocity**Real **unbalance_force**

```
struct node_type
#include <openfdem_nodal.h>
```

Public Members

Real **mass**

Real **x0**

Real **y0**

Real **x**

Real **y**

UInt **n_group**

UInt ***group**

UInt **old_map**

```
struct Seismic_structure
#include <openfdem_seismic.h> this is a brief description
```

```
seismic_structure(method, window, 4);
```

Param method

this variable does...

Param window

Public Members

UInt **method**

Real **window**

UInt **ae_num**

ae_type ***ae**

```
struct solid_elastic_t
#include <openfdem_solid_materials.h>
```

Public Members

Real **E**

Real **v**

Real **den**

Real **K**

Real **G**

```
struct Solid_mat_structure
#include <openfdem_mat.h>
```

Public Members

UInt **n_mat**

UInt **n_cohmat**

UInt **n_conmat**

UInt **n_hydro_mat**

UInt **n_hydro_coh**

UInt **n_thermal_mat**

mat_element_t ***elemat**

mat_cohelement_t ***cohelemat**

mat_contact_t ***conmat**

mat_fluid_t **water**

mat_gas_t **gas**

mat_fluid_element_t ***hydro_ele**

mat_fluid_cohesive_t ***hydro_coh**

```
mat_thermal_element_t *thermal_ele
```

```
short hysteretic_flag
```

```
struct solid_MC_t
```

```
#include <openfdem_solid_materials.h>
```

Public Members

Real **E**

Real **v**

Real **den**

Real **K**

Real **G**

Real **ten**

Real **coh**

Real **fri**

```
struct solid_rigid_t
```

```
#include <openfdem_solid_materials.h>
```

Public Members

Real **den**

```
struct solid_soften_t
```

```
#include <openfdem_solid_materials.h>
```

Public Members

Real **E**

Real **v**

Real **den**

Real **K**

Real **G**

Real **stretch**

Real **coh**

Int **damage**

```
struct solid_transverse_t  
#include <openfdem_solid_materials.h>
```

Public Members

Real **E_x**

Real **E_y**

Real **den**

Real **v_xy**

Real **v_yx**

Real **G**

```
struct solidcoh_EM_t  
#include <openfdem_solid_cohesive_materials.h>
```

Public Members

Real **pn**

Real **pt**

Real **ten**

Real **coh**

Real **fri**

Real **GI**

Real **GII**

Real **beta_n**

Real **beta_t**

Real **thermal_h**

```
struct solidcoh_linear_t  
#include <openfdem_solid_cohesive_materials.h>
```

Public Members

Real **pn**

Real **pt**

Real **ten**

Real **coh**

Real **fri**

Real **GI**

Real **GII**

```
struct solidcoh_OP_t  
#include <openfdem_solid_cohesive_materials.h>
```

Public MembersReal **pn**Real **pt**Real **ten**Real **coh**Real **delta_n**Real **delta_s**

```
struct solidcohani_EM_t
#include <openfdem_solid_cohesive_materials.h>
```

Public MembersReal **power**Real **dip**Real **pn**Real **pt**Real ***ten**Real ***coh**Real ***fri**Real ***GI**Real ***GII**

```
struct solidcohdyn_EM_t
#include <openfdem_solid_cohesive_materials.h>
```

Public Members

Real **pn**

Real **pt**

Real **ten**

Real **coh**

Real **fri**

Real **GI**

Real **GII**

Real **c_rate_n**

Real **c_rate_s**

Real **exp_n**

Real **exp_s**

```
struct solidcohett_EM_t
{
    #include <openfdem_solid_cohesive_materials.h>
```

Public Members

Real **pn**

Real **pt**

Real ***ten**

Real ***coh**

Real ***fri**

Real ***GI**

Real ***GII**

```
struct solidcon_dynamic_MC_t
#include <openfdem_solid_contact_materials.h>
```

Public Members

Real **kn**

Real **ks**

Real **static_fri**

Real **residual_fri**

Real **slip_rate**

```
struct solidcon_MC_t
#include <openfdem_solid_contact_materials.h>
```

Public Members

Real **fri**

Real **kn**

Real **ks**

file openfdem_mat.h

```
#include " ../../common/openfdem_enum.h" #include "solid_mat/openfdem_solid_materials.h" #include
"solid_mat/cohesive/openfdem_solid_cohesive_materials.h" #include "solid_mat/contact/openfdem_solid_contact_materials.h"
```

TypeDefs

typedef struct *Solid_mat_structure* ***Solidmat**

file openfdem_solid_materials.h

file push_element_materials.h

```
#include " ../../../../io/parser.h"
```

Functions

```
void push_element_materials(General general, Element element, Group group, Solidmat solidmat, char *keyword)
```

file openfdem_solid_cohesive_materials.h

file push_cohesive_element_materials.h

```
#include "../io/parser.h"
```

Functions

```
void push_cohesive_element_materials(General general, Cohelement cohelement, Group group, Solidmat solidmat, char *keyword)
```

file openfdem_solid_contact_materials.h

file push_contact_materials.h

```
#include "../io/parser.h"
```

Functions

```
void push_contact_materials(General general, Solidmat solidmat, Contact contact, char *keyword)
```

file openfdem_nodal.h

```
#include "../common/openfdem_struct.h"
```

Typedefs

```
typedef struct Nodal_structure *Nodal
```

file solid_mechanics.h

```
#include "../solve/openfdem.h" #include "../common/openfdem_math.h"
```

Functions

```
void Openfdem_element_force_update(Openfdem openfdem)
```

Parameters

openfdem -

- nodal forces **

```
void Openfdem_cohesive_force_update(Openfdem openfdem)
```

Parameters

openfdem -

- nodal forces **

file openfdem_solid.h

```
#include "../common/openfdem_struct.h"#include "../common/openfdem_commonom.h"
```

Typedefs

typedef struct *Element_structure* *Element

file solid_mechanics_elastic_triangle.h

```
#include "../solve/openfdem.h"
```

Functions

void Solid_mechanics_elastic_triangle(Element element, Solidmat solidmat, General general, UInt elementid, Nodal nodal)

file solid_mechanics_soften_triangle.h

```
#include "../solve/openfdem.h"
```

Functions

void Solid_mechanics_soften_triangle(Element element, Solidmat solidmat, UInt elementid, Nodal nodal, Real *soften_state)

file solid_mechanics_transverse_triangle.h

```
#include "../solve/openfdem.h"
```

Functions

void Solid_mechanics_transverse_triangle(Element element, Solidmat solidmat, General general, UInt elementid, Nodal nodal)

file openfdem_DFN.h

Typedefs

typedef struct *DFN_structure* *DFN

file openfdem_solid_cohesive.h

```
#include "../common/openfdem_struct.h"#include "../common/openfdem_commonom.h"
```

TypeDefs

```
typedef struct Coh_element_structure *Cohelement
```

file openfdem_sesmic.h

TypeDefs

```
typedef struct Seismic_structure *Seismic
```

file solid_mechanics_cohesive_4_node_EM.h

```
#include “../solve/openfdem.h”
```

Functions

```
void Solid_mechanics_cohesive_4_node_EM(Nodal nodal, UInt cohelement_id, Real CurrentTime, Real  
TimeStep, Cohelement cohelement, Element element,  
Contact contact, Solidmat solidmat, Seismic seismic,  
General general)
```

file solid_mechanics_cohesive_4_node_EM_anisotropic.h

```
#include “../solve/openfdem.h”
```

Functions

```
void solid_mechanics_cohesive_4_node_EM_anisotropic(Nodal nodal, UInt cohelement_id, Real  
CurrentTime, Real TimeStep, Cohelement  
cohelement, Element element, Solidmat solidmat,  
Seismic seismic)
```

file solid_mechanics_cohesive_4_node_EM_hetero.h

```
#include “../solve/openfdem.h”
```

Functions

```
void solid_mechanics_cohesive_4_node_EM_hetero(Nodal nodal, UInt cohelement_id, Real  
CurrentTime, Real TimeStep, Cohelement  
cohelement, Element element, Solidmat solidmat,  
Seismic seismic)
```

file solid_mechanics_cohesive_4_node_linear.h

```
#include “../solve/openfdem.h”
```

Functions

```
void Solid_mechanics_cohesive_4_node_linear(Nodal nodal, UInt cohelement_id, Real CurrentTime,  

                                             Real TimeStep, Cohelement cohelement, Element  

                                             element, Solidmat solidmat, Seismic seismic)
```

file contact_add_list_from_cohesive.h

```
#include "../../../solid/solid_mechanics_contact/openfdem_solid_contact.h" #include "../../../common/openfdem_math.h"
```

Functions

```
void contact_add_list_from_cohesive(Contact contact, UInt ielem, UInt jelem, short slip)  

void malloc_contact(Contact contact, UInt ielem, UInt jelem)  

void realloc_contact(Contact contact, UInt ielem, UInt jelem)
```

file contact_detection.h

```
#include "../../../solve/openfdem.h"
```

Functions

```
void Openfdem_contact_detection(Openfdem openfdem)
```

Parameters

openfdem –

- contact detection **

file contact_detection_method.h

```
#include "../../../solve/openfdem.h"
```

Functions

```
void Contact_dection_method_NBS(Contact contact, General general, Element element, Cohelement  

                                 cohelement, Nodal nodal)  

void Contact_dection_method_mNBS(Contact contact, General general, Element element, Cohelement  

                                 cohelement, Nodal nodal)  

void Contact_dection_method_LIG(Contact contact, General general, Element element, Cohelement  

                                 cohelement, Nodal nodal)
```

file contact_force.h

```
#include "../../../solve/openfdem.h"
```

Functions

```
void Openfdem_contact_force(Openfdem openfdem)
```

Parameters

`openfdem` –

- nodal forces **

file contact_force_triangle_interact.h

```
#include "../solve/openfdem.h"
```

Functions

```
void Contact_force_triangle(General general, Element element, Solidmat solidmat, UInt conmat,  
Contact contact, Nodal nodal)
```

```
void Contact_force_triangle_LIG(General general, Element element, Solidmat solidmat, UInt conmat,  
Contact contact, Nodal nodal)
```

file get_potential_contact_couple.h

```
#include "../solid/solid_mechanics/openfdem_solid.h"#include "../solid/solid_mechanics_cohesive/openfdem_solid_cohesive.h"  
"../solid/solid_mechanics_contact/openfdem_solid_contact.h"#include "../common/openfdem_struct.h"
```

Functions

```
UInt get_potential_contact_couple(Element element, Contact contact, Cohelement cohelement)
```

file openfdem_solid_contact.h

```
#include "../common/openfdem_struct.h"#include "../common/openfdem_commonom.h"
```

Typedefs

```
typedef struct Contact_structure *Contact
```

*dir /Users/ekaterinaossetchkina/CProjects/openfdem_solver/openfdem src/src/solid/
solid_mechanics_cohesive/DFN*

dir /Users/ekaterinaossetchkina/CProjects/openfdem_solver/openfdem src/src/solid/materials

*dir /Users/ekaterinaossetchkina/CProjects/openfdem_solver/openfdem src/src/solid/
solid_mechanics_cohesive/seismic*

dir /Users/ekaterinaossetchkina/CProjects/openfdem_solver/openfdem src/src/solid

*dir /Users/ekaterinaossetchkina/CProjects/openfdem_solver/openfdem src/src/solid/
materials/solid_mat*

dir /Users/ekaterinaossetchkina/CProjects/openfdem_solver/openfdem src/src/solid/materials/solid_mat_cohesive

dir /Users/ekaterinaossetchkina/CProjects/openfdem_solver/openfdem src/src/solid/materials/solid_mat_contact

dir /Users/ekaterinaossetchkina/CProjects/openfdem_solver/openfdem src/src/solid/solid_mechanics

dir /Users/ekaterinaossetchkina/CProjects/openfdem_solver/openfdem src/src/solid/solid_mechanics_cohesive

dir /Users/ekaterinaossetchkina/CProjects/openfdem_solver/openfdem src/src/solid/solid_mechanics_contact

dir /Users/ekaterinaossetchkina/CProjects/openfdem_solver/openfdem src/src

**CHAPTER
TEN**

INDEX

- genindex