OpenFDEM

Release 3.5.3

Grasselli's Geomechanics Group

Aug 01, 2023

CONTENTS:

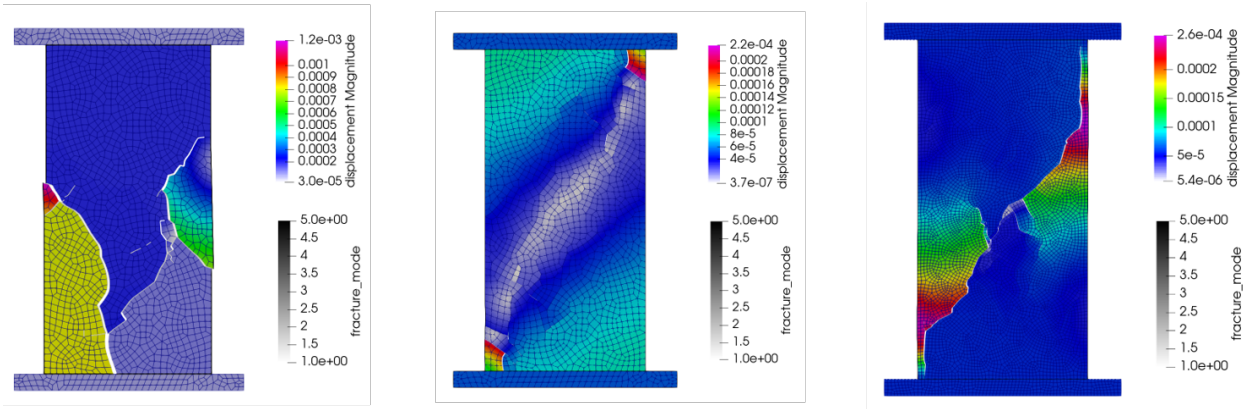
KEYWORD COMMANDS TUTORIAL MANUAL

CHAPTER

ONE

ABOUT OPENFDEM

OpenFDEM aims to be a free and object-oriented finite and discrete element solver for solving multiscale, multi-phase and multiphysics (3M) problems that can be operated on various platforms, such as Linux, Unix and Windows (95/98/NT/2000/XP) systems. The applications are, but not limited to, mechanical, thermal and fluid problems. The program is also optimized from design to algorithm to boost the computation speed. This project started from 2018 when I was a PhD candidate in Monash University, and the very basic version was finished in 2020 when I graduated from Institute of Rock and Soil Mechanics, China Academy of Science. After that, I turned to a full-time developer and co-PI of this project in University of Toronto since September in 2021, under the supervision of Professor Giovanni Grasselli. More details about OpenFDEM were presented in the Global FDEM workshop "FDEM-2022-Modeling innovations and numerical experiments in geomechanics" which held in Toronto on December 9, 2022. The recording of this workshop can be found at https://geogroup.utoronto.ca/global-fdem-2022/.



Website:

- https://openfdem.com/
- https://geogroup.utoronto.ca/
- https://xiaofengli-uoft.github.io/Mainpage

Global FDEM workshop 2022 reports->@

- https://geogroup.utoronto.ca/global-fdem-2022/

Special issue of Combined Finite-Discrete Element Modelling in Rock Mechanics and Rock Engineering ->@

- http://www.jrmge.cn/newscontent-4-135.html

Xiaofeng Li, Giovanni Grasselli University of Toronto

CHAPTER

TWO

CONTENTS

2.1 About: What Can OpenFDEM Do

OpenFDEM (an Open Hybrid Finite-Discrete Element solver) is a scientific software for the numerical solution of partial differential equations, also open to other coupling of physical problems such as thermal, hydro, fluid dynamics, rigid collision, etc. It can deal with such problems of (1D, 2D and 3D) and time problems (static and/or transient).

2.1.1 Object Oriented Architecture (C++ and CUDA®)

The main feature of **OpenFDEM** is the use of modern C++ object-oriented programming, and the organization of data structure defining from the explicit input keywords. It consists of a working environment in which any problem is defined using a limited number of objects, which makes • it approachable and easy to use at both the development and application levels as the initial intention. • the environment structured and concise. Thus, it provides researchers with advanced development tools and great freedom to add new features.

2.1.2 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.

- **Highly accurate and reliable** - The kernel provides high-order integration schemes and solving methods to seek more reliable numerical results which are comparable to theoretical solutions. The element type has a maximum order of three to produce the large deformation behavior in the entity, and the new kinematic scheme to construct a nonlinear deformation. The Hilber-Hughes-Taylor (HHT) time integration scheme (second-order accuracy) is employed for the explicit solver. The Y-like codes are limited to the first order time integration and constant strain element.
- **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

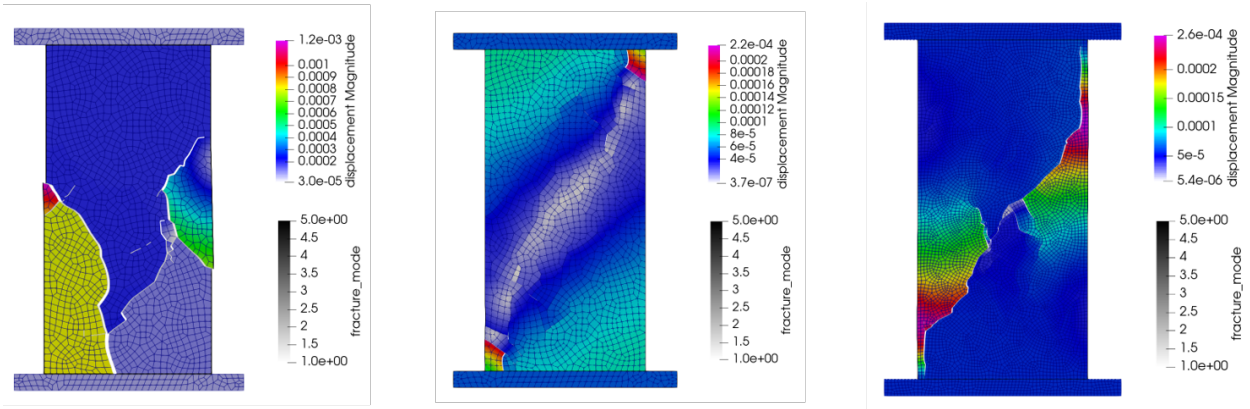


Fig. 2: Quadratic elements are supported in OpenFDEM. Combined linear element (quadrilateral and triangle elements, left), quadratic quadrilateral element (middle) and refined linear quadrilateral element (right).

(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.

TODO:: DFN picture (page 8)

- **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(NlogN) 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.

TODO:: Grain scale fracture imaging

- **Large material library including the state-of-the-art models for phase field of quasi-brittle 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

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

dilation shear law and so on). (The OpenFDEM is the only code supports materials beyond the community of rock mechanics)

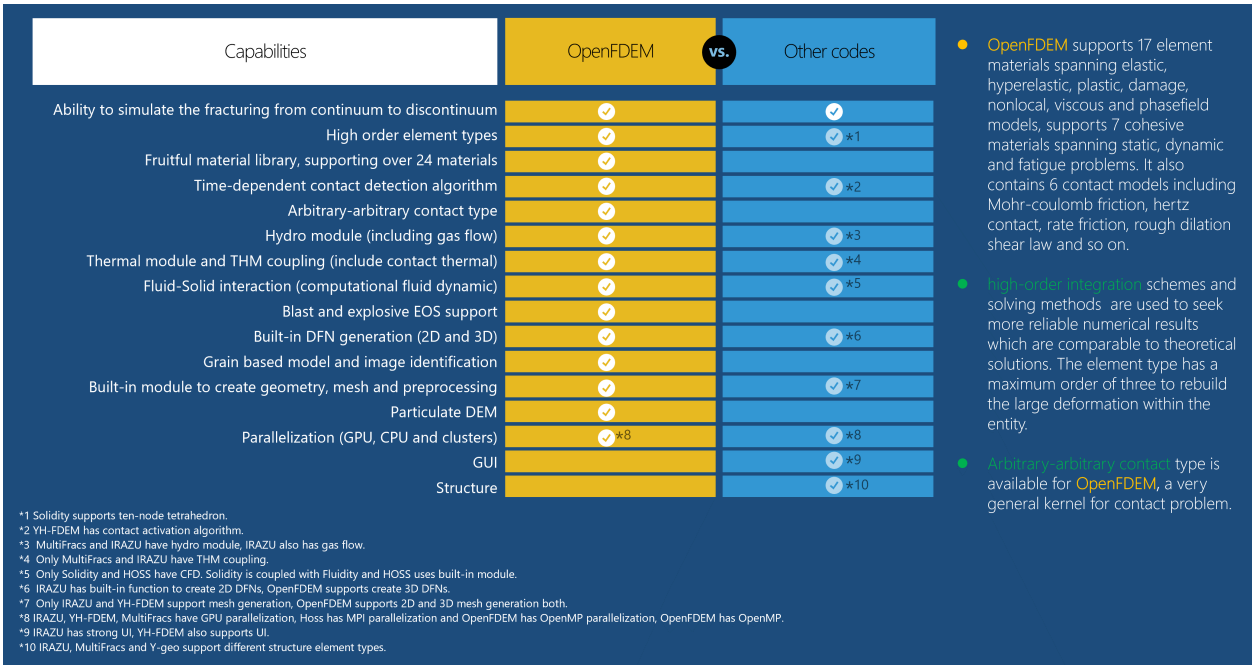


Fig. 6: New material library in OpenFDEM.

Fig. 7: Phasefield module in OpenFDEM.

- **Advanced analysis solvers** - Linear dynamic solver (implicit and explicit), linear static solver (PETSC), eigenvalue 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.

2.1.3 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.
- **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.

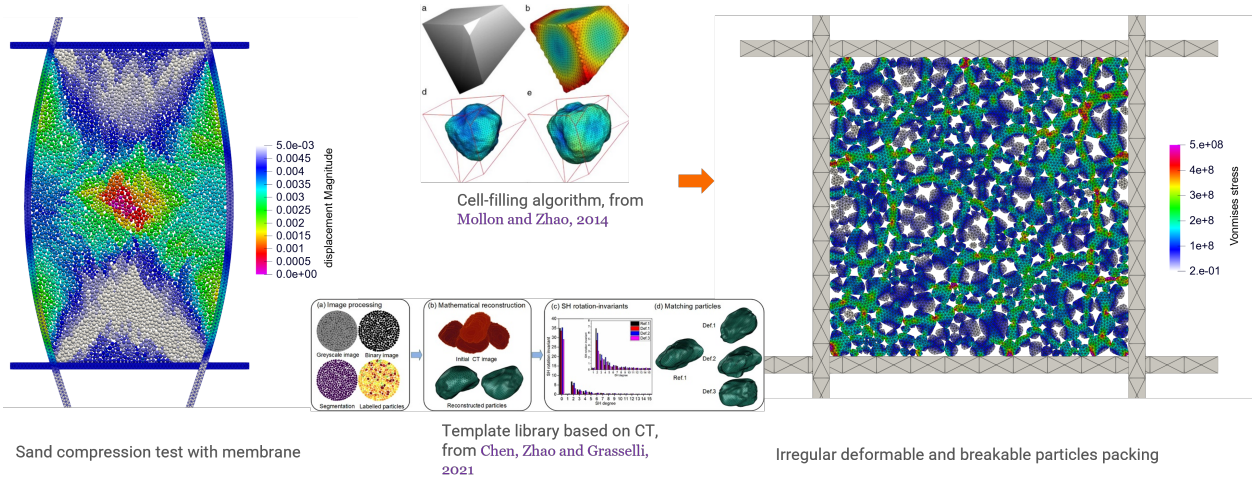


Fig. 8: Sand compression test with membrane (left) and irregular deformable and breakable particles packing (right).

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).

2.1.4 Fluid Dynamic Module

- **Analysis Procedures** - matrix flow for pore seepage, transient incompressible fracture flow, transient compressible fracture flow and gas flow problems.

Fig. 10: Blast considering gas expansion, gas flow by hydro module (left) and without gas expansion by mechanical module (right) ((Wang et al. 2021, 2022)).

- **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.

2.1.5 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.
- **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.

Fig. 11: Fluid injection in fractured rock block.

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

2.1.6 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 between FDEM and MPM makes the solid interacting with fluid possible.

Post-Processing

- Export to VTK format is supported, allowing to use VTK based visualization tools (such as ParaView) for post-processing on different operating platforms.
- Export to Tecplot format is supported.
- Export historic variables which are monitored at each step to `.csv` is supported.

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)

2.1.7 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.

2.2 Contact information

2.3 System Requirements

2.3.1 Build Requirements

To compile `OpenFDEM`, you neeed a complier supporting `C++ 17` and the following packages are required:

- **CMake** (>=3.5.1)
- **OpenMP** - a high-performance, freely available package for multi core acceleration

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

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

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

- **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)

2.3.2 Post-Processing

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

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

2.3.3 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.

2.4 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.

2.4.1 Source Code Download

The source code is hosted on the University of Toronto Gitlab: OpenFDEM Gitlab.

2.4.2 System Requirements

- Windows x64
- Visual Studio (> 2015)

All external dependencies are included in the OpenFDEM download.

2.5 Copyright

The OpenFDEM documentation is compiled and maintained by: Xiaofeng Li, Katia (Ekaterina) Ossetchkina and Grace (Yuqi) Hu.

```
  ____                    ______ _____  ______ __  __
 / __ \                  |  ____|  __ \|  ____|  \/  |
| |  | |_ __   ___ _ __ | |__  | |  | | |__  | \  / |
| |  | | '_ \ / _ \ '_ \|  __| | |  | |  __| | |\/| |
| |__| | |_) |  __/ | | | |    | |__| | |____| |  | |
 \____/| .__/ \___|_| |_|_|    |_____/|______|_|  |_|
       | |                     OpenFree Finite Element
       |_|                         and Discrete Element Method Solver

   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.
```

2.6 Collaboration Opportunities

Under construction

2.7 Publications

Under construction

2.8 Theory

2.8.1 theory2

In this paper, the following conventions are assumed for the notations: (a) Gibbs notation is employed for tensor algebra and the order is denoted by the number of underdots (e.g. $\underset{\Omega}{\mathbf{u}} = u_i \mathbf{e_i}$,$\sigma = \sigma_{ij} \mathbf{e_i} \otimes \mathbf{e_j}$); (b) the dot symbol means the contraction between two tensors (e.g. $\underset{\Omega}{\mathbf{T}} = \sigma \cdot \underset{\Omega}{\mathbf{n}}$); (3) the colon symbol denotes for the double contraction of two second-order tensors (e.g. $\sigma = \underline{\mathbf{D}} : \varepsilon$); (4) the wedge symbol $\wedge$ is for the cross product (e.g. $\mathbf{e}_3 = \mathbf{e}_1 \wedge \mathbf{e}_2$) and (5) the tensorial product $\sigma = \underset{\Omega}{\mathbf{x}} \otimes \underset{\Omega}{\mathbf{F}}$ states for the linear production two tensors.

2.1 Weak formulation of updated Lagrange

Consider an arbitrary body $\Omega \subset \mathbb{R}^D \left(D \in \{ 1 , 2 , 3 \}\right)$ with external boundary $\partial\Omega$ and the internal discontinuity boundary Γ, $\underset{\Omega}{\mathbf{x}}$ is the current position vector of the point and $\underset{\Omega}{\mathbf{u}}\left(\underset{\Omega}{\mathbf{x}}, t\right) \subset \mathbb{R}^D$ is the displacement at time $t \in [0 , t_{all}]$. The infinitesimal strain tensor based on small deformation and deformation gradients assumptions is characterized as $\varepsilon\left(\underset{\Omega}{\mathbf{x}}, t\right) \in \mathbb{R}^{D \times D}$.

Balance of mass:

in $\forall \underset{\Omega}{\mathbf{x}} \in \Omega$

Balance of momentum:

$\frac{d}{dt} \int_\Omega \rho \underset{\Omega}{\dot{\mathbf{u}}}\left(\underset{\Omega}{\mathbf{x}}\right) d\Omega = \int_\Omega \rho \underset{\Omega}{\ddot{\mathbf{u}}}\left(\underset{\Omega}{\mathbf{x}}\right) d\Omega = \int_{\partial\Omega} \underset{\Omega}{\mathbf{t}} d\partial\Omega + \int_\Omega \underset{\Omega}{\mathbf{b}} d\Omega$ in $\forall \underset{\Omega}{\mathbf{x}} \in \Omega$

ρ is the density of the solid body. The force is split into tractions $\underset{\Omega}{\mathbf{t}}$ acted on boundaries $\partial\Omega$and into body force $\underset{\Omega}{\mathbf{b}}$ in volume $\Omega \mapsto \mathbb{R}^D$. Similarly, the conservation of the moment of momentum can be written as

$\int_\Omega \rho \underset{\Omega}{\ddot{\mathbf{u}}}\left(\underset{\Omega}{\mathbf{x}}\right) \wedge \underset{\Omega}{\mathbf{x}} d\Omega = \int_{\partial\Omega} \underset{\Omega}{\mathbf{t}} \wedge \underset{\Omega}{\mathbf{x}} d\partial\Omega + \int_\Omega \underset{\Omega}{\mathbf{b}} \wedge \underset{\Omega}{\mathbf{x}} d\Omega$ in $\forall \underset{\Omega}{\mathbf{x}} \in \Omega$

In terms of the Cauchy stress tensor, the traction force satisfies $\underset{\Omega}{\mathbf{t}} = \sigma \cdot \underset{\Omega}{\mathbf{n}}$, Eq is rewritten according to Gausss theorem $\int_{\partial\Omega} \underset{\Omega}{\mathbf{t}} d\partial\Omega = \int_{\partial\Omega} \sigma \cdot \underset{\Omega}{\mathbf{n}} d\partial\Omega = \int_\Omega \sigma \cdot \nabla d\Omega$. Since the stress tensor is symmetric based on Eq as $\sigma = \sigma^T$, the conservation of momentum is

$\rho \underset{\Omega}{\ddot{\mathbf{u}}}\left(\underset{\Omega}{\mathbf{x}}\right) - \nabla \cdot \sigma - \underset{\Omega}{\mathbf{b}} = 0$ in $\forall \underset{\Omega}{\mathbf{x}} \in \Omega$

Boundary conditions:

$\sigma \cdot \underset{\Omega}{\mathbf{n}} = \underset{\Omega}{\mathbf{t}}$ on $\forall \underset{\Omega}{\mathbf{x}} \in \partial\Omega_t$

$\underset{\Omega}{\mathbf{u}} = \underset{\Omega}{\overline{\mathbf{u}}}$ on $\forall \underset{\Omega}{\mathbf{x}} \in \partial\Omega_u$

Initial conditions:

$\underset{\Omega}{\mathbf{u}}\left(\underset{\Omega}{\mathbf{x}}\,,\ t\ =\ 0\right)=\overline{\underset{\Omega}{\mathbf{u}}\left(\underset{\Omega}{\mathbf{x}}\right)}$ on $\forall\underset{\Omega}{\mathbf{x}}\in\partial\Omega_u$

$\underset{\Omega}{\mathbf{u}}\left(\underset{\Omega}{\mathbf{x}}\,,\ t\ =\ 0\right)=\overline{\underset{\Omega}{\dot{\mathbf{u}}}\left(\underset{\Omega}{\mathbf{x}}\right)}$ on $\forall\underset{\Omega}{\mathbf{x}}\in\partial\Omega_u$

The spatial version of the principle of virtual work states that the body Ω is in equilibrium when the Cauchy stress satisfies the initial condition

$\int_\Omega\left(\left(\nabla\ \cdot\ \sigma\right)\ \cdot\ \delta\ \underset{\Omega}{\mathbf{v}}\ -\ \left(\underset{\Omega}{\mathbf{b}}\ -\ \rho\ \underset{\Omega}{\ddot{\mathbf{u}}}\right)\ \cdot\ \delta\ \underset{\Omega}{\mathbf{v}}\right)d\Omega=0$ in $\forall\underset{\Omega}{\mathbf{v}}\in\Omega$

Integration by parts of the first term in Eq gives

$\int_\Omega\left(\nabla\ \cdot\ \sigma\right)\cdot\delta\underset{\Omega}{\mathbf{v}}\,d\Omega=\int_\Omega\left(\sigma\ :\ \nabla\ \delta\ \underset{\Omega}{\mathbf{v}}\right)d\Omega-\int_{\partial\Omega}\left(\sigma^T\ \delta\ \underset{\Omega}{\mathbf{v}}\ \cdot\ \underset{\Omega}{\mathbf{n}}\right)d\partial\Omega=\int_\Omega\left(\sigma\ :\ \nabla\ \delta\ \underset{\Omega}{\mathbf{v}}\right)d\Omega-\int_{\partial\Omega_t}\left(\underset{\Omega}{\mathbf{t}}\ \cdot\ \delta\ \underset{\Omega}{\mathbf{v}}\right)d\partial\Omega$ in $\forall\underset{\Omega}{\mathbf{v}}\in\Omega$

Substitution of Eq. back into Eq gives

$\int_\Omega\left(\sigma\ :\ \nabla\ \delta\ \underset{\Omega}{\mathbf{v}}\ -\ \left(\underset{\Omega}{\mathbf{b}}\ -\ \rho\ \underset{\Omega}{\ddot{\mathbf{u}}}\right)\ \cdot\ \delta\ \underset{\Omega}{\mathbf{v}}\right)d\Omega=\int_{\partial\Omega_t}\underset{\Omega}{\mathbf{t}}\cdot\delta\underset{\Omega}{\mathbf{v}}\,d\partial\Omega$ in $\forall\underset{\Omega}{\mathbf{v}}\in\Omega$

2.2 Spatial discretization in finite domain

In this section, Eq is discretised to derive the mass matrix, internal force and external force which is convenient to the matrix notation. Let the generic finite element $e\in\Omega$ is defined by n_{node} nodes with shape function $\mathcal{N}^{(e)}\left(\underset{\Omega}{\mathbf{x}}\right)$ associated with node i having coordinate $\underset{\Omega}{\mathbf{x}}$. The global interpolation matrix is defined as [57]

$\mathcal{N}^g\left(\underset{\Omega}{\mathbf{x}}\right)=\left[diag\ \left[\mathcal{N}_1^g\ \left(\underset{\Omega}{\mathbf{x}}\right)\right]\ \ diag\ \left[\mathcal{N}_2^g\ \left(\underset{\Omega}{\mathbf{x}}\right)\right]\ \cdots\ diag\ \left[\mathcal{N}_{n_{poin}}^g\ \left(\underset{\Omega}{\mathbf{x}}\right)\right]\right]$

where $diag\left[\mathcal{N}_1^g\ \left(\underset{\Omega}{\mathbf{x}}\right)\right]$ is the $n_{\dim}\times n_{\dim}$ diagonal matrix. The element displacement in global coordinate is

for $\forall\underset{\Omega}{\mathbf{x}}\in\Omega^e\ {}^h\underset{\Omega}{\mathbf{u}}\left(\underset{\Omega}{\mathbf{x}}\right)=\sum_{i=1}^{n_{poi}}\mathcal{N}_i\left(\underset{\Omega}{\mathbf{x}}\right)\underset{\Omega}{\mathbf{u}}=\mathcal{N}^g\underset{\Omega}{\mathbf{u}}$

$$\mathcal{B}^g=\begin{bmatrix}\mathcal{N}_{1,1}^g & 0 & \mathcal{N}_{2,1}^g & 0 & \cdots & \mathcal{N}_{n_{poin},1}^g & 0\\ 0 & \mathcal{N}_{1,2}^g & 0 & \mathcal{N}_{2,2}^g & \cdots & 0 & \mathcal{N}_{n_{poin},2}^g\\ \mathcal{N}_{1,2}^g & \mathcal{N}_{1,1}^g & \mathcal{N}_{2,2}^g & \mathcal{N}_{2,1}^g & \cdots & \mathcal{N}_{n_{poin},2}^g & \mathcal{N}_{n_{poin},1}^g\end{bmatrix}$$

Therefore, the discretised virtual work expression considering the kinetic is

$\int_{{}^h\Omega}\left[\sigma^T\ \mathcal{B}^g\ \delta\ \underset{\Omega}{\mathbf{v}}\ -\ \left(\underset{\Omega}{\mathbf{b}}\ -\ \rho\ \underset{\Omega}{\ddot{\mathbf{u}}}\right)\ \cdot\ \mathcal{N}^g\ \delta\ \underset{\Omega}{\mathbf{v}}\right]d\Omega-\int_{\partial^h\Omega_t}\underset{\Omega}{\mathbf{t}}\cdot\mathcal{N}^g\delta\underset{\Omega}{\mathbf{v}}\,d\partial\Omega=0$ in $\forall\delta\underset{\Omega}{\mathbf{v}}\in\Omega$

Since the above equation is satisfied for all vectors $\delta\underset{\Omega}{\mathbf{v}}$, equation can be reformulated as

$\mathbf{M}^{FE}\underset{\Omega}{\ddot{\mathbf{u}}}+\underset{\Omega}{\mathbf{f}}^{int}\left(\underset{\Omega}{\mathbf{u}}\right)-\underset{\Omega}{\mathbf{f}}^{ext}=0$

while the internal force $\underset{\Omega}{\mathbf{f}}^{int}\left(\underset{\Omega}{\mathbf{u}}\right)$, external force $\underset{\Omega}{\mathbf{f}}^{ext}$and inertia mass $\mathbf{M}^{FE}$ are

$\underset{\Omega}{\mathbf{f}}^{int}\left(\underset{\Omega}{\mathbf{u}}\right)=\int_{{}^h\Omega}\left(\mathcal{B}^g\right)^T\cdot\sigma\left(\underset{\Omega}{\mathbf{u}}\right)d\Omega$

$\underset{\Omega}{\mathbf{f}}^{ext}=\int_{{}^h\Omega}\left(\mathcal{N}^g\right)^T\rho\underset{\Omega}{\mathbf{b}}\,d\Omega+\int_{\partial^h\Omega_t}\left(\mathcal{N}^g\right)^T\underset{\Omega}{\mathbf{t}}d\partial\Omega=0$

$\mathbf{M}^{FE}=\int_{{}^h\Omega}\left(\mathcal{N}^g\right)^T\rho\mathcal{N}^g d\Omega$

$\mathcal{B}^g$ and $\mathcal{N}^g$ are matrices incorporating the interpolation functions and their spatial derivatives in global, respectively. The actual force vectors are assembled as

$$\underset{\Omega}{\mathbf{f}}^{int} = \underset{n_{elem}}{\overset{e=1}{\mathbf{A}}} \left(\underset{\Omega(e)}{\mathbf{f}}^{int} \right)$$
$$\underset{\Omega}{\mathbf{f}}^{ext} = \underset{n_{elem}}{\overset{e=1}{\mathbf{A}}} \left(\underset{\Omega(e)}{\mathbf{f}}^{ext} \right)$$

where $\underset{n_{elem}}{\overset{e=1}{\mathbf{A}}}$ is the finite element assembly operator and the element force vector can be obtained as

$$\underset{\Omega(e)}{\mathbf{f}}^{int} \left(\underset{\Omega}{\mathbf{u}} \right) = \int_{e\Omega} \left(\mathcal{B}^e \right)^T \cdot \sigma \left(\underset{\Omega}{\mathbf{u}} \right) d\Omega$$

$$\underset{\Omega(e)}{\mathbf{f}}^{ext} = \int_{e\Omega} \left(\mathcal{N}^e \right)^T \rho \underset{\Omega}{\mathbf{b}} \, d\Omega + \int_{\partial^e \Omega_t} \left(\mathcal{N}^e \right)^T \underset{\Omega}{\mathbf{t}} d\partial\Omega = 0$$

The superscripts e denotes the variables in local element.

2.3 Dynamic relaxation and time integration scheme

The continuum-discontinuum method (CDM) employs the explicit time integration based on velocity verlet algorithm to solve Eq . ${}^{n+1}\underset{\Omega}{\dot{\mathbf{u}}}$ and ${}^{n}\underset{\Omega}{\mathbf{u}}$ are the velocity vector and displacement vector at ${}^{n+1}t$ and ${}^{n}t$ for a time step Δt

$${}^{n}\underset{\Omega}{\mathbf{u}} = \frac{1}{2\Delta t} \left({}^{n+1}\underset{\Omega}{\ddot{\mathbf{u}}} - {}^{n}\underset{\Omega}{\dot{\mathbf{u}}} \right)$$

$${}^{n}\underset{\Omega}{\mathbf{u}} = {}^{n+1}\underset{\Omega}{\dot{\mathbf{u}}} - {}^{n}\underset{\Omega}{\mathbf{u}}$$

If the damping matrix $\mathbf{C}^{FE} = \alpha \mathbf{M}^{FE}$ is considered, the governing equation is written as

$$\mathbf{M}^{FE} \underset{\Omega}{\ddot{\mathbf{u}}} + \mathbf{C}^{FE} \underset{\Omega}{\mathbf{u}} + \underset{\Omega}{\mathbf{f}}^{int} \left(\underset{\Omega}{\dot{\mathbf{u}}} \right) - \underset{\Omega}{\mathbf{f}}^{ext} = 0$$

Eq. can be rewritten in the form of nodal velocities

$${}^{n+1}\underset{\Omega}{\dot{\mathbf{u}}} = \left[(1 - 2\,\alpha\,\Delta t)^n \, \underset{\Omega}{\dot{\mathbf{u}}} + \frac{2\Delta t}{\mathbf{M}^{FE}} \left({}^{n}\underset{\Omega}{\mathbf{f}}^{ext} - {}^{n}\underset{\Omega}{\mathbf{f}}^{int} \left({}^{n}\underset{\Omega}{\mathbf{u}} \right) \right) \right]$$

while the damping coefficient is $\alpha = 2\xi\omega$, ξ is the damping ratio and ω is the frequency parameter. The stability of the explicit scheme is depended on the critical timestep, which is determined by

$$\Delta t_{cri} = \gamma \frac{2}{\omega} \left(\sqrt{\xi^2 + 1} - \xi \right)$$

In CDM, each triangular element is considered as elastic with constant strain. During each timestep, the element strain rate is computed as

$$\dot{\varepsilon} = \frac{1}{2} \left(\nabla \underset{\Omega}{\mathbf{u}} + \nabla^T \underset{\Omega}{\dot{\mathbf{u}}} \right), \dot{\theta} = \frac{1}{2} \left(\nabla \underset{\Omega}{\mathbf{u}} - \nabla^T \underset{\Omega}{\dot{\mathbf{u}}} \right), \nabla \underset{\Omega}{\mathbf{u}} \cong \frac{1}{2A} \sum_s \left(\underset{\Omega}{\dot{\mathbf{u}}}^{(a)} + \underset{\Omega}{\dot{\mathbf{u}}}^{(b)} \right) \cdot \mathbf{n} \Delta s$$

The incremental strain tensor is denoted as $\Delta\varepsilon_{ij} = \dot{\varepsilon}_{ij}\Delta t$, $\mathbf{n}$ is unit vector in normal direction, A is the area and s is the length of edge, the incremental stress is updated according to the constitutive relations of the isotropic elastic blocks as

$$\Delta\sigma = \lambda \underset{v}{\dot{\varepsilon}} \, \delta\Delta t + 2\mu\dot{\mathbf{e}}\Delta t$$

where δ is Kronecker symbol, λ and G are Lame constants. The updated stress and strain at n step are

$${}^{n+1}\sigma = {}^{n}\sigma + \Delta\sigma, {}^{n+1}\dot{\varepsilon} = {}^{n}\dot{\varepsilon} + \Delta\varepsilon$$

2.4 Contact algorithm in CDM

The contact algorithms in CDM is divided into: (a) **neighbour search** and (b) **geometric resolution.** Neighbour search is a rough search aims to provide a list of possible blocks in contact including global search e.g. an altering digital tree (ADT) [58], no binary search (NBS) contact detection algorithm [59], CGRID [60], DESS algorithm [61], master-slave algorithm[62] and local search, e.g. node-to-surface (NTS) algorithm[63], RIGID algorithm[64], inside-outside algorithm [65]. Geometric resolution algorithms strongly depend on complexity of the geometric representation of cells. The discrete function representation (DFR) algorithm has a computational complexity of order O(N) [66] and the common-plane (CP), which bisects the space between two contacting blocks, is advantageous for vertex-to-vertex or edge-to-vertex contacts where the definition of the contact normal is a non-trivial problem and the method has a complexity of order O(N) [67]. The number of iterations of CP is depended on the accuracy of the initial guess of the CP position, and then a fast common plane (FCP) method [68] and a shortest link (SL) method [69] are proposed to increase efficiency of the CP method.

In this study, the contact detection algorithm is developed for blocks with evident size using square bounding box method in cooperation with the NBS algorithm (shown in Figure 2), which aims to divide the NBS model to several groups (list in Table 1), the flow chart of the enhanced NBS algorithm is

$$\underset{\Omega}{\mathbf{x}}^k = 1 + int\left(\frac{\underset{\Omega_i}{\mathbf{x}} - \underset{\Omega_{\min}}{\mathbf{x}}}{d(0)} + \frac{1}{2}\right)$$

|image1|

Figure 2. The schematic algorithm of NBS contact detection in CDM.

The enhanced NBS contact detection method used in mGbCDM

Table 1: Table 1

Step 1: Loop the blocks and find the maximum size buffer box for the initial group box $d(0) = d_{\max}$;
Step 2: Divide the blocks into n groups with size of buffer box for the nth group box as $d(n) = d_{\max} \cdot \alpha^{n-1}$, $\alpha \in (0\ ,\ \ 1]$;
Step 3: All the blocks are mapped in to the grid space with edge length of $d(0)$ as depicted in Figure 2 , the central point of the block $\underset{\Omega}{\mathbf{x}}^k$ is computed in Eq ;
Step 4: Loop all the blocks and detect contacts for the first group, the contact couple groups is identified when $\left\lvert \underset{\Omega}{\mathbf{x}}^{(t)} - athbfx^{(c)} \right\rvert < \max\left(d^{(t)} + d^{(c)}\right)$, the contact state can be recognized as neighboring contacts or center contacts;
Step 5: Repeat step 3 and step 4 for all groups of the remaining blocks and identify the states of the contacts;

2.9 Tutorials

2.9.1 Tutorial 0: Create Geometry

wait for updates.

2.9.2 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.

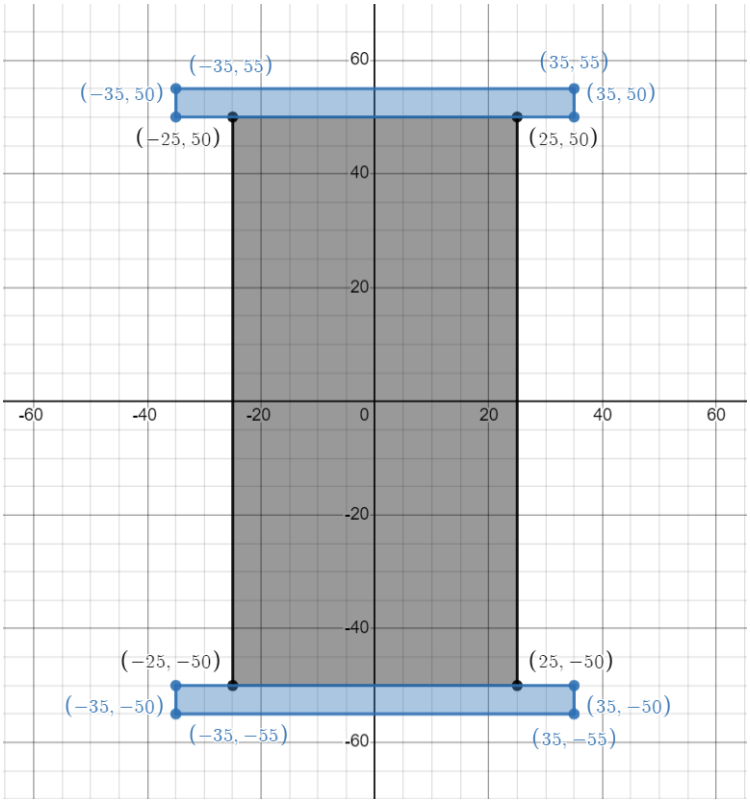


Fig. 16: Figure 1: Geometry of the uniaxial compression strength test

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.

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 geometris.

```
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

The material properties of this model is shown as the table below:

Parameter	Value
Continuum Triangular Elements	
model	elastic
density (kg/m^3)	2700
E (Pa)	30e9
ν	0.3
damp	1.0
Cohesive Material Properties	
model	EM
tension (Pa)	1e6
cohesion (Pa)	3e6
friction	0.3
GI (J/m^2)	10
GII (J/m^2)	50
Contact Material Properties	
model	MC
friction	0.3

Note: Check *element materal*, *cohesive materal*, *contact materal* to see more materials.

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 the 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.

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.

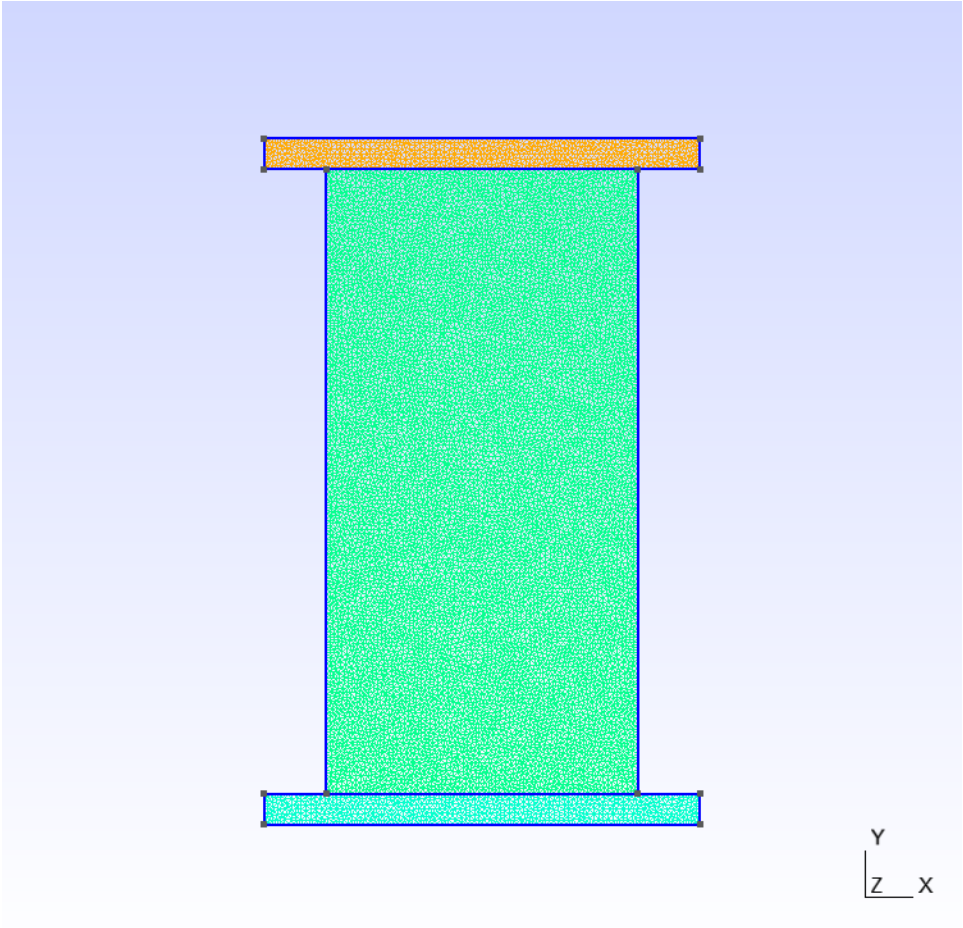
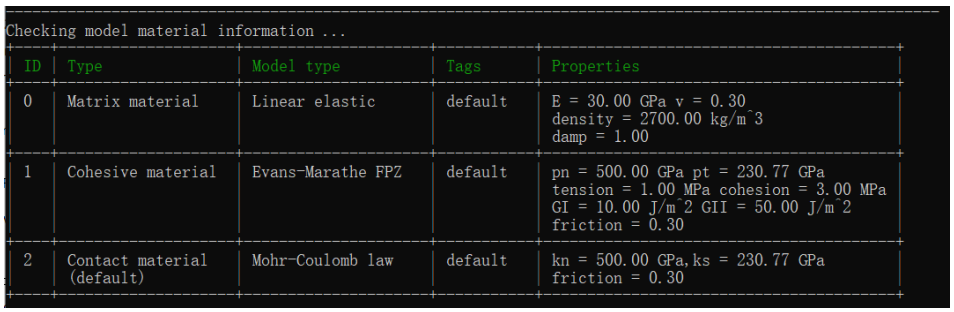


Fig. 17: Figure 2: Mesh created by Gmsh

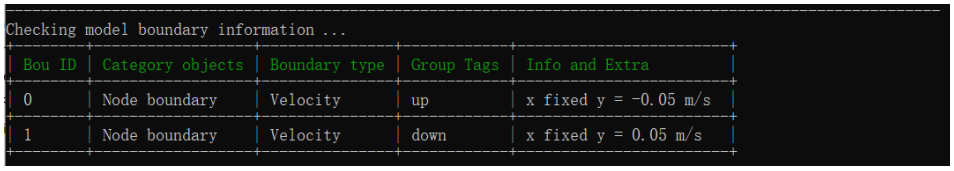
group ID	Type	Number of Object	Tags	Extra
0	Node	782	up	
1	Node	773	down	
2	Element	20622	rock	Triangle, T3
3	Element	1372	up	Triangle, T3
4	Element	1354	down	Triangle, T3
	Total nodes	63421		
	Total elements	23348		
	Total cohesives	31121		
	Total entities	20624		

Fig. 18: 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-Marathe 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  | Mohr-Coulomb law   | default  | kn = 500.00 GPa,ks = 230.77 GPa         |
|    | (default)         |                    |          | friction = 0.30                         |
+----+-------------------+--------------------+----------+-----------------------------------------+
```

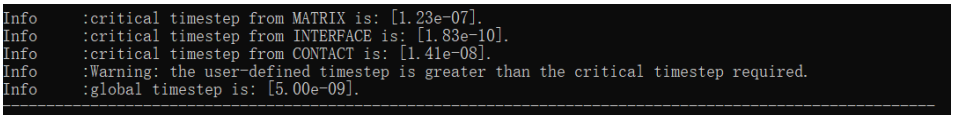
Fig. 19: Figure 4: Check the material assignment from command window



```
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. 20: 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: [1.41e-08].
Info     :Warning: the user-defined timestep is greater than the critical timestep required.
Info     :global timestep is: [5.00e-09].
```

Fig. 21: Figure 6: Check the timesteps from command window

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

4.0 Results

Wait for updates.

```
Start running ...
Info     :Nodal neumann boundaries are updated.
Info     :Edge neumann boundaries are updated.
Info     :Nodal dirichlet boundaries are updated.
Info     :Edge dirichlet boundaries are updated.
+------------+-----------------+-----------------+-----------------+------------+------------+---------------+
|    step    | physical time(s)|    unbalance    |     kinetic     |   broken   |  contact   | run time(s)   |
|            |                 |   force(N)      |      ratio      |    CZM     |   pairs    |               |
+------------+-----------------+-----------------+-----------------+------------+------------+---------------+
|          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. 22: Figure 7: Start to run the program

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 parallization will be␣
↪automatically turned on
# when you use the following command, otherwise the serilization 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 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
```

(continues on next page)

(continued from previous page)

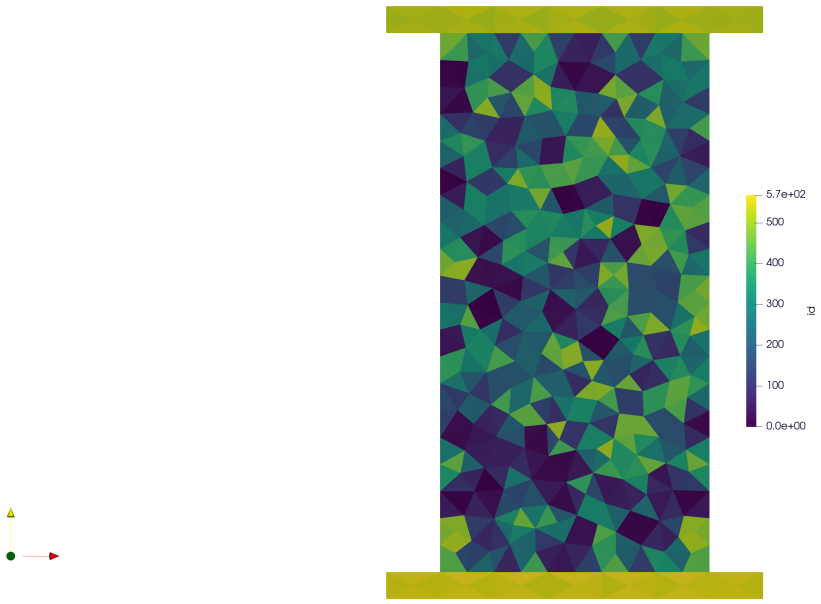
```
of.geometry.mesh delaunay

# 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

#################################### 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
of.boundary.nodal.velocity 'down' x 0.0 y 0.05

#################################### 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
```

2.9.3 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).

1.0 Tutorial Prerequistes

The following files are needed to follow along the tutorial:

- Job-3.inp (click to download from GitHub)

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]
```

3.0 Full Script

- UCS_test_Gmsh.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 parallization will be automatically␣
↪turned on
# when you use the following command, otherwise the serilization will be used by default
of.set.omp 15

#################################### 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 #############################
↪#####
```

(continues on next page)

(continued from previous page)

```
# 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
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
```

2.9.4 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.

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'
```

2.0 Run the Program

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

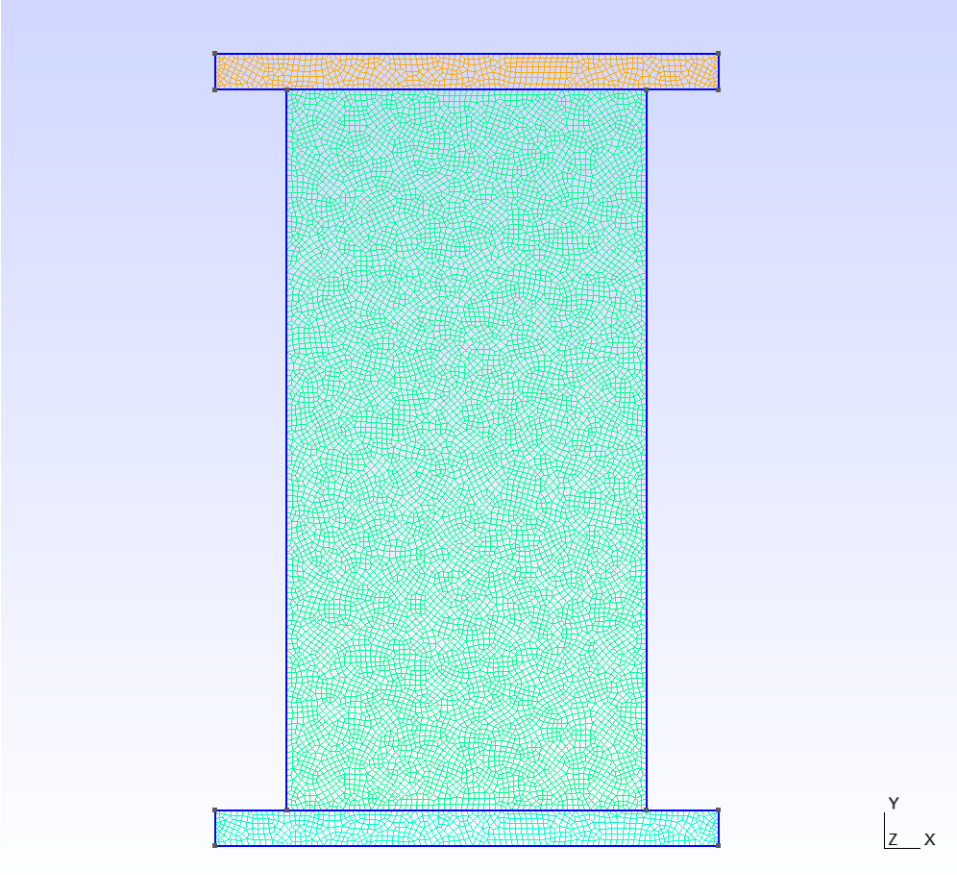


Fig. 23: Figure 1: Uniaxial Compression Test Mesh with Quadrangle Elements

The program will also show the mesh has quadrilateral 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. 24: Figure 2: Uniaxial Compression Test Mesh with Quadrangle Elements

3.0 Full Script

- UCS_test_quad.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 parallization will be␣
↪automatically turned on
# when you use the following command, otherwise the serilization 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 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
# combine the delaunay to quadrangle element types
```

(continues on next page)

(continued from previous page)

```
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
# 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.
```

(continues on next page)

(continued from previous page)

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

2.9.5 Tutorial 4: Brazilian Disc Test

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

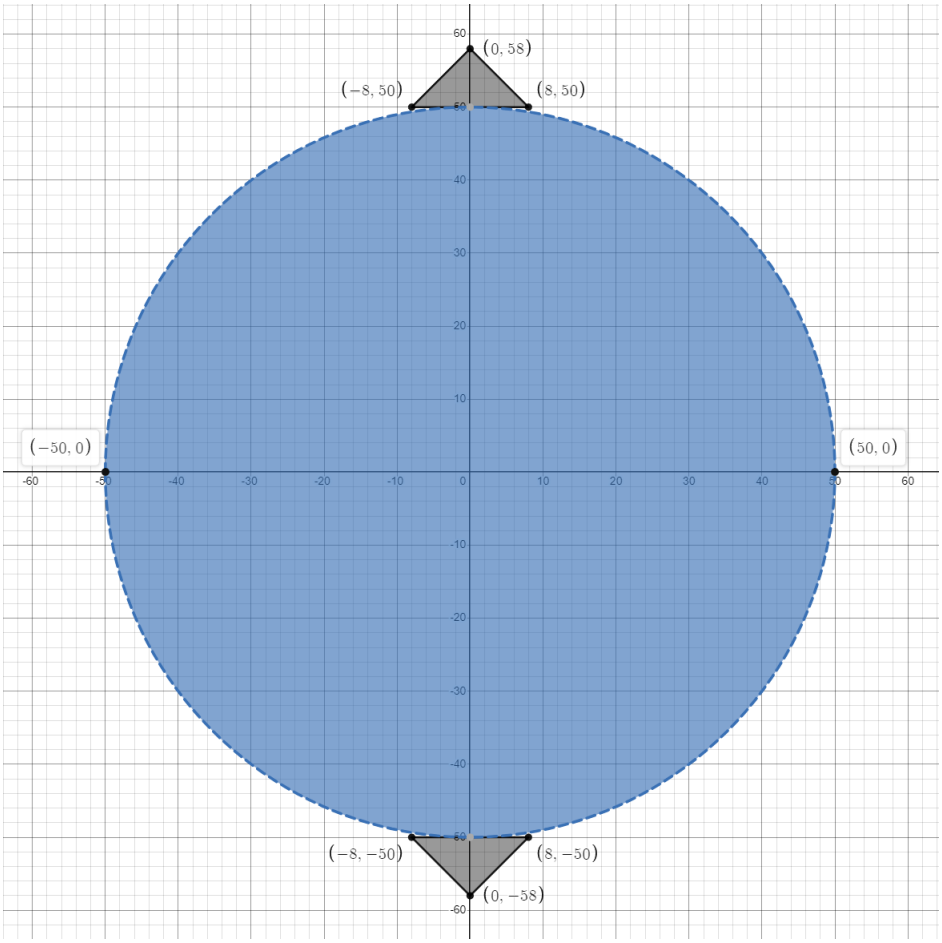


Fig. 25: Figure 1: Geometry of the Brazilian disc test

1.0 Tutorial Prerequistes

The following files are needed to follow along the tutorial:

- example_br.geo (click to download from GitHub)

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.

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

The material properties of this model is shown as the table below:

Parameter	Value
Continuum Triangular Elements - Rock	
model	elastic
density (kg/m^3)	2700
E (Pa)	30e9
ν	0.3
damp	0.6
Continuum Triangular Elements - Platens	
model	rigid
density (kg/m^3)	7000
Cohesive Material Properties	
model	EM
tension (Pa)	1e6
cohesion (Pa)	3e6
friction	0.3
GI (J/m^2)	10
GII (J/m^2)	50
Contact Material Properties	
model	MC
friction	0.3

Note: Check *element materal*, *cohesive materal*, *contact materal* to see more materials.

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 the 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.

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.

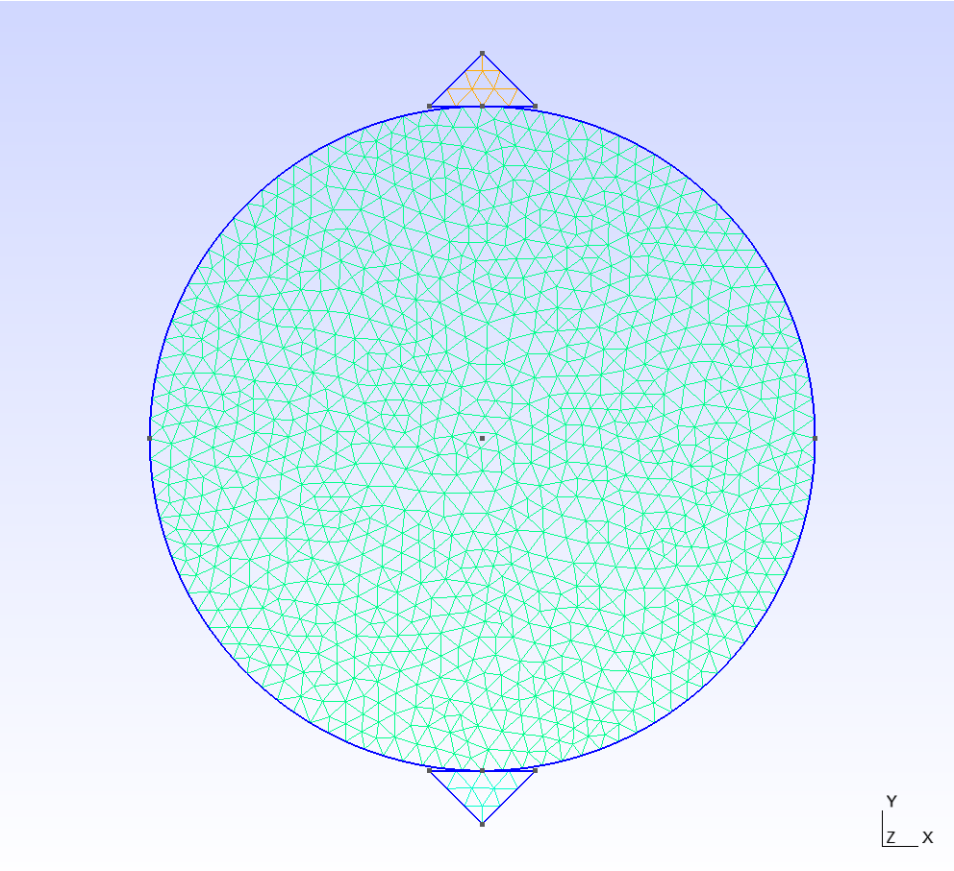


Fig. 26: Figure 2: Mesh created by Gmsh

5.0 Results

Wait for updates.

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 parallization will be␣
↪automatically turned on
# when you use the following command, otherwise the serilization will be used␣
↪by default
of.set.omp 15

#################################### 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
```

(continues on next page)

(continued from previous page)

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

2.9.6 Tutorial 5: Apply In-situ Stress

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

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

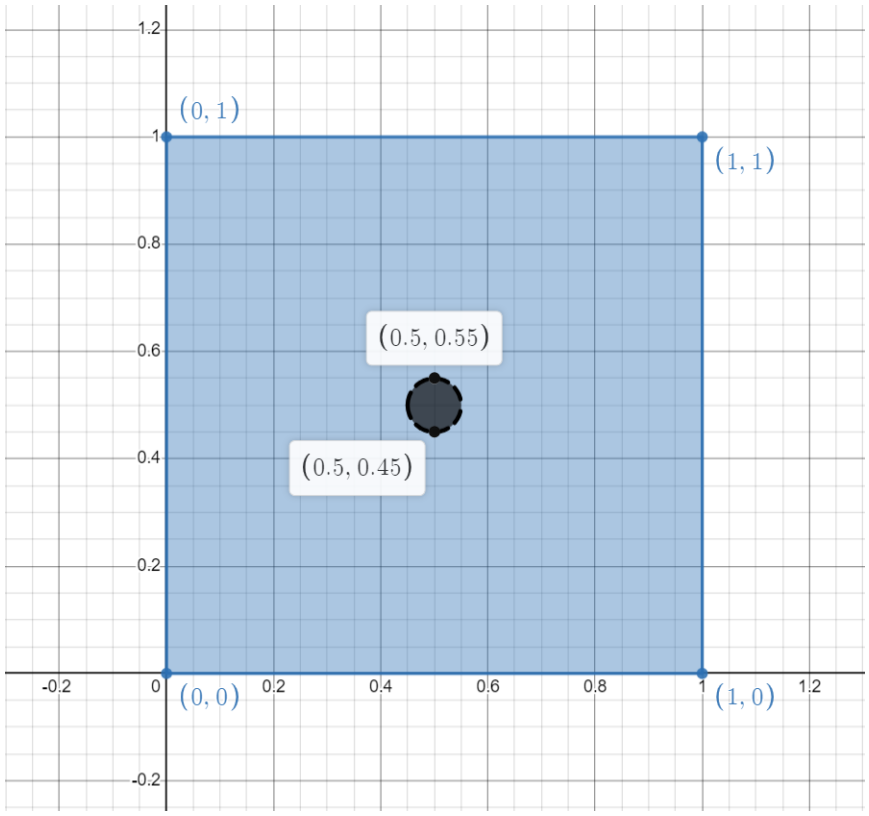


Fig. 27: Figure 1: Geometry of the Rock and Excavation

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

The material properties of this model is shown as the table below:

Parameter	Value
Continuum Triangular Elements	
model	elastic
density (kg/m^3)	2700
E (Pa)	30e9
ν	0.3
damp	1.0
Contact Material Properties	
model	MC
friction	0.3

Note: Check *element materal*, and *contact materal* to see more materials.

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 equlibrium of in-situ stresses**

Parameter	Value
Cohesive Material Properties	
model	EM
tension (Pa)	1e6
cohesion (Pa)	20e6
friction	0.8
GI (J/m^2)	2
GII (J/m^2)	40

```
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.

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.

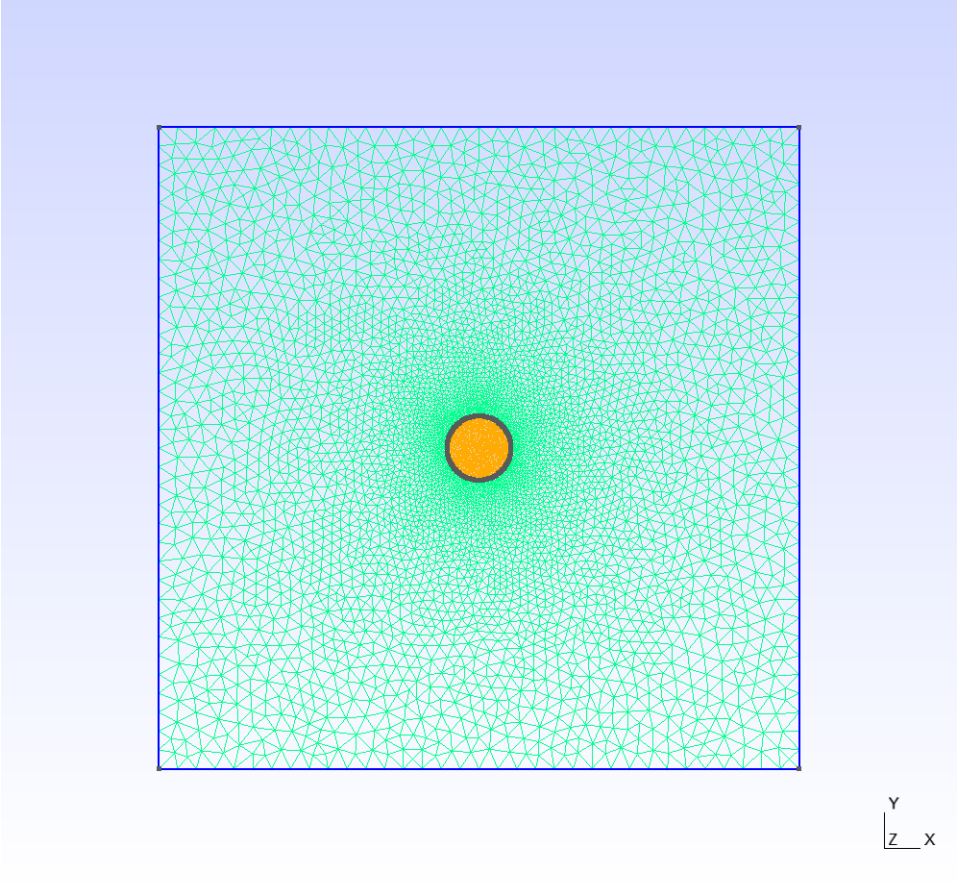


Fig. 28: Figure 2: Mesh created by Gmsh

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

```
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. 29: Figure 3: Check the mesh setting from command window

4.0 Results

Wait for updates.

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 parallization will be␣
↪automatically turned on
# when you use the following command, otherwise the serilization 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
```

(continues on next page)

(continued from previous page)

```
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

# 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 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 ############################
↪##############

# 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 equlibrium 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'
```

(continues on next page)

(continued from previous page)

```
# 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
```

2.9.7 Tutorial 6: Hydro Seepage

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

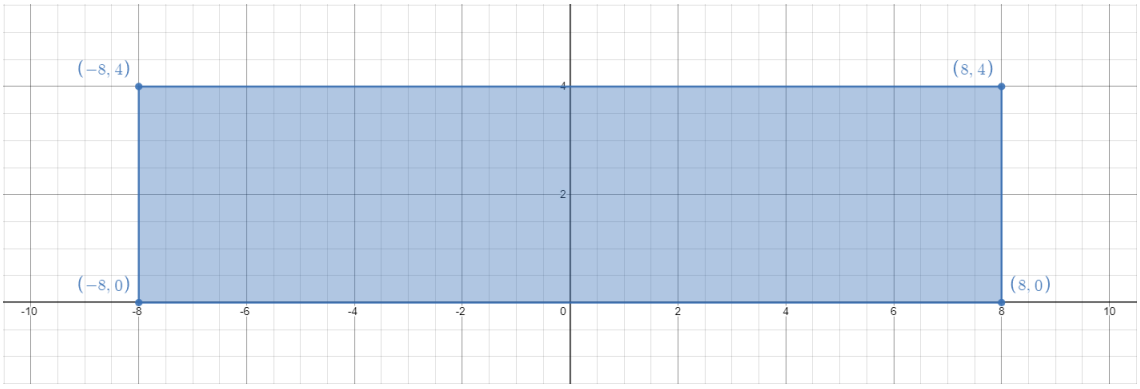


Fig. 30: Figure 1: Geometry of a Dam Model

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.

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

Parameter	Value
Continuum Triangular Elements	
model	elastic
density (kg/m^3)	2700
E (Pa)	5e8
ν	0.2
Contact Material Properties	
model	MC
friction	0.5
Hydraulic Properties	
permeability	2e-8
Biot_K (Pa)	22-9
Biot_c	0.1
prosity	0.2

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' permiability 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.

3.0 Run the Program

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

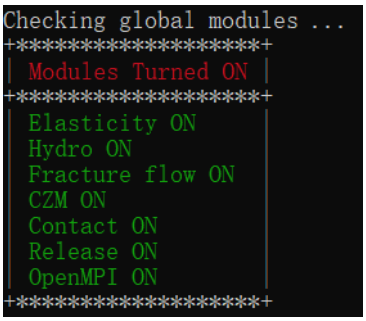


Fig. 31: Figure 2: Applied Modules

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-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        | Mohr-Coulomb law | default | kn = 5.56 GPa, ks = 4.17 GPa              |
|    | (default)               |                 |         | 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. 32: 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. 33: Figure 4: Hydro Boundary Conditions

4.0 Results

Wait for updates.

5.0 Full Script

- Hydro_Seepage.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␣
```

(continues on next page)

(continued from previous page)

```
↪your input file
# will be created by default
of.set.result "result"

# Set the number of cores you want to use, the parallization will be␣
↪automatically turned on
# when you use the following command, otherwise the serilization 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 ############################
↪#############
# 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 ############################
↪###############
```

(continues on next page)

(continued from previous page)

```
# 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
```

2.9.8 Tutorial 7: Hydro Fracture Flow

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

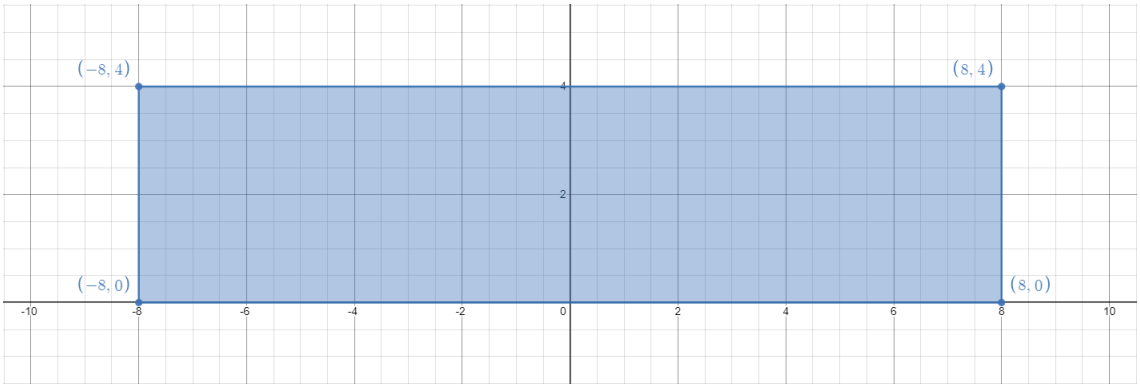


Fig. 34: Figure 1: Geometry of the uniaxial compression strength test

1.0 Code

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

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

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

2. **Assign the material properties to CZM.**

Parameter	Value
Contact Material Properties	
model	MC
friction	0.5

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

3. **Set fracture flow parameters**

Parameter	Value
Hydro fluid material	
density (kg/m^3)	1000.0
K (Pa)	20e9
viscosity (Pa)	1e-3
Hydro Fracture material	
a0 (m)	1e-4
amin (m)	1e-4
amax (m)	3e-4
power	3.0
b	1.0

```
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
```

2.0 Run the Program

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

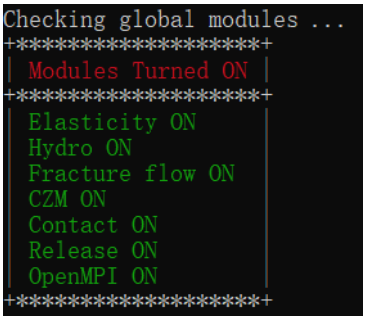


Fig. 35: Figure 2: Applied Modules

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

Hydro boundary conditions.

```
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        | Mohr-Coulomb law      | default | kn = 5.56 GPa, ks = 4.17 GPa                  |
|    | (default)               |                       |         | 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. 36: Figure 3: Hydro Fracture Materials

```
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. 37: Figure 4: Hydro Boundary Conditions

3.0 Results

Wait for updates.

4.0 Full Script

- Hydro_Fracture_Flow.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 number of cores you want to use, the parallization will be␣
↪automatically turned on
# when you use the following command, otherwise the serilization will be used␣
↪by default
of.set.omp 15
```

(continues on next page)

(continued from previous page)

```
##################################### 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

################################### 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
```

(continues on next page)

(continued from previous page)

```
of.step 60000
# finalize the model and clear the memory
of.finalize
```

2.9.9 Tutorial 8: Thermal Flux

Thermal module will be reviewed in this tutorial.

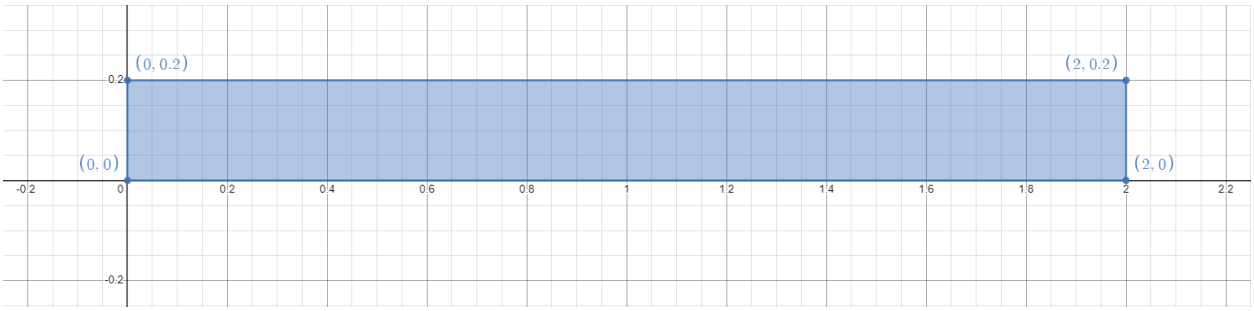


Fig. 38: Figure 1: Geometry of the uniaxial compression strength test

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.

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

Parameter	Value
Continuum Triangular Elements	
model	elastic
density (kg/m^3)	2700
E (Pa)	5e8
ν	0.2
Cohesive Material Properties	
model	EM
tension (Pa)	1e6
cohesion (Pa)	3e6
friction	0.3
GI (J/m^2)	10
GII (J/m^2)	50
heat_exchange ($°C$)	20
Contact Material Properties	
model	MC
friction	0.3
conductivity	200
Thermal Properties	
conductivity	160
spec-heat	0.2
expansion	1e-6

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.

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 thermal module is ON in this model.

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.

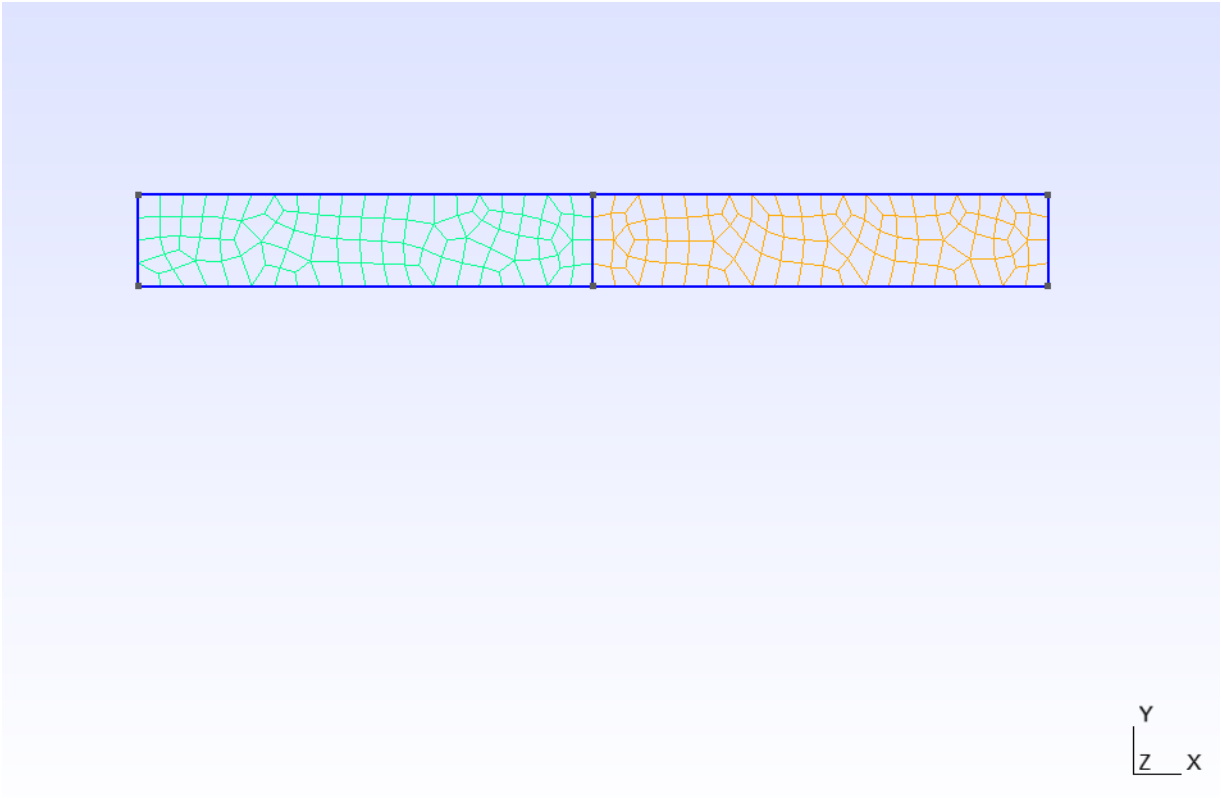


Fig. 39: Figure 2: Mesh created by Gmsh

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

Fig. 40: Figure 3: Global modules

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 = 0.00 GPa pt = 0.00 GPa tension = 1.00 MPa cohesion = 3.00 MPa GI = 10.00 J/m^2 GII = 50.00 J/m^2 friction = 0.30 h.T = 20.00 W/(m^2.C)
2	Contact material (default)	Mohr-Coulomb law	default	kn = 5.56 GPa,ks = 4.17 GPa friction = 0.30 conductivity = 200.00 W/(m.C)
3	Thermal matrix material	Thermal transportation	default	conductivity = 160.00 W/(m.C) specific heat = 0.20 J/(kg.C) alpha = 0.00 heat exchange = 0.00 W/(m^2.C)

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	right	x fixed y fixed
1	Node boundary	Velocity	up	x fixed y fixed
2	Node boundary	Velocity	down	x fixed y fixed
3	Thermal boundary	Temperature	left	T = 50.00 C
4	Thermal boundary	Temperature	right	T = 0.00 C

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

4.0 Results

Wait for updates.

5.0 Full Script

- Thermal_Flux.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 number of cores you want to use, the parallization will be␣
↪automatically turned on
# when you use the following command, otherwise the serilization 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
```

(continues on next page)

(continued from previous page)

```
of.mat.cohesive 'default' EM ten 1e6 coh 3e6 fric 0.3 GI 10 GII 50 heat_
↪exchage 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
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 output ##############################
↪##############
# Set the interval of writing ParaView results.
of.history.pv.interval 500
of.history.pv.field default
of.history.pv.fracture default

#################################### execute model ###########################
↪##############
# 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
```

2.9.10 Tutorial 9: Grain-based Model (GBM)

Grain based modeling on UCS test.

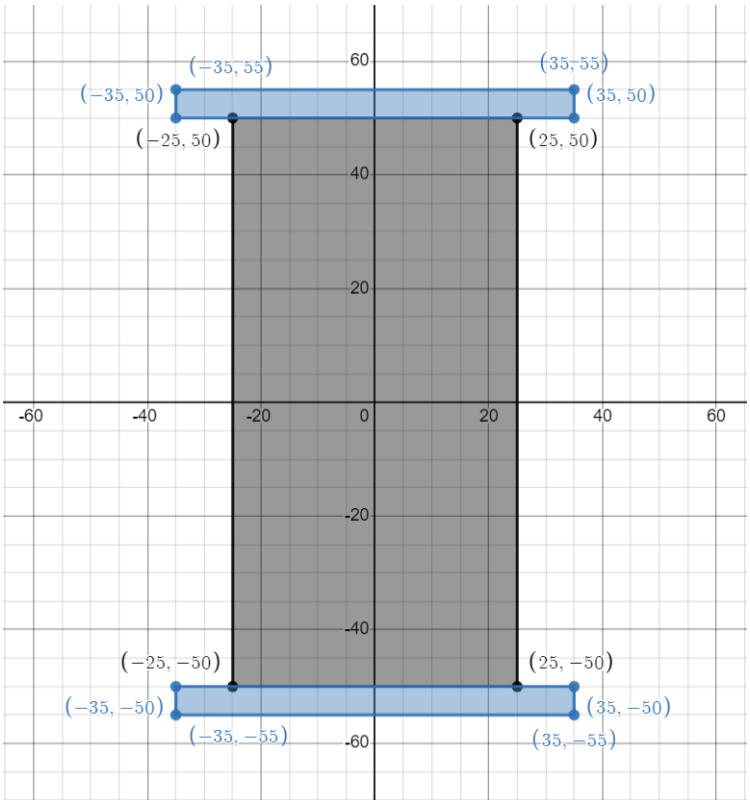


Fig. 43: Figure 1: Geometry of the uniaxial compression strength test

1.0 Tutorial Prerequistes

The following files are needed to follow along the tutorial:

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

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.

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 1000000
```

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.

4.0 Run the Program

Wait for updates.

5.0 Results

Wait for updates.

6.0 Full Script

- GBM.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
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 parallization will be␣
↪automatically turned on
# when you use the following command, otherwise the serilization will be used␣
↪by default
of.set.omp 15

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

(continues on next page)

(continued from previous page)

```
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

# 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'
```

(continues on next page)

(continued from previous page)

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

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

2.9.11 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.

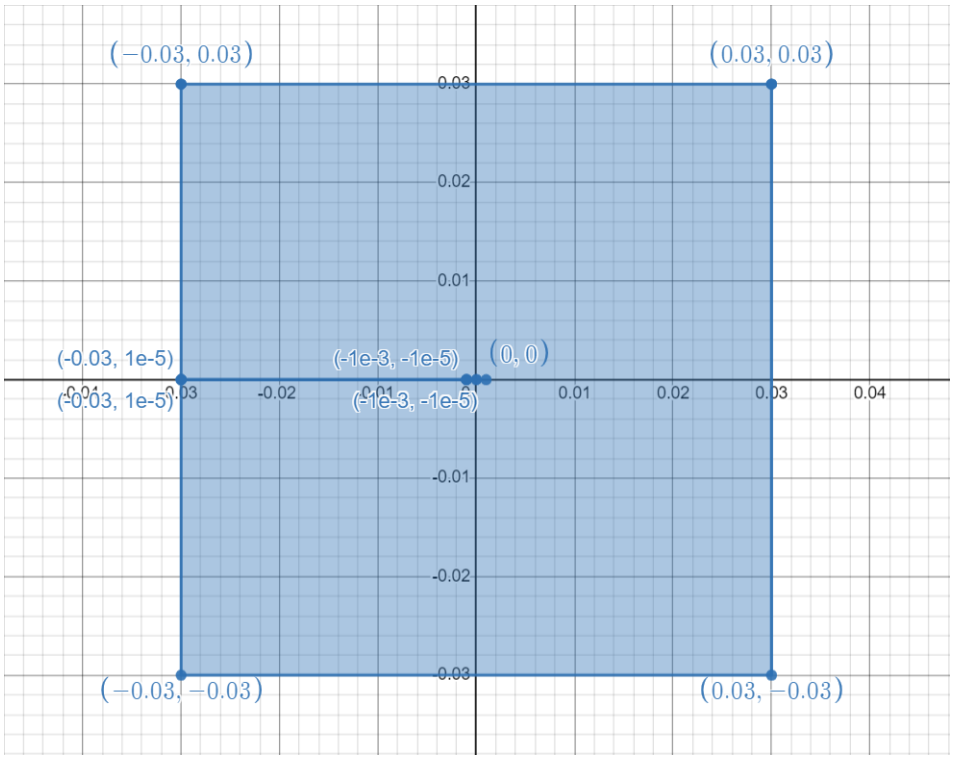


Fig. 44: Figure 1: Geometry of the uniaxial compression strength test

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.

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. **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 'bottow_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 'bottow_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.

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 phase field module is ON in this model.

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.

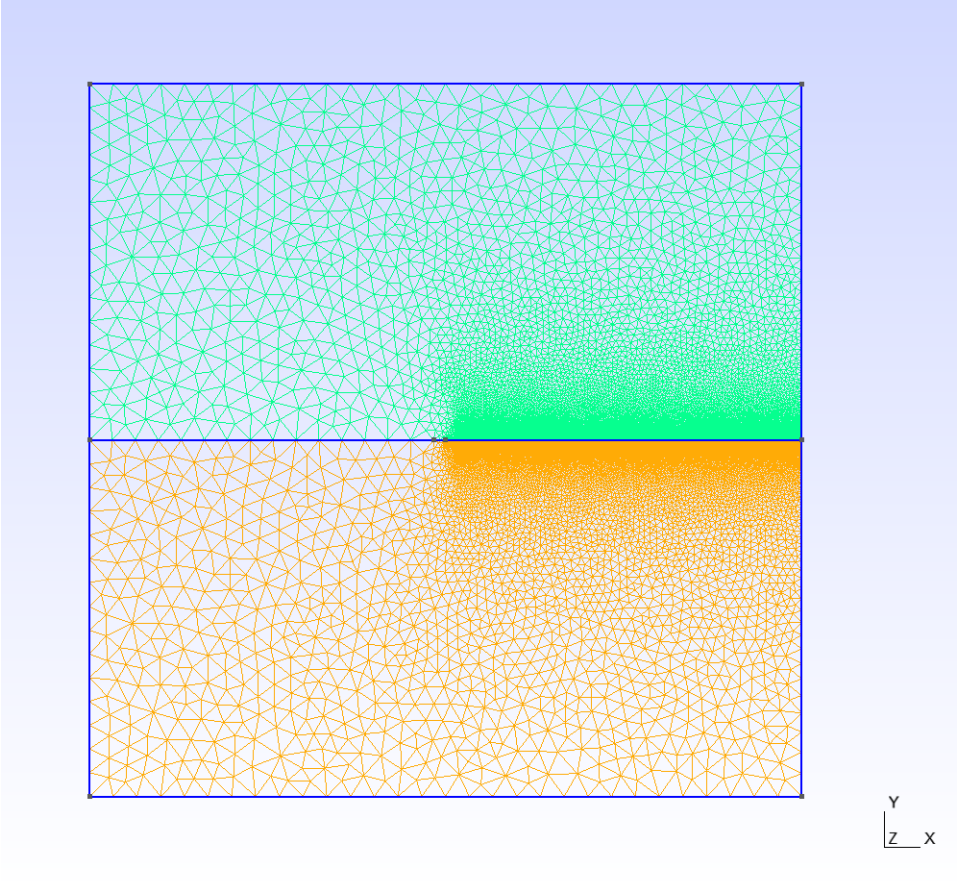


Fig. 45: Figure 2: Mesh created by Gmsh

```
Checking global modules ...
+********************+
| Modules Turned ON  |
+********************+
| Elasticity ON      |
| Hydro OFF          |
| CZM OFF            |
| Contact OFF        |
| Phase field ON     |
| Release ON         |
| OpenMPI ON         |
+********************+
```

Fig. 46: Figure 3: Global modules

```
Checking model material information ...
+----+-----------------+-------------------------------+-----------+-------------------------------------+
| ID | Type            | Model type                    | Tags      | Properties                          |
+----+-----------------+-------------------------------+-----------+-------------------------------------+
| 0  | Matrix material | Linear elastic (phasefield)   | plate     | E = 30.00 GPa v = 0.30              |
|    |                 |                               |           | density = 2700.00 kg/m^3 G_c = 200.00|
|    |                 |                               |           | l_c = 0.00 nita = 1.00              |
|    |                 |                               |           | damp = 0.00                         |
+----+-----------------+-------------------------------+-----------+-------------------------------------+
| 1  | Matrix material | Linear elastic (phasefield)   | Mat2      | E = 20.00 GPa v = 0.30              |
|    |                 |                               |           | density = 2700.00 kg/m^3 G_c = 100.00|
|    |                 |                               |           | l_c = 0.00 nita = 1.00              |
|    |                 |                               |           | damp = 0.00                         |
+----+-----------------+-------------------------------+-----------+-------------------------------------+
| 2  | Contact material| Mohr-Coulomb law              | default   | kn = 500.00 GPa,ks = 230.77 GPa     |
|    | (default)       |                               |           | friction = 0.30                     |
+----+-----------------+-------------------------------+-----------+-------------------------------------+
```

Fig. 47: Figure 4: Check the material assignment from command window

4.0 Results

Wait for updates.

5.0 Full Script

- Phase_Field.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"

# 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 parallization will be␣
↪automatically turned on
# when you use the following command, otherwise the serilization will be used␣
↪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]
```

(continues on next page)

(continued from previous page)

```
[-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 'bottow_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'

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

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

################################### execute model ##############################
↪##############
```

(continues on next page)

(continued from previous page)

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

2.9.12 Tutorial A: Paraview

ParaView is an open source post-processing visualization engine users can use to visualize their model results. Users can go to ParaView to download their software. There is an official PraView documentation online. This tutorial will just briefly introduce a few useful functions to view the results from OpenFDEM.

1. **Go to the toolbar > File > Open… and choose the file path of the results and click OK. It will import all results under this category. You may also expand the group and select a single file.**

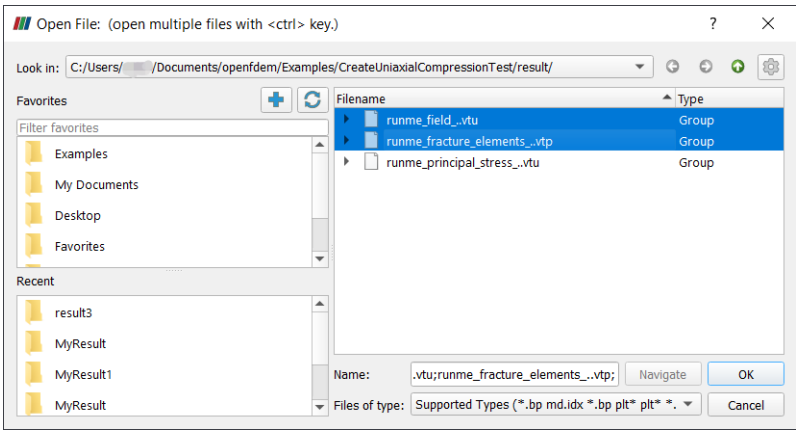


Fig. 48: Figure 1: Import Files to ParaView

2. **Click Apply to show the data.**
3. **If the figure of the result is too small, choose reset to rescale the model.**
4. **Select "runme_field_0.vtu" and change the result to "Stress"**
5. **To go to the next frame, users may use the arrow, select specific frame number, or enter the frame number. In this example, the last frame will be used.**
6. **Select "Clamp and update every timestep" for Automatic Rescale Range Mode.**
7. **Go to the last step of the result. The sample is fractured under compression.**
8. **If the result is not shown in the correct color scheme, click "Rescale to Visible Data Range".**
9. **Select "runme_fracture_elements_0" and choose "Fracture Mode" with type of "Feature Edges" to show the cracking boundaries.**
10. **To only show the sample without platens, users can select the "runme_field_0.vtu" file and use threshold filter in filters.**
11. **To filter the sample out, choose "ele_group_rock" with condition of above upper thresold 1 and apply.**
12. **You may show results with the rock sample only now.**

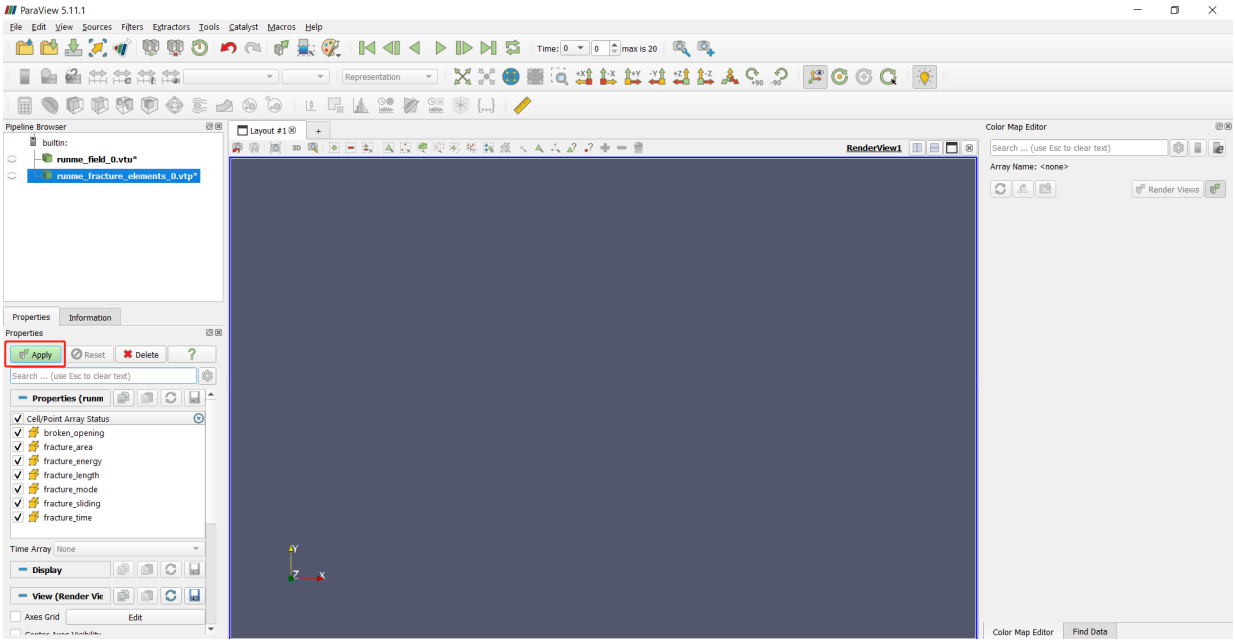


Fig. 49: Figure 2: Apply the data to show

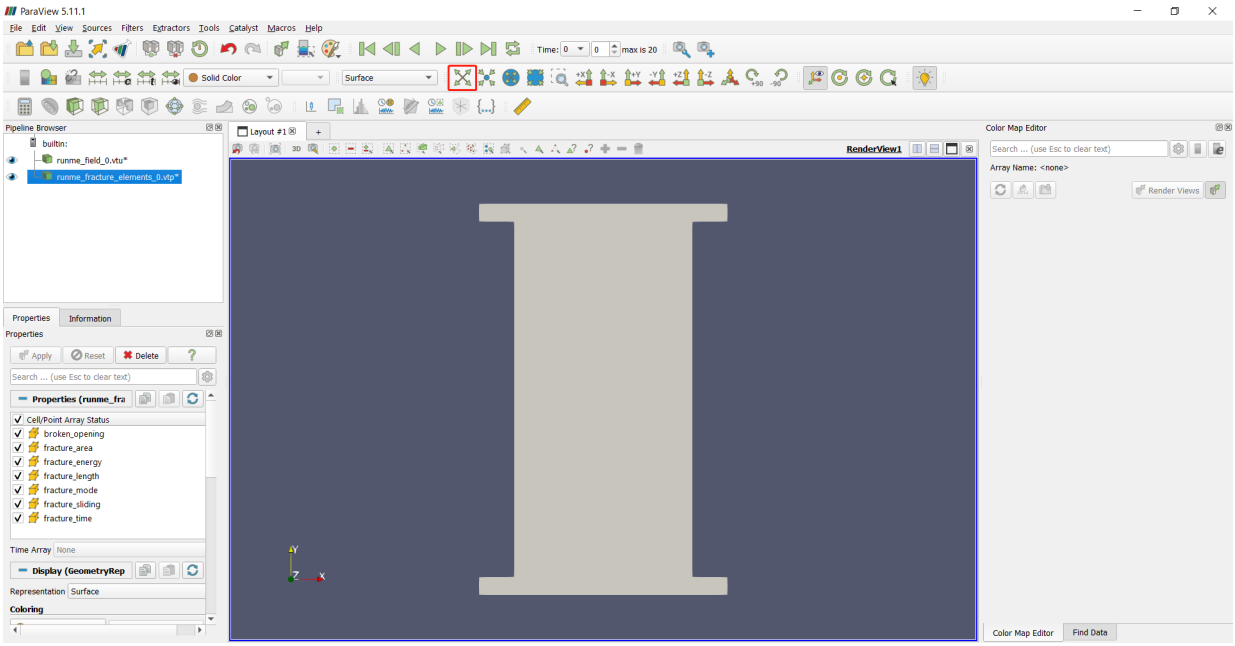


Fig. 50: Figure 3: Reset the view

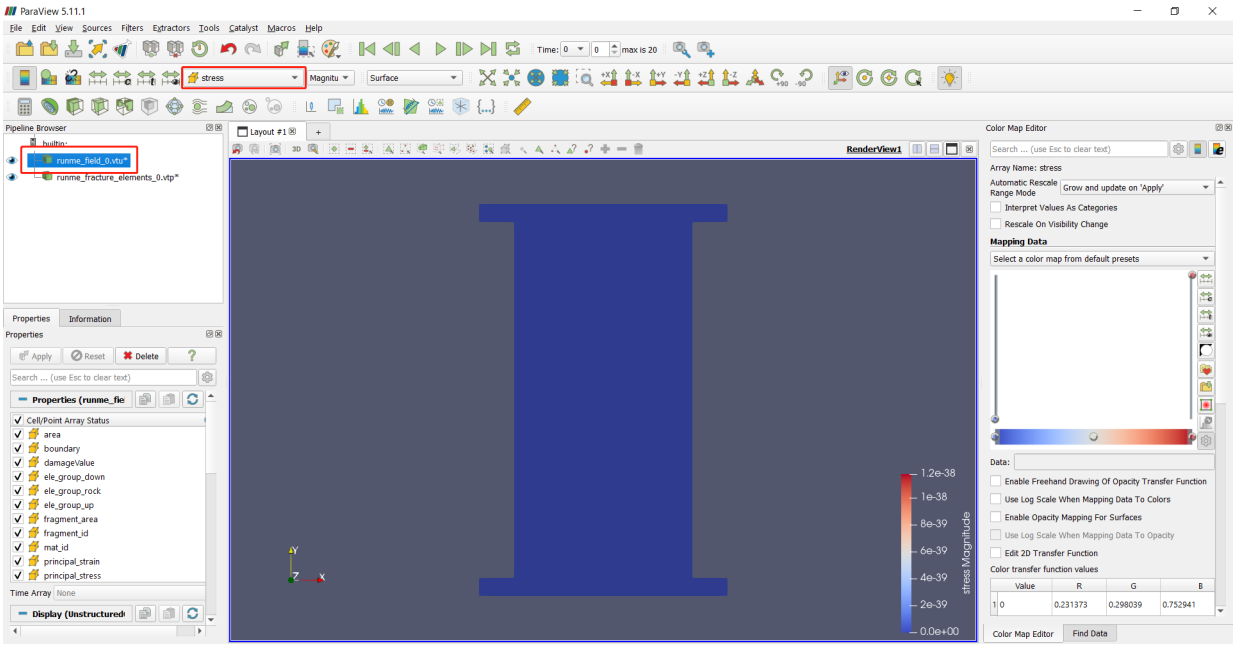


Fig. 51: Figure 4: Show the Stress

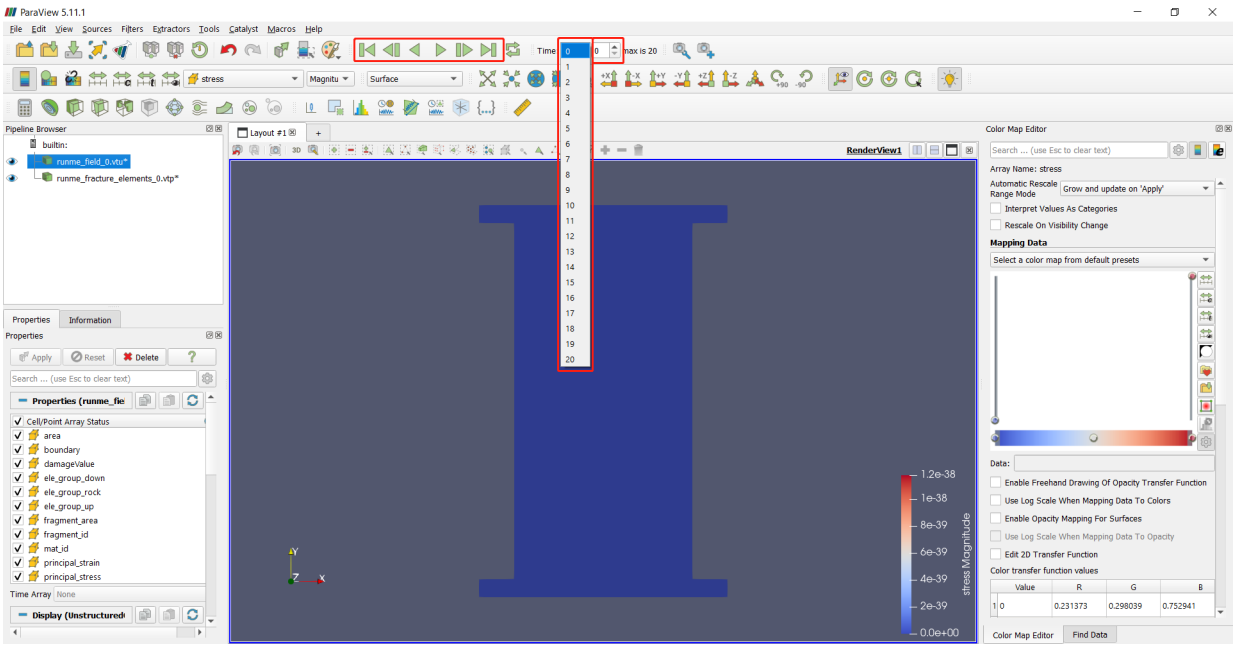


Fig. 52: Figure 4: Show a specific frame

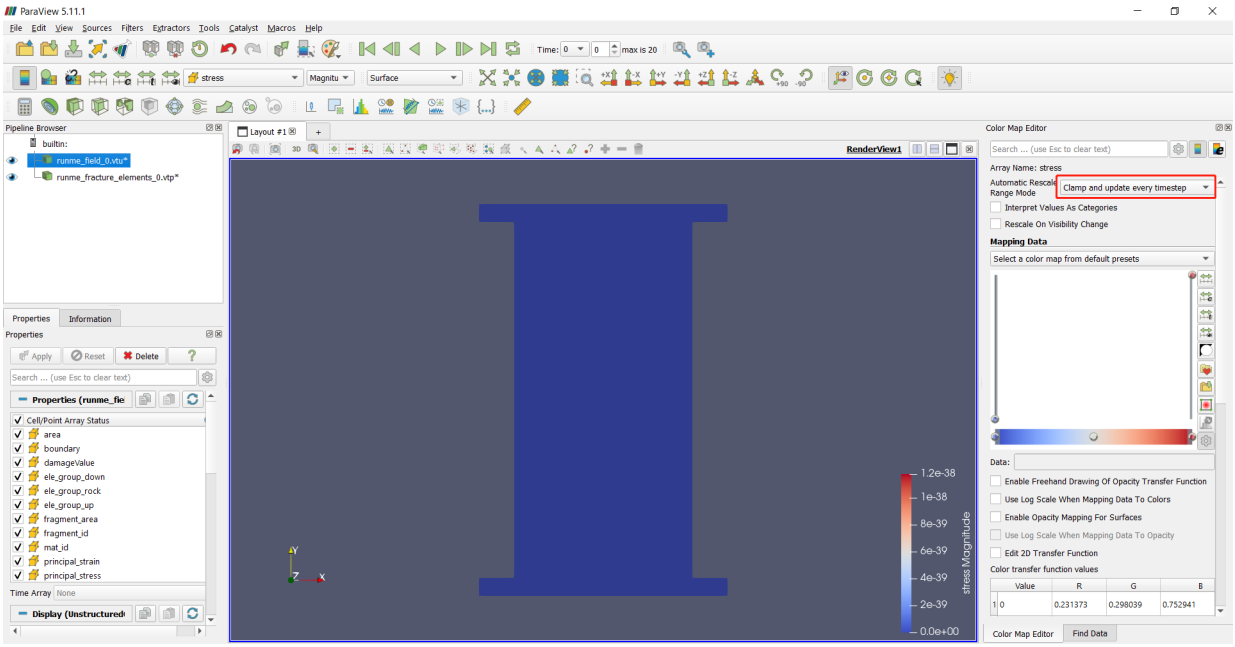


Fig. 53: Figure 5: Clamp the range

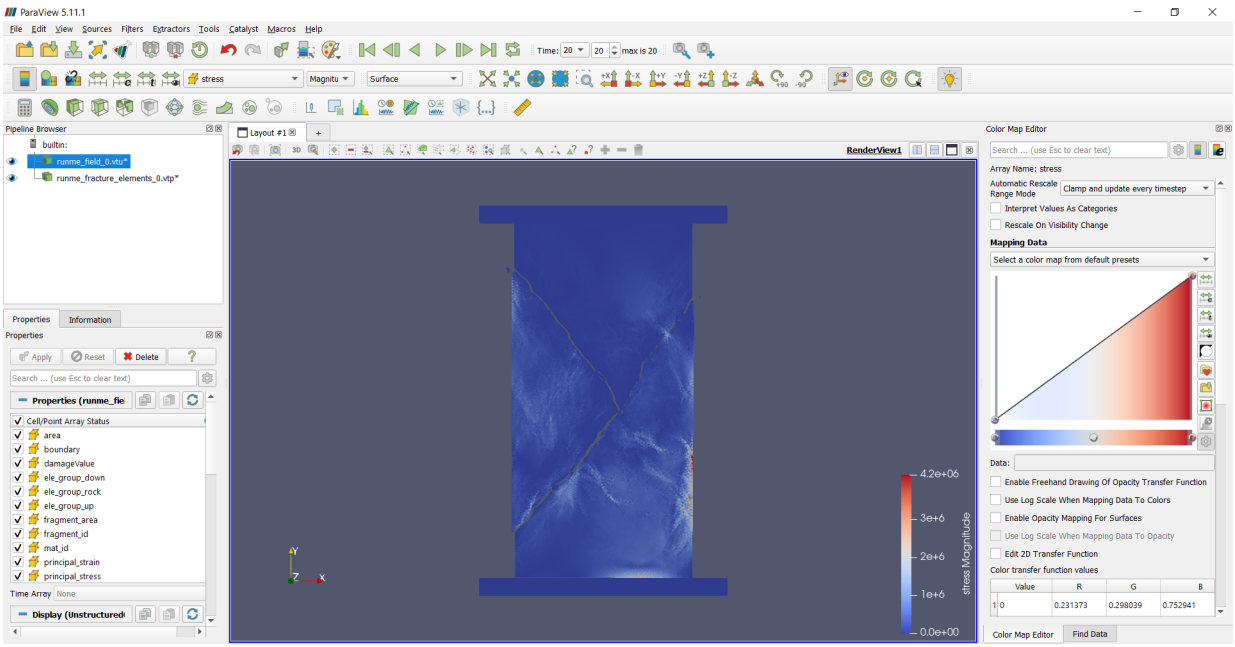


Fig. 54: Figure 6: Last step of the UCS test

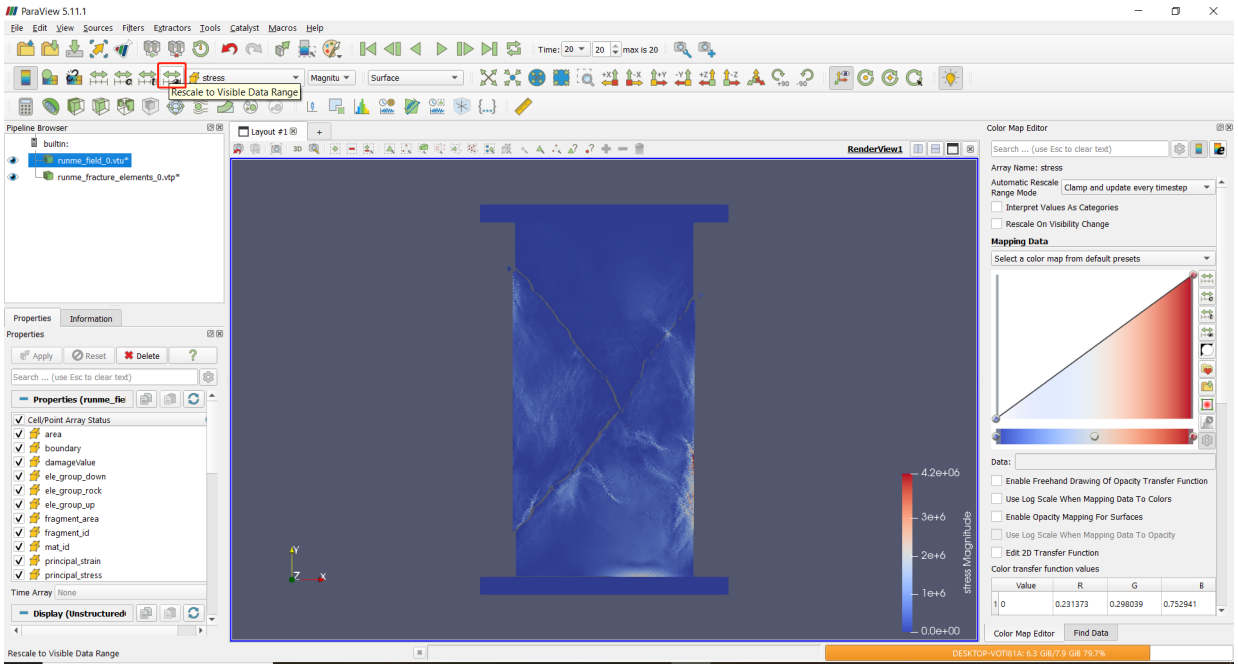


Fig. 55: Figure 7: Rescale to the data range

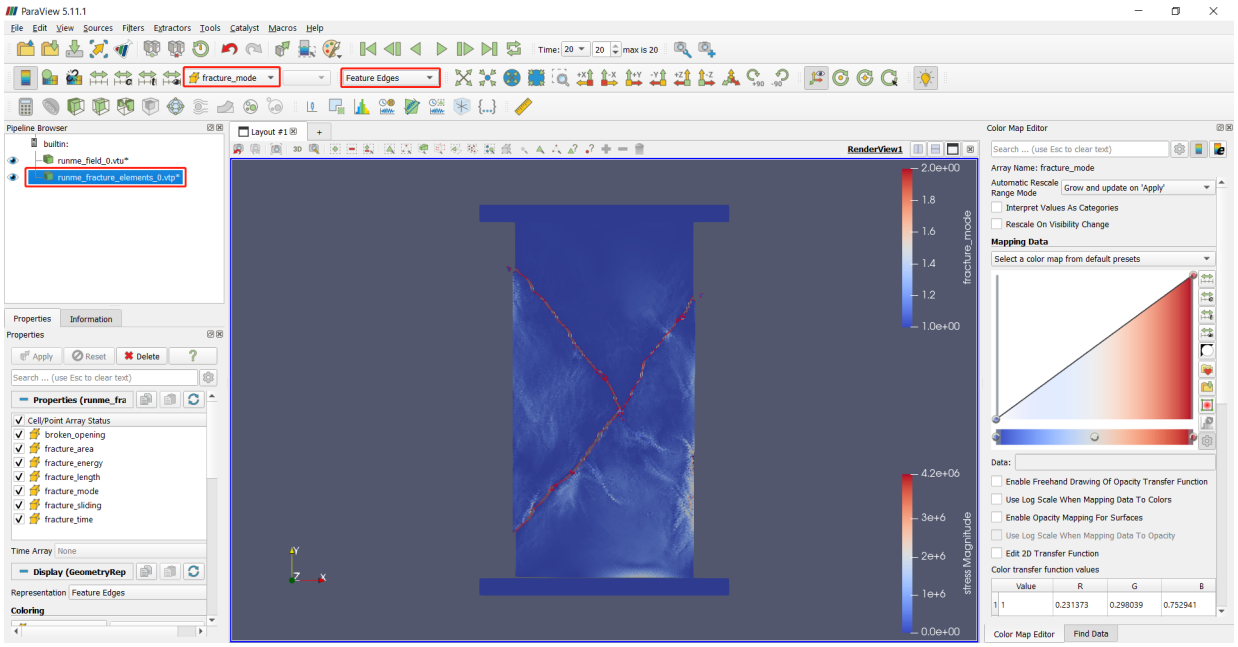


Fig. 56: Figure 8: Fracture mode

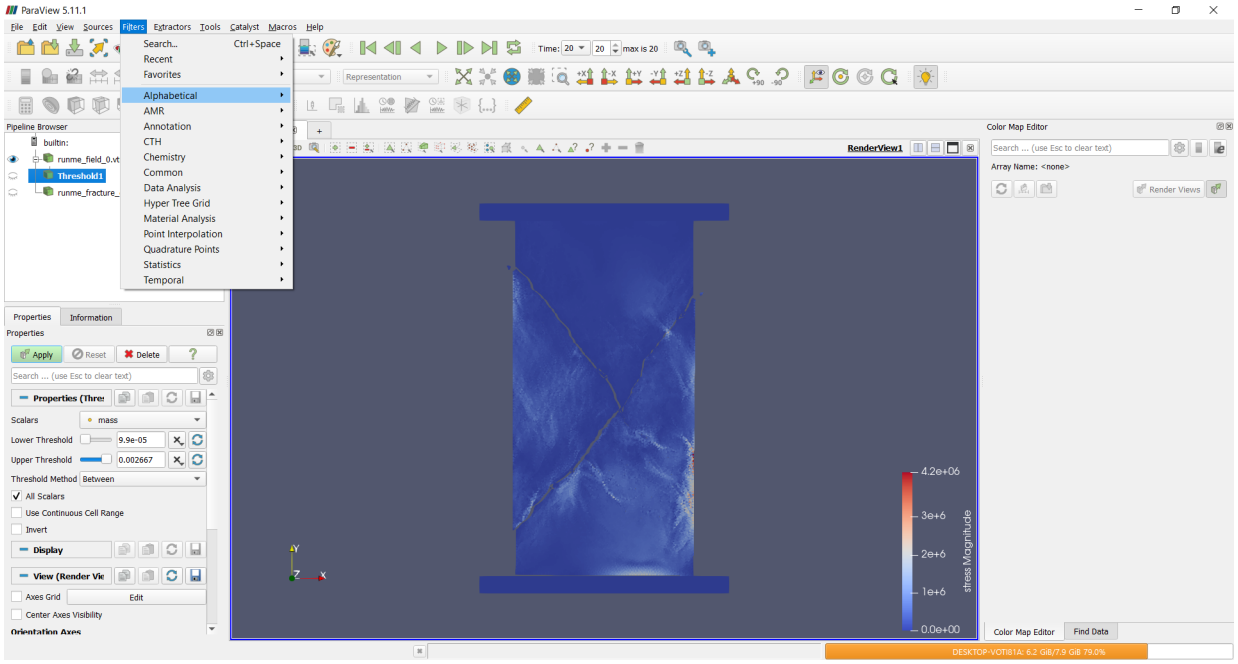


Fig. 57: Figure 9: Filters

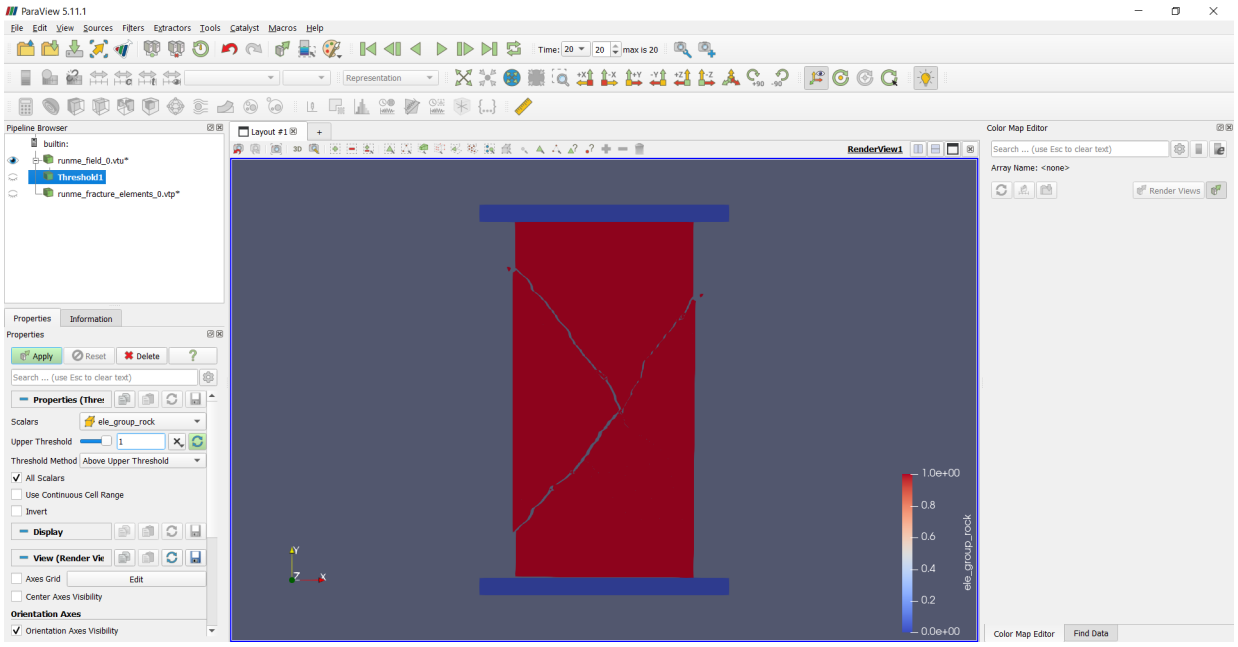


Fig. 58: Figure 10: Threshold

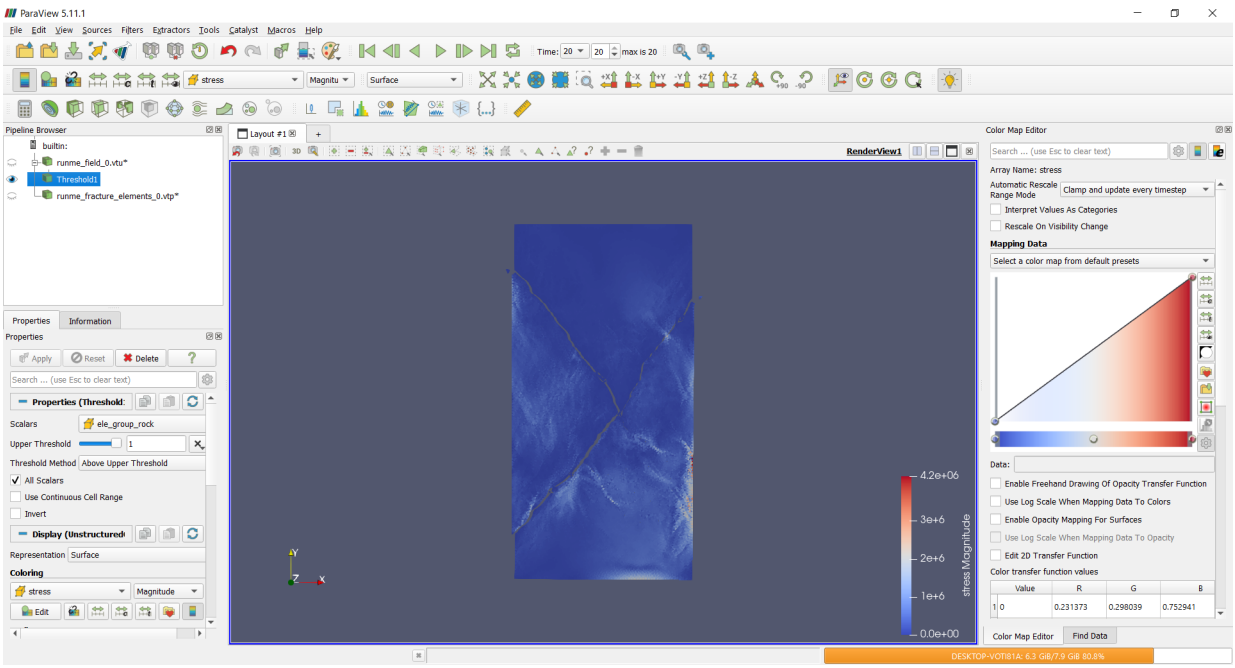
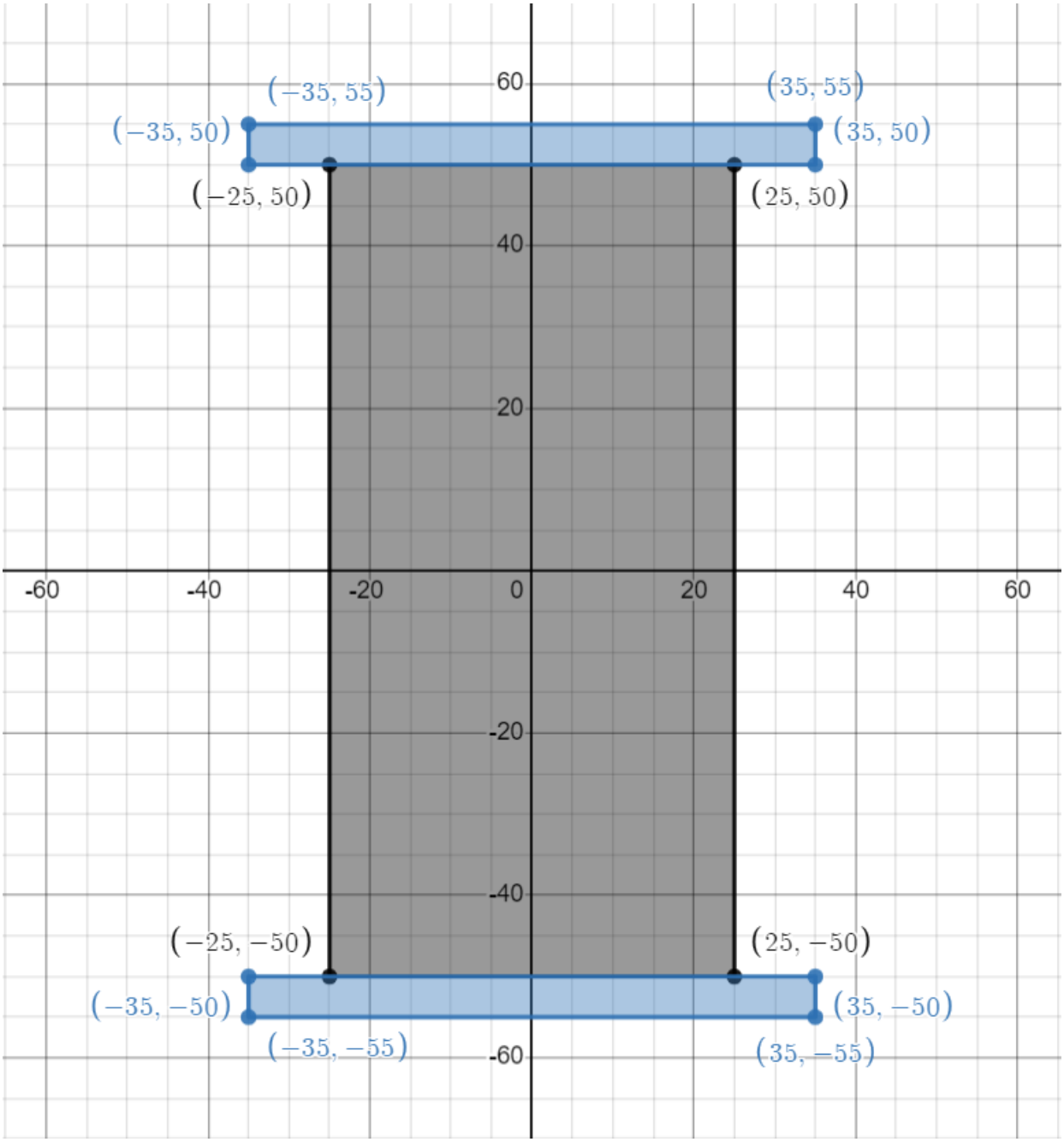


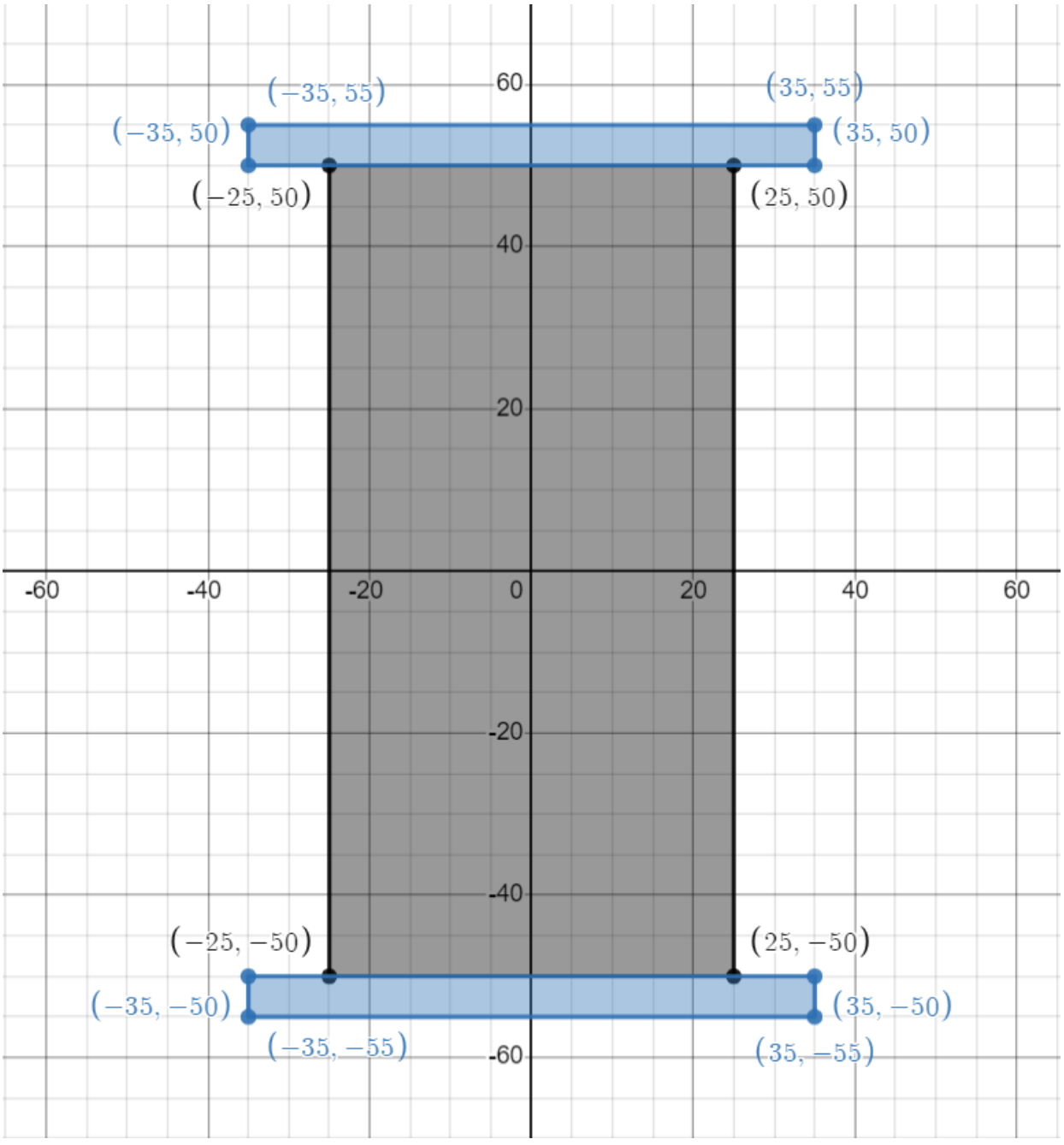
Fig. 59: Figure 11: Result

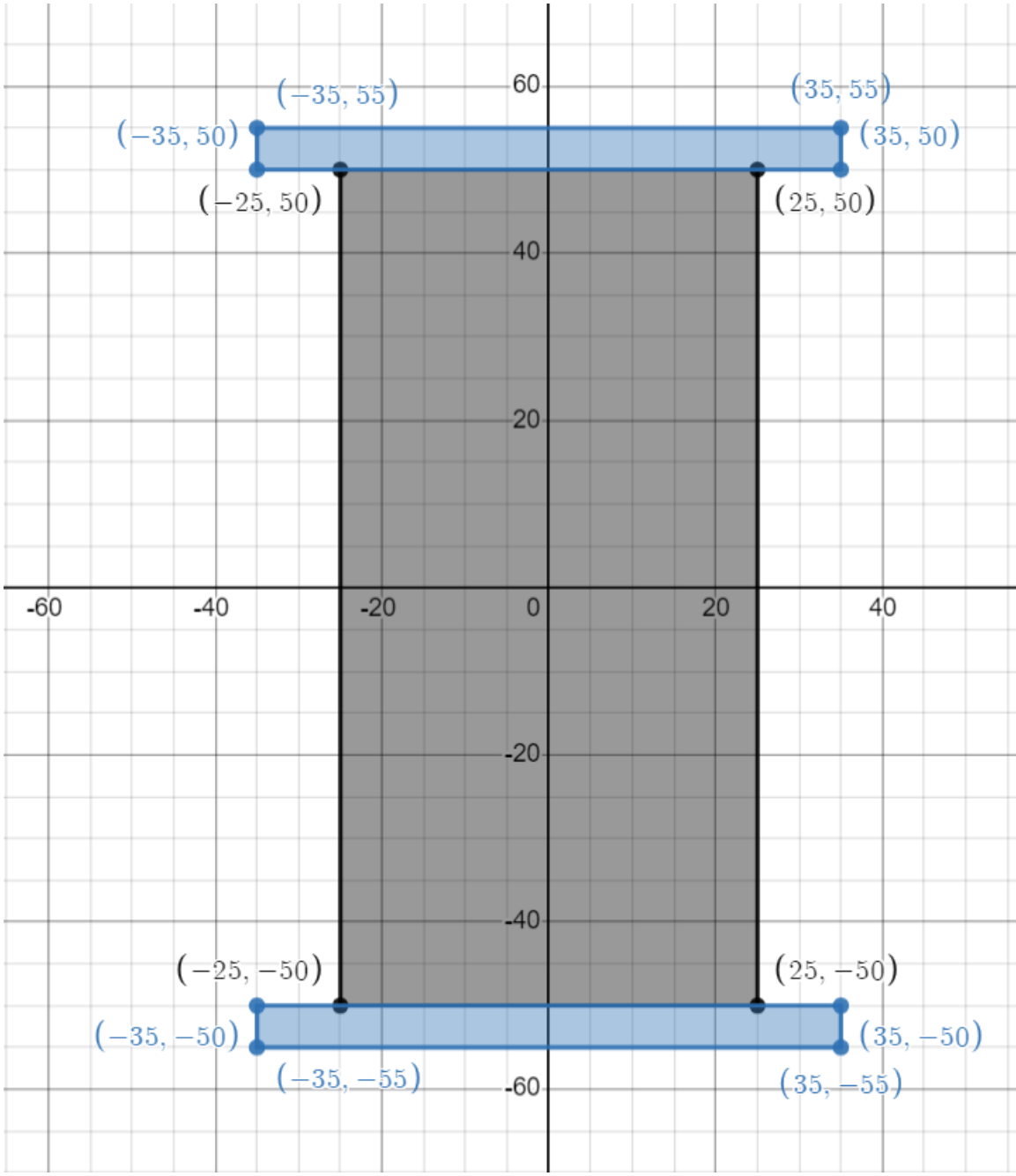
2.9.13 Create the Geometry



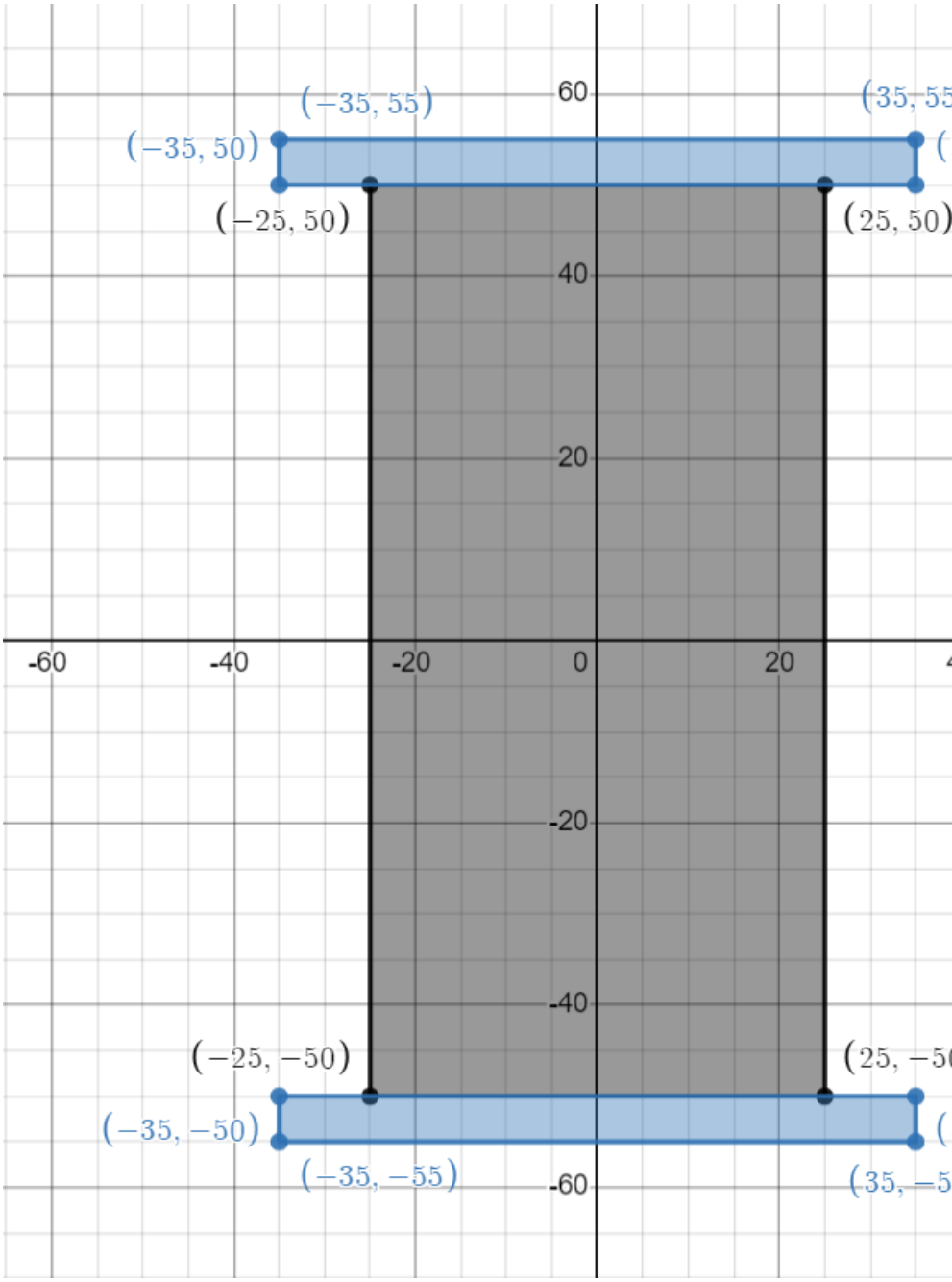
tutorial0_Geometry

2.9.14 Lab Tests

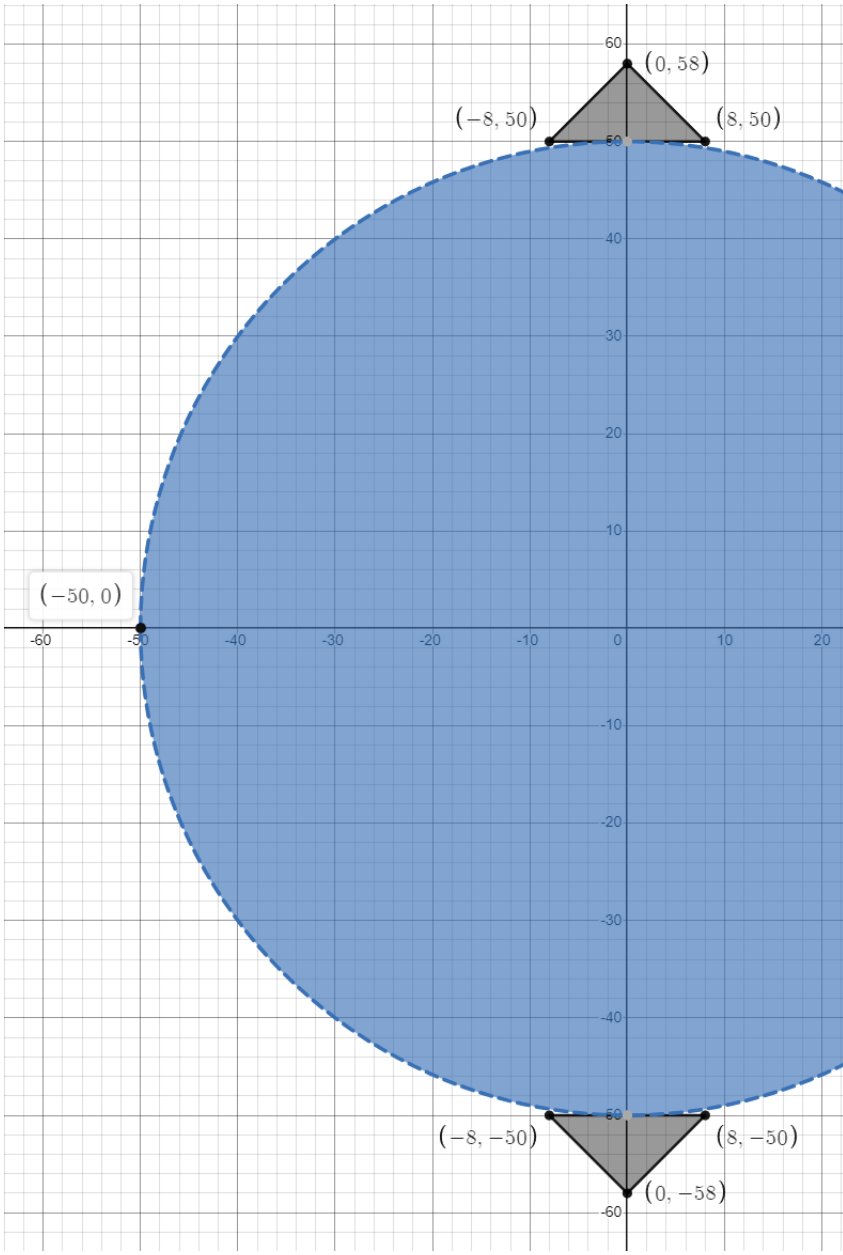




tutorial1_UCS_Test



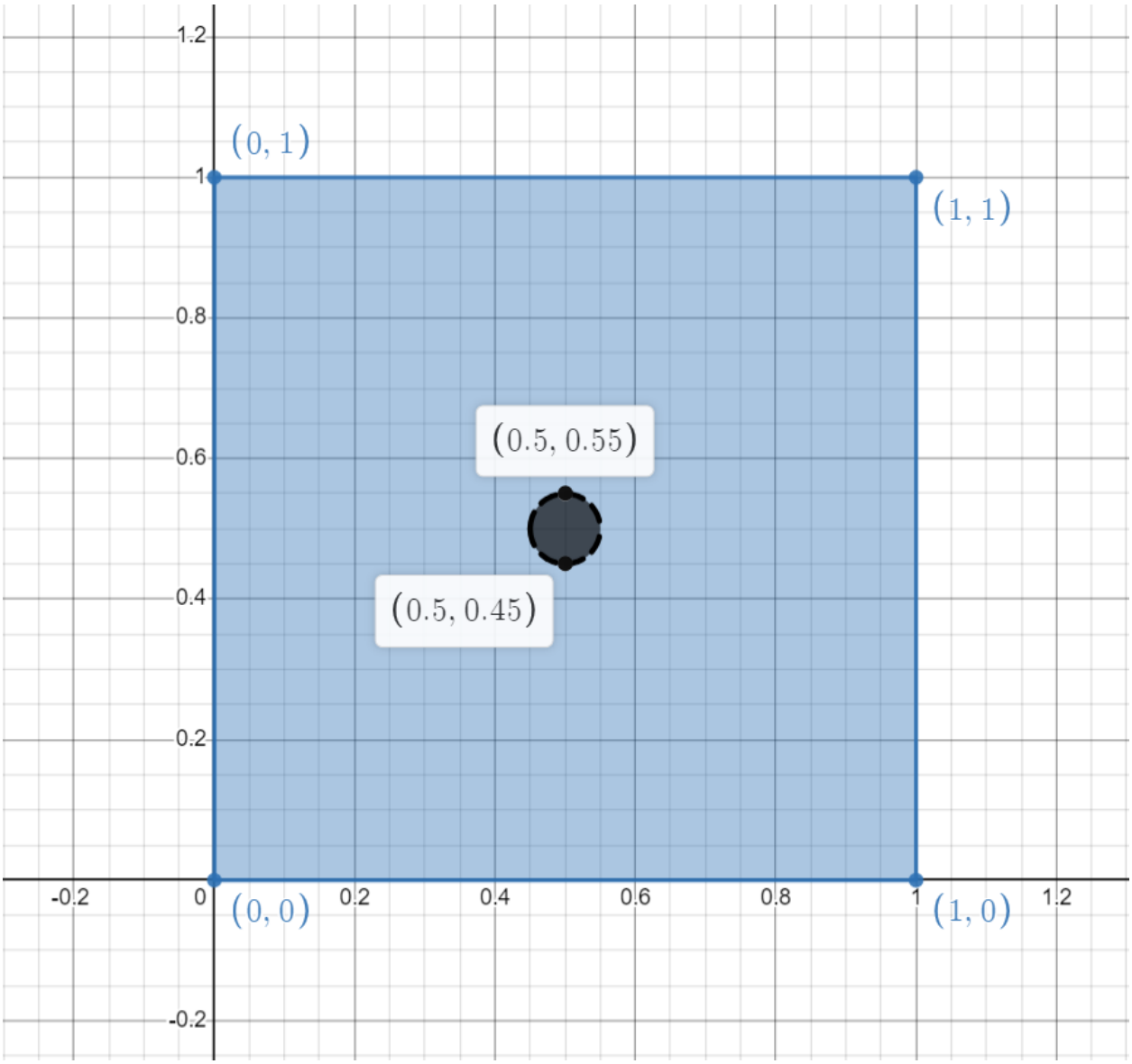
tutorial2_UCS_with_Gmsh



tutorial3_UCS_Test_with_Quadrangle_Elements

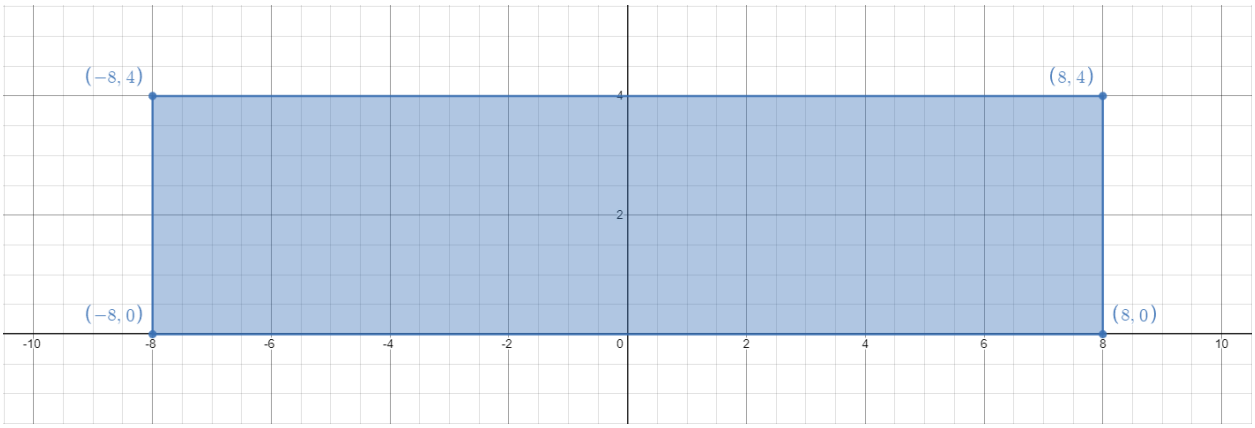
tutorial4_BD_Test

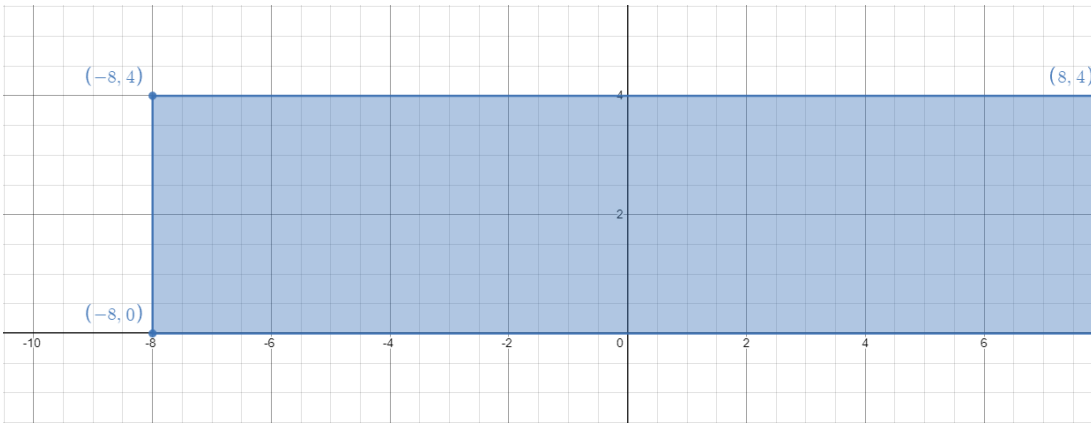
2.9.15 Excavation



tutorial5_In_Situ_Stress

2.9.16 Hydro Module

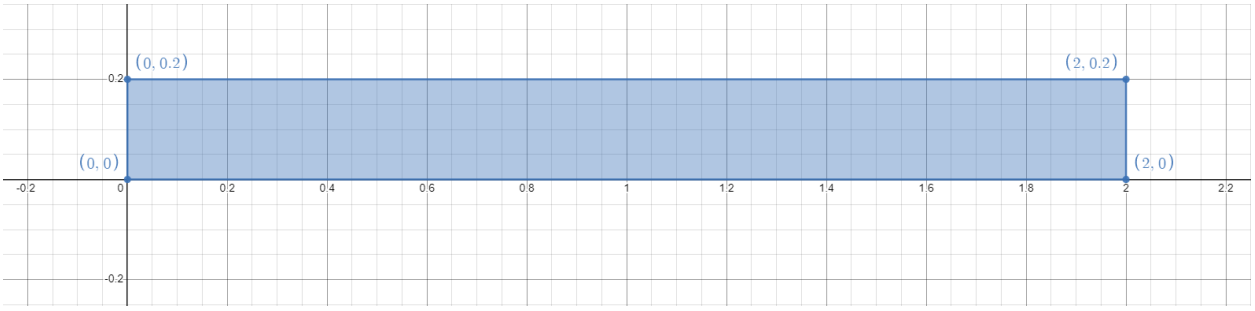




tutorial6_Hydro_Seepage

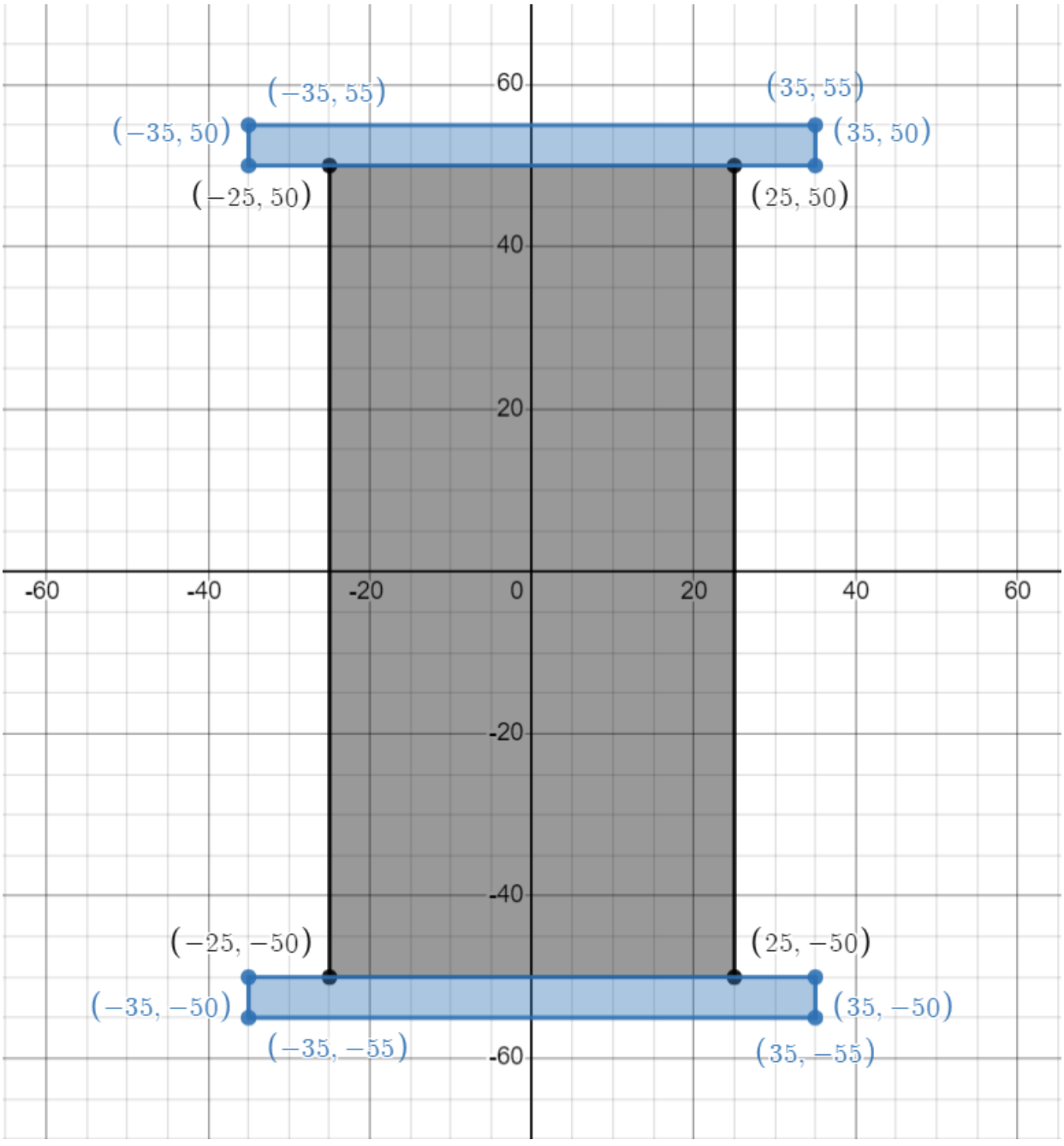
tutorial7_Hydro_Fracture_Flow

2.9.17 Thermal Module



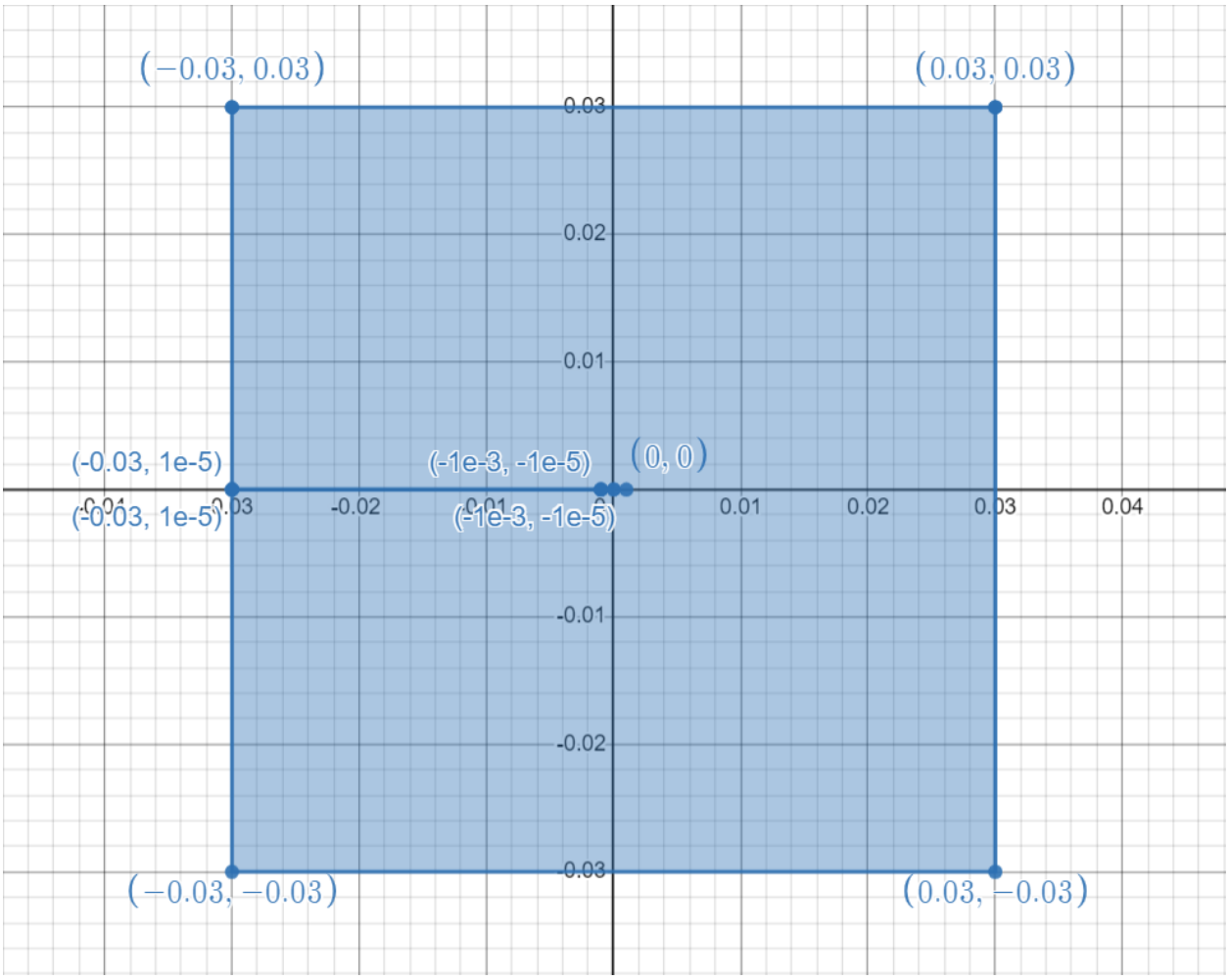
tutorial8_Thermal_Flux

2.9.18 GBM



tutorial9_GBM

2.9.19 Phase Field



tutorial10_Phase_Field

2.10 User Guide

2.10.1 General

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
```

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
```

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.final
```

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”
```

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”
```

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
```

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
```

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'
```

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'
```

of.save.FEM

of.save.FEM *file_name* <string>
(Mandetory 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'
```

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
```

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
```

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
```

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
```

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
```

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
```

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
```

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
of.set.config blast off

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

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
```

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”
```

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
```

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.

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

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
```

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
```

2.10.2 Blast

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
```

2.10.3 Boundary

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

$f[i] = x \times table[i]$

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'
```

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

$v[i] = x \times table[i]$

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'
```

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
```

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
```

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'
```

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
```

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 $v[i] = x \times table[i]$

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'
```

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
```

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
```

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'
```

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
```

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
```

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
```

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'
```

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'
```

2.10.4 DFN

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
```

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
```

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'
```

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'
```

2.10.5 FEM

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
```

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
```

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
```

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
```

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
```

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
```

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
```

2.10.6 Geometry

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
```

of.geometry.cut.square

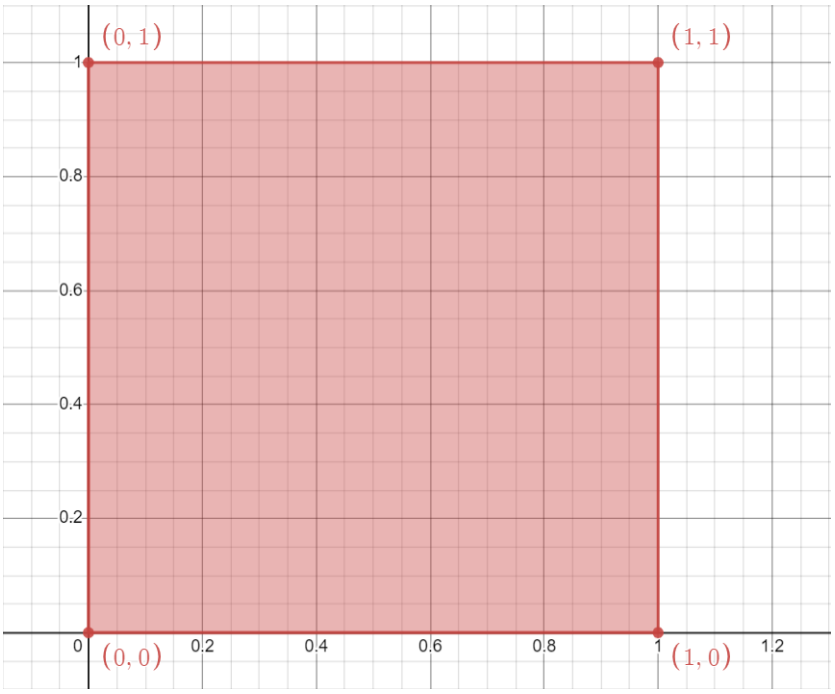
of.geometry.cut.square *new_geometry_tag* <string> *object_geometry_tag* <string> *region* <double, double, double, double>
Cut a geometry in existing geometry using the square method.

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 sqaure:

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

Example:

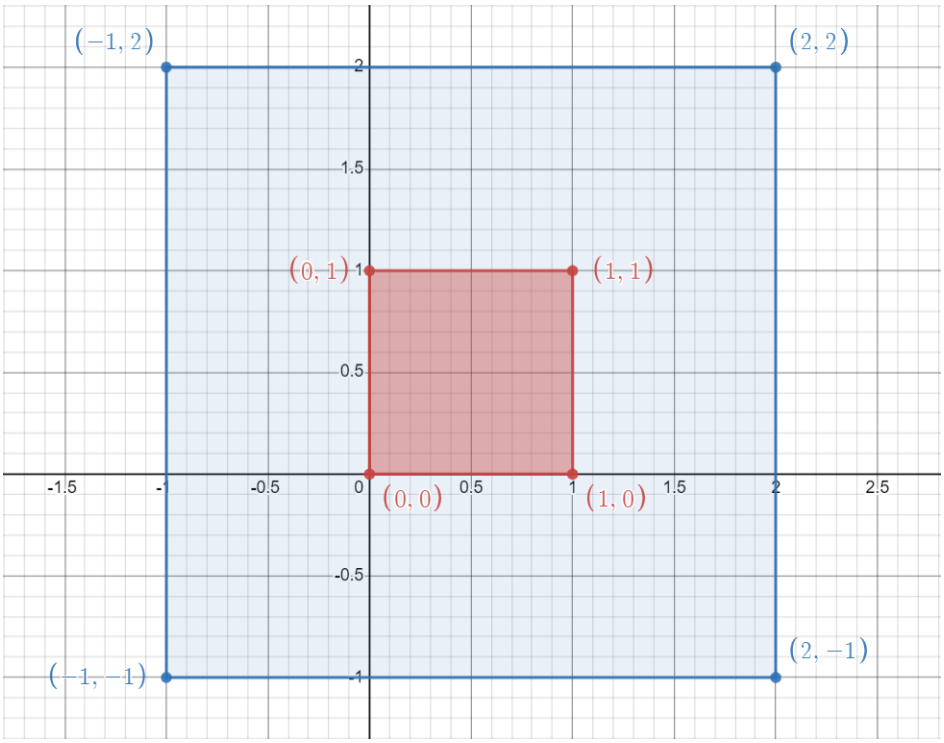
```
of.geometry.cut.square [tunnel] [rock] x [0 1] y [0 1]
```

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.

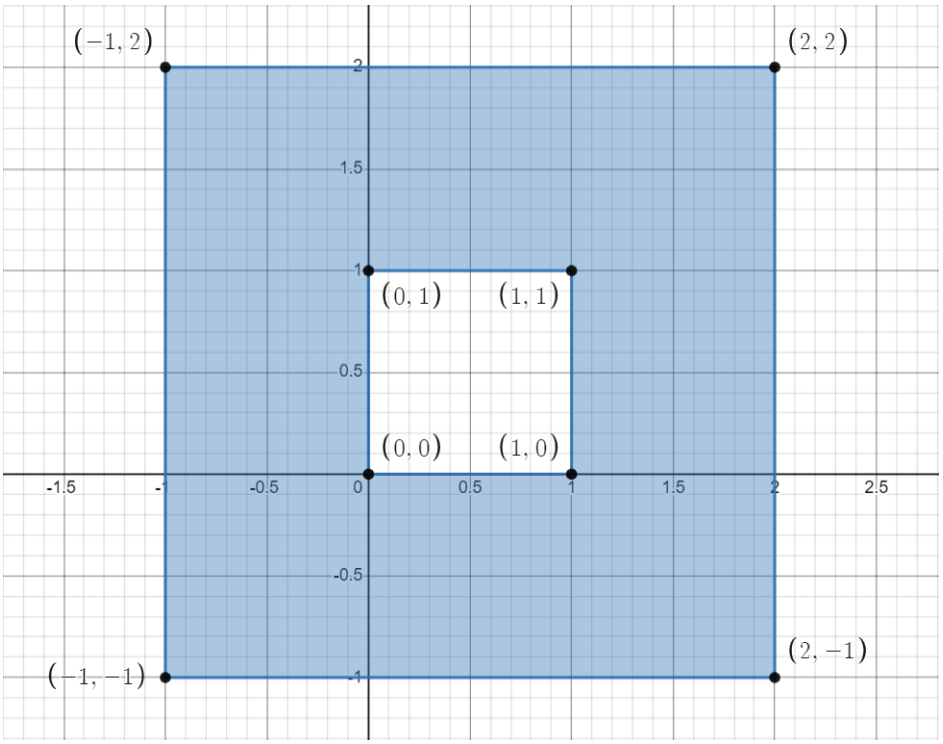
object_geometry_tag: tag of the original geometry group

region: There are three methods to define a sqaure:

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]
```



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.

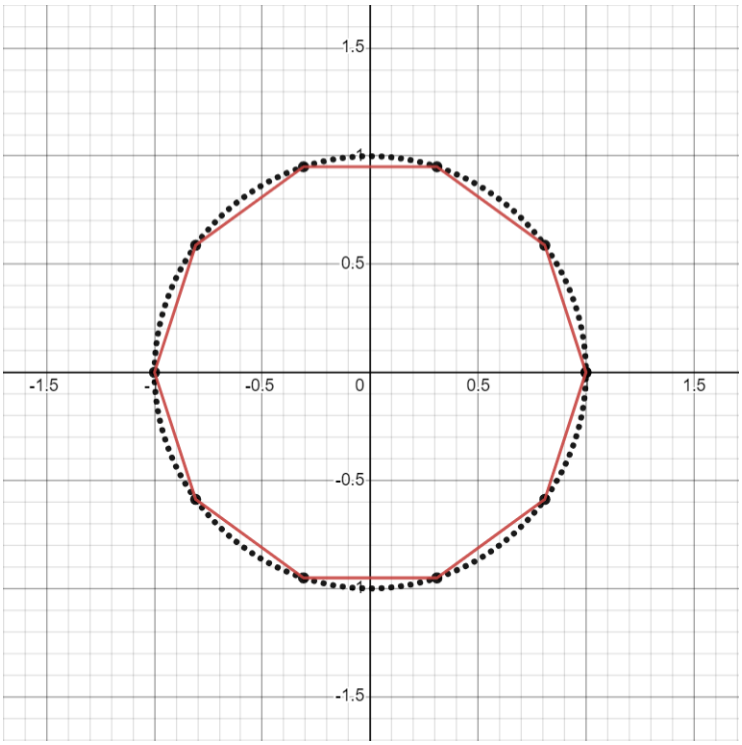
Parameters:

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:

```
# The circle will not be a perfect circle. Instead, it will be approximated by a decagon.
of.geometry.circle [rock] center [0,0] rad 1.0 segments 10
```



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.

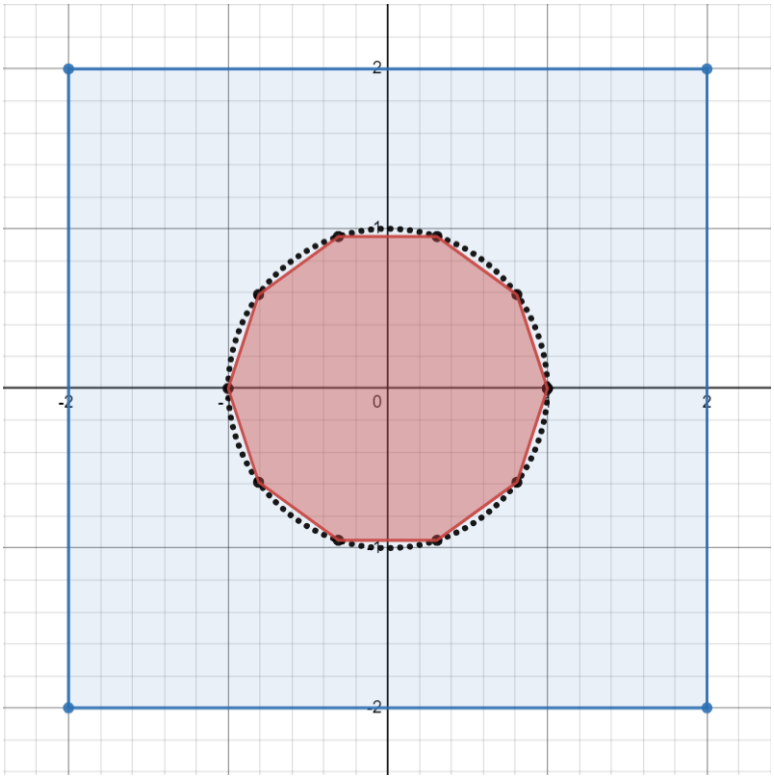
object_geometry_tag: tag of the original geometry group

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 10
```



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.

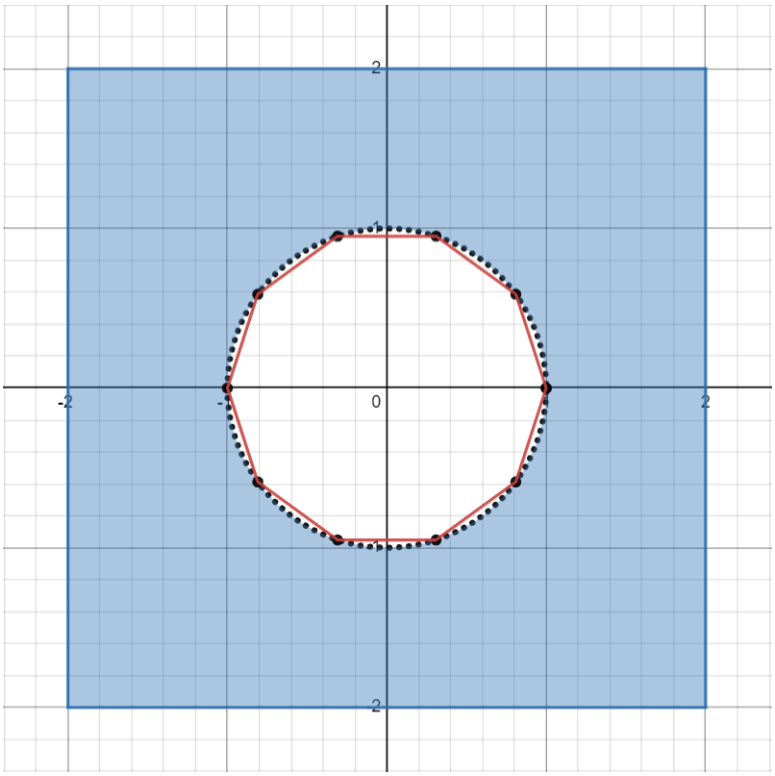
object_geometry_tag: tag of the original geometry group

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 10
```



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:

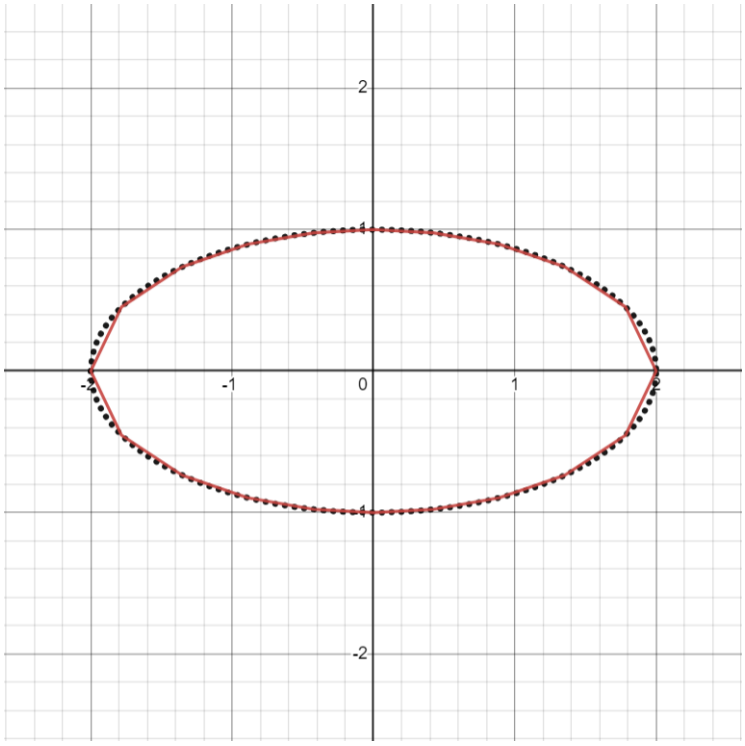
geometry_tag: tag of the geometry group

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, 2.0] theta 0 segments 10
```



of.geometry.cut.ellipse

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

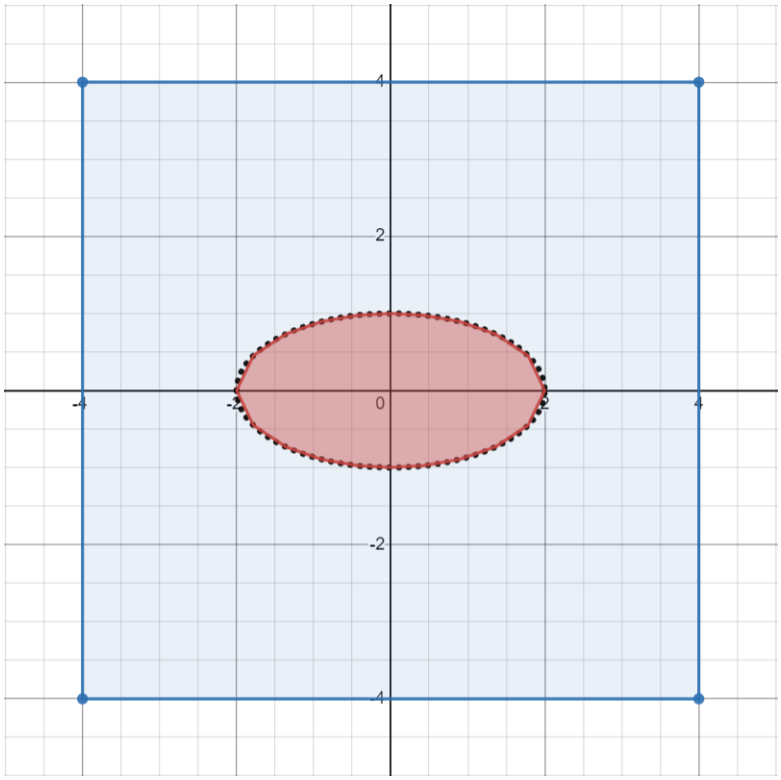
object_geometry_tag: tag of the original geometry group

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, 2.0] theta 0 segments 10
```



of.geometry.remove.ellipse

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

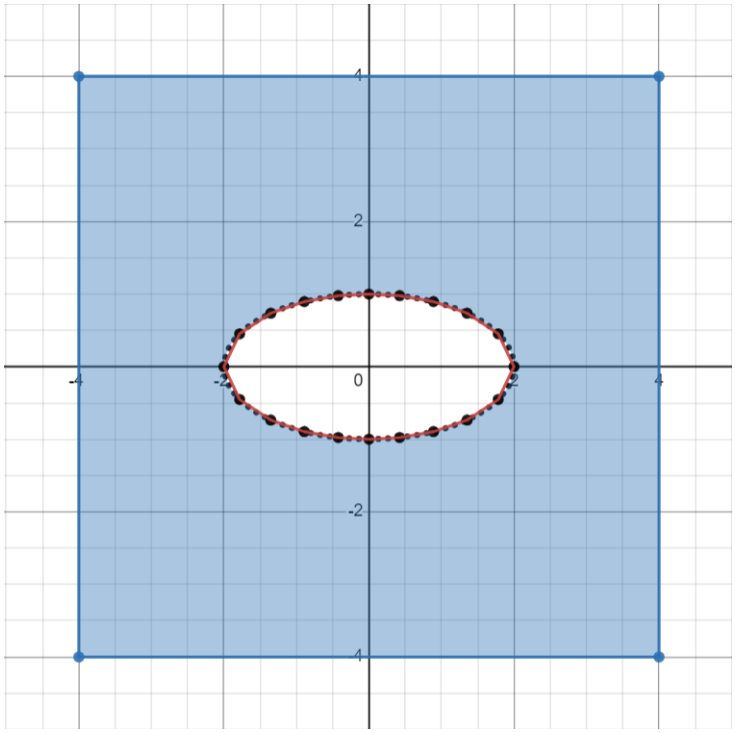
object_geometry_tag: tag of the original geometry group

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, 2.0] theta 0 segments␣
↪10
```



of.geometry.polygon

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

of.geometry.cut.polygon

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

of.geometry.remove.polygon

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

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'
```

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'
```

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] 'moutain'
```

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]
```

of.geometry.cut.joint

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

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 minsize 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]
```

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
```

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
```

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
```

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
```

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
```

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]
```

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
```

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]
```

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.gemetry.mesh meshadapt order 2
```

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]
```

2.10.7 Group

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]
```

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]
```

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]
```

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]
```

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 group 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]
```

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 group 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]
```

of.group.edge.from.cohcohelement

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]
```

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]
```

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]
```

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]
```

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]
```

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]
```

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]
```

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]
```

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]
```

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]
```

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]
```

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]
```

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]
```

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]
```

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]
```

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]
```

2.10.8 Mesh

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]
```

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]
```

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]
```

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]
```

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]
```

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]
```

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]
```

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]
```

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]
```

2.10.9 Material

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	*ela*stic	*den*sity - density of the mat	
		v - Poisson's ratio	
		E - young modulus	
		K - bulk modulus (E-v or K-G you can choose one)	
		G - shear modulus	
		*tab*le - time dependent for all kinds of modules, including E, K, G, lambda and also	
		*dam*p - stiffness damp ratio, 0.0 by default	
		The following parameters for phasefield:	
		Gc - energy release rate	
		lc - characteristic length	
		nita - phase viscous factor	
Neo-hooken	neo	*den*sity - density of the mat	
		v- Poisson's ratio	
		E - young modulus	
		K - bulk modulus (E-v or K-G you can choose one)	
		G - shear modulus	

continues on next page

Table 2 – continued from previous page

Material type	Constitutive keyword	Parameter keywords	Units
		*tab*le - time dependent for all kinds of modules, including E, K, G, lambda and also	
		*dam*p - stiffness damp ratio, 0.0 by default	
Excavation mat	*exc*avation	*den*sity - density of the mat	
		v - Poisson's ratio	
		E - young modulus	
		K - bulk modulus (E-v or K-G you can choose one)	
		G - shear modulus	
		*tab*le - time dependent for all kinds of modules, including E, K, G, lambda a	
		*dam*p - stiffness damp ratio, 0.0 by default	
Rigid bodies	*Rig*id	*den*sity - density of the mat	
Mohr-Coulomb Plastic	MC	*den*sity - density of the mat	
		v - Poisson's ratio	
		E - young modulus	
		K - bulk modulus (E-v or K-G you can choose one)	
		G - shear modulus	
		*ten*sion - tension strength, cut-off Mohor-Coulomb model	
		*coh*esion - cohesive strength, c	
		*dil*ation - dilation angle, non associated flow, equal to friction by default	
		*fri*ction - friction angle	
		*tab*le - time dependent for all kinds of modules, including E, K, G, lambda a	
		*dam*p - stiffness damp ratio, 0.0 by default	
Drucker-Prager plastic	DP	*den*sity - density of the mat	
		v - Poisson's ratio	
		E - young modulus	
		K - bulk modulus (E-v or K-G you can choose one)	
		G - shear modulus	
		*ten*sion - tension strength, cut-off Mohor-Coulomb model	
		*ksh*ear - cohesive strength, c	
		*qdi*lation - dilation angle, non associated flow, equal to friction by default	
		*qfri*ction - friction angle	
		*dam*p - stiffness damp ratio, 0.0 by default	
Transverse Mat	*Tra*nsverse	*den*sity - density of the mat	
		v_xy - Poisson's ratio in x-y direction	
		v_yx - Poisson's ratio in y-x direction	
		E_x - young modulus in x	
		E_y - young modulus in y	
		G - shear modulus	
		*dam*p - stiffness damp ratio, 0.0 by default	
Explosive EOS	JWL	*den*sity - density of the explosive	
		E0 - chemical energy of the explosive	
		detonation - detonation of the explosive	
		A - parameter A	
		B - parameter B	
		R1 - parameter R1	
		R2 - parameter R2	
		*ome*ga - parameter R1	
		x - firestarter position, x	
		y - firestarter position, y	
		t0 - fire starting time	
Mazars concrete damage	*Maz*ars	*den*sity - density of the mat	

continues on next page

Table 2 – continued from previous page

Material type	Constitutive keyword	Parameter keywords	Units
		v - Poisson's ratio	
		E - young modulus	
		At - tension parameter	
		Ac - compression parameter	
		Bt - tension factor	
		Bc - compression factor	
		e0 -	
		*dam*p - stiffness damp ratio, 0.0 by default	
Johns on-Holmquist-II	JH2	*den*sity - density	
		v - Poisson's ratio	
		E - young modulus	
		A - parameter A	
		B - parameter B	
		C - parameter C	
		M - parameter M	
		N - parameter N	
		D1 - parameter D1	
		D2 - parameter D2	
		T - parameter T	
		HEL- parameter HEL	
		PHEL- parameter PHEL	
		SHEL- parameter SHEL	
		K1- parameter K1	
		K2- parameter K2	
		K3- parameter K3	
		*dam*p - stiffness damp ratio, 0.0 by default	
Burger creep	*Bur*gers	To be added in the tutorial, if you want to use this model, contact the devel	
Power creep law	*Pow*er	To be added in the tutorial, if you want to use this model, contact the devel	
Johns on-Holmquist-II	JH2	To be added in the tutorial, if you want to use this model, contact the devel	
Yang blast model	*Yang*	To be added in the tutorial, if you want to use this model, contact the devel	

Example:

```
of.mat.element 'default' elastic den 2700 E 30e9 v 0.3 damp 2.0
```

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	*EM*	*ten*sion - tension strength	
		*coh*esion - cohesion strength	
		*fri*ction - 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 them not	
		*tta*ble - table for tension	
		ctable - table for cohesion	
		ftable - table for friction	
		GItable - table for GI	
		GIItable - table for GII	
Strain rate Evans_Marathe cohesive law	*EM_dyn*	*ten*sion - tension strength	
		*coh*esion - cohesion strength	
		*fri*ction - 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 ther	
		*tta*ble - table for tension	
		ctable - table for cohesion	
		ftable - table for friction	
		GItable - table for GI	
		GItable - table for GII	
		n_rate - critical strain rate in normal	
		n_pow - dynamic increase factor in normal	
		s_rate - critical strain rate in tangential	
		s_pow - dynamic increase factor in tangential	
Ortiz and Pandolfi exponential Cohesive Law	*OP*	*ten*sion - tension strength	
		*coh*esion - cohesion strength	
		pn - normal stiffness, will be assigned by default	
		pt - tangential stiffness, will be assigned by default	
		op - peak COD in normal	
		sp - peak COD in tangential	
Linear Cohesive Law	*LINEAR*	*ten*sion - tension strength	
		*coh*esion - cohesion strength	
		*fri*ction - 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	
		We also have a table to each variables if you want ther	
		*tta*ble - table for tension	
		ctable - table for cohesion	

continues on next page

Table 3 – continued from previous page

Material type	Constitutive keyword	Parameter keywords	Units
		ftable - table for friction	
		GItable - table for GI	
		GIItable - table for GII	
Heterogeneous Evans_Marathe cohesive law	*EM_het*	To be added in the tutorial, if you want to use this mo	
Anistropic Evans_Marathe cohesive law	*EM_ani*	To be added in the tutorial, if you want to use this mo	

Example:

```
of.mat.cohesive 'default' EM ten 1e6 coh 3e6 fric 0.3 GI 10 GII 50
```

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	*MC*	*fric*tion - friction	
		kn - normal stiffness, will be assigned by default $$pn = \lambda(K_1 + K_2)/2$$	Pa
		ks - tangential stiffness, will be assigned by default $$pn = \lambda(G_1 + G_2)/2$$	Pa
Rate dependent Mohr-Coulomb Slip	*rMC*	To be added in the tutorial, if you want to use this model, contact the developer	
Giovanni Grasselli rough shear contact	*roughshear*	*fric*tion - friction	
		kn - normal stiffness, will be assigned by default $$pn = \lambda(K_1 + K_2)/2$$	Pa
		ks - tangential stiffness, will be assigned by default $$pn = \lambda(G_1 + G_2)/2$$	Pa
		theta -parameter	
		C -parameter	
		A0 - parameter	
		B -parameter	
		tension - tension strength	
Dynamic Mohr-Coulomb Slip	*dMC*	*s_fric*tion - static friction	
		*d_fric*tion - dynamic friction	
		slip_rate- slip rate	/s
		kn - normal stiffness, will be assigned by default $$pn = \lambda(K_1 + K_2)/2$$	Pa
		ks - tangential stiffness, will be assigned by default $$pn = \lambda(G_1 + G_2)/2$$	Pa
Hertz contact model	*HERTZ*	To be added in the tutorial, if you want to use this model, contact the developer	
Friction	*FRICTION*	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
```

of.mat.hydro.matrix

To be added in the tutorial soon. If you want to use this model, please contact the developer.

of.mat.hydro.fracture

To be added in the tutorial soon. If you want to use this model, please contact the developer.

of.mat.mpm.fluid

To be added in the tutorial soon. If you want to use this model, please contact the developer.

of.mat.mpm.solid

To be added in the tutorial soon. If you want to use this model, please contact the developer.

of.mat.hydro.fluid

To be added in the tutorial soon. If you want to use this model, please contact the developer.

of.mat.hydro.gas

To be added in the tutorial soon. If you want to use this model, please contact the developer.

of.mat.nonlocal

To be added in the tutorial soon. If you want to use this model, please contact the developer.

of.mat.thermal

To be added in the tutorial soon. If you want to use this model, please contact the developer.

2.10.10 History

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
```

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]
```

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'
```

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'
```

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'
```

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'
```

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.

2.10.11 General

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.

2.10.12 Blast

Function	Description
of.blast.position	Create a blast borehole by its center and radius.

2.10.13 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 initial velocity boundaries on the nodal groups.
of.boundary.nodal.acceleration	Create the 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 initial velocity boundaries on the edge groups.
of.boundary.edge.acceleration	Create the 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 the table.

2.10.14 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.

2.10.15 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.

2.10.16 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.cir	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.ellip	create a 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.polyg	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.t	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, 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.

2.10.17 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.cohelemen	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.gbn	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.intersec	Get the bool intersect of the second - last element groups and store them into the first tags.
of.group.element.bool.subtrac	Get the bool subtract of the second to other- last element groups and store them into the first tags.
of.group.cohelement.bool.unic	Get the bool union of the second - last cohesive element groups and store them into the first tags.
of.group.cohelement.bool.inte	Get the bool intersect of the second - last cohelement groups and store them into the first tags.
of.group.cohelement.bool.subt	Get the bool subtract of the second to other- last cohelement groups and store them into the first tags.

2.10.18 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.

2.10.19 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.

2.10.20 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.

2.11 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**

2.12 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_commom.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 **d1aray)

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 ***d2aray)

void **parse_3D_double_array**(FILE *fileptr, double dinit, long n1, long n2, long n3, double ****d3aray)

void **parse_1D_int_array**(FILE *fileptr, long initial_int, long max_size, long actual_size, long **i1aray)

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 ***i2aray)

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* *d1aray)

void **parse_vector**(FILE *fileptr, double dinit, *of_vector* *d1aray)

void **parse_1D_vector**(FILE *fileptr, double dinit, long max_size, long actual_size, *of_vector* **d1aray)

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_GAUSSE**

enumerator **RANDOM_UNIFORM**

enumerator **RANDOM_EXPONENTINAL**

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_commom.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 xfli@whrsm.ac.cn

date creation: Fri Jun 18 2010 date last modification: Tue Sep 29 2020

function: common math

2.12.1 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

2.12.2 LICENSE

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

2.12.3 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", "%1ld", "%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 xfli@whrsm.ac.cn

date creation: Fri Jun 18 2010 date last modification: Tue Sep 29 2020

function: time

2.12.4 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/sesmic/openfdem_sesmic.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)

2.13 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 ***potentional_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 Members

Real **conductivity**

Real **specific_heat**

Real **expansion**

struct **Nodal_structure**

#include <openfdem_nodal.h>

Public Members

UInt **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_sesmic.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 Members

Real **pn**

Real **pt**

Real **ten**

Real **coh**

Real **delta_n**

Real **delta_s**

struct **solidcohani_EM_t**

#include <openfdem_solid_cohesive_materials.h>

Public Members

Real **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 **solidcohhet_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_commom.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_commom.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_dection**(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_potentional_contact_couple.h**

#include "../../solid/solid_mechanics/openfdem_solid.h"#include "../../solid/solid_mechanics_cohesive/openfdem_solid_cohesive.
"../../solid/solid_mechanics_contact/openfdem_solid_contact.h"#include "../../common/openfdem_struct.h"

Functions

UInt **get_potentional_contact_couple**(*Element* element, *Contact* contact, *Cohelement* cohelement)

file **openfdem_solid_contact.h**

#include "../../common/openfdem_struct.h"#include "../../common/openfdem_commom.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/sesmic**

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

2.14 Index

- genindex