Quake-DFN

version 2.0

User's Guide

Kyungjae Im

November 2025

Table of Contents

Installation and Test

Installation guide

Quake-DFN is written in Julia. VSCode IDE is recommended as it is coded and tested in VSCode. The required Julia packages are listed below. We checked that the following steps work well in Windows, Mac, and Linux systems.

1) Install Julia and VSCode in the most recent versions (check "add to PATH").

2) Open VSCode, go to extensions (), and install "Julia Language Support"

3) Open Julia REPL (*control* + *shift* + *p* for pc or *command* +*shift* + *p* for Mac) and select "Julia: start REPL".

4) Install the packages (or run: docs/*InstallPackage.jl*)

```
using Pkg
Pkg.add("DelimitedFiles")
Pkg.add("PyPlot")
Pkg.add("PyCall")
Pkg.add("JLD2")
Pkg.add("LowRankApprox")
Pkg.add("Clustering")
Pkg.add("LinearAlgebra")
Pkg.add("Printf")
Pkg.add("SpecialFunctions")
Pkg.add("StaticArrays")
Pkg.add("TriangularDislocation")
```

5) restart VSCode and make sure the Quake-DFN folder is the root (file → open folder → select Quake-DFN folder)

Test Simulation

Unpacking and running

Unpack or clone the Quake-DFN and open the QuakeDFN folder in VSCode (file → open folder in VSCode). The text file on the root (*Input_BulkFaultGeometry.txt*) contains the input file of the BP5QD benchmark problem (Jiang et al., 2022). This single file is sufficient to conduct simulations. The simulation can be implemented by running sequentially *RUN1_Hmatrix_Structure.jl* → *RUN2_BUILD_INPUT.jl* → *RUN3_QUAKEDFN.jl*. The result is automatically saved in the "results" folder. The result can be visualized by running the visualization scripts in the folder.

More details of the test simulation

The input txt file (*Input_BulkFaultGeometry.txt*) contains the rock properties and fault geometry of the BP5QD SEAS benchmark problem (Jiang et al., 2022). The geometry can be visualized by running *Plot_BulkFaultGeometry.jl*. Once you run it, Figure 1a will pop up. The detailed geometry is embedded within large surrounding loading faults that apply a constant loading rate. Zooming in shows the actual geometry of the BP5QD problem. Each row in the *Input_BulkFaultGeometry.txt* represents one block of the geometry shown in the plot.

For rectangular elements, the faults can be discretized by running the *RUN1_Hmatrix_Structure.jl* → *RUN2_BUILD_INPUT.jl*. This would generate the *Input_Discretized.jld2*. This file contains refined geometry with the input parameter that is directly used in the Quake-DFN simulation. This file can be visualized by running *Plot_Discretized_Input.jl* (figure 1b). The default setting of this plot is initial velocity distributions without loading fault. Note that the friction parameters and initial conditions can be re-adjusted at any time after building this input file. Hence, element-by-element adjustment is possible. However, if the fault geometry needs to be changed, *Input_Discretized.jld2* should be rebuilt.

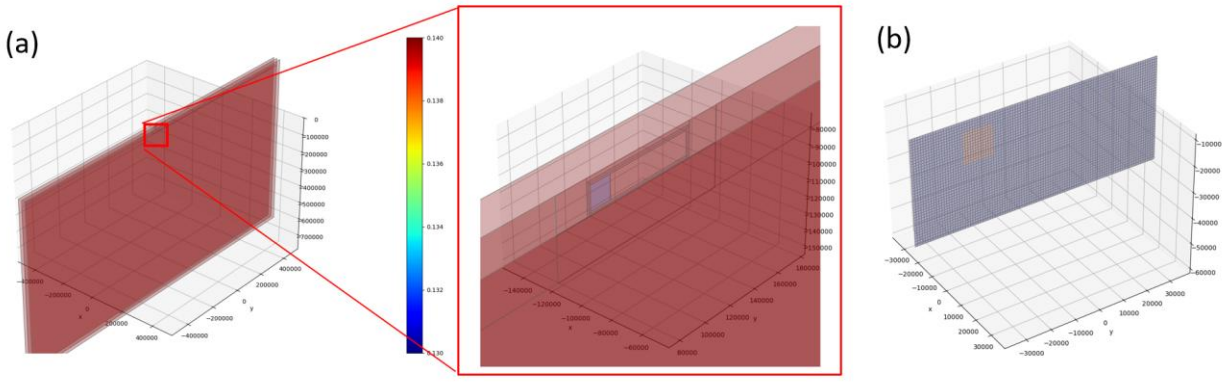


Figure 1. Test simulation geometry. (a) un-discretized fault geometry (*Plot_BulkFaultGeometry.jl*). (b) after discretization (*Plot_Discretized_Input.jl*)

Now, the simulation can be conducted by running the *RUN3_QUAKEDFN.jl*. Once the simulation is done, the result file is automatically saved in the folder *Results*. Two files will be generated at the end of the simulation: *Result/Result.jld2* and *Result/Result_Input.jld2*. The first contains simulation results, such as velocity and slip at each recorded step. The latter file contains input parameters. Several "*.jl" files provided in the *Results* folder generate different plots. Running *Result/2_3DPlot_and_Animation.jl* presents a 3D snapshot of the simulation result (default: velocity) at a given recorded step. Figure 2 shows the results for PlotStep = 30, 60, and 90.

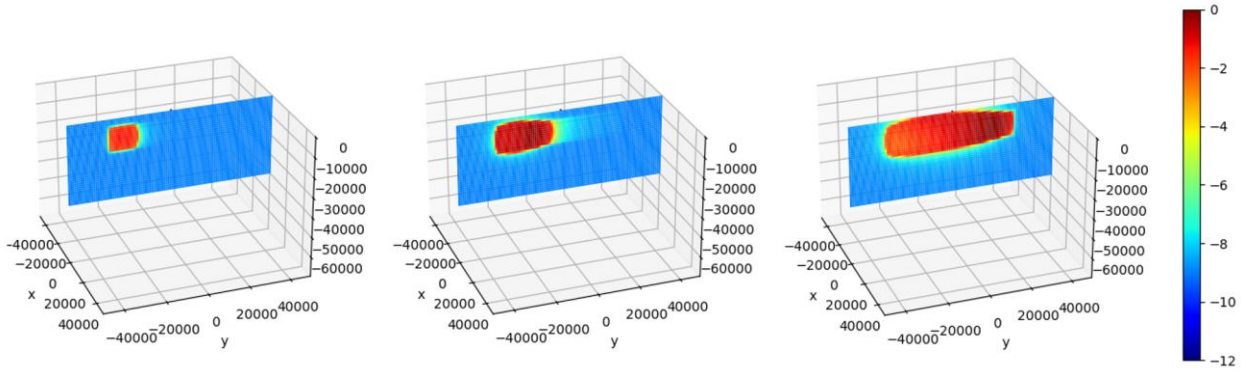


Figure 2. Simulation results plotted by *Result/2_3DPlot_and_Animation.jl* with PlotStep = 30, 60, and 90.

1. Quake-DFN Introduction

Quake-DFN is the boundary element simulator developed for earthquake rupture simulation of discretely distributed faults governed by the rate and state friction law. The simulator formulation, by default, is consistent with the widely used quasi-dynamic formulation with radiation damping, but it includes the lumped mass effect, which represents overshooting. The overshoot effect can be increased if needed. The simulation can be conducted with/without H-Matrix compression.

1.1 Workflow summary

Input files

The simulator works with one essential input that defines bulk (un-discretized for rectangular elements) fault geometry and two supplementary input files that define detailed parameters in the discretized fault geometry and external stress. The three input files are listed below.

1) *Input_BulkFaultGeometry.txt*: This essential input file defines fault geometry and information about input parameters. The actual input file (*Input_Discretized.jld2*) will be built by running *RUN_BUILD_INPUT.jl* (See section 2.1).

2) *QuickParameterAdjust.jl*: This file reads the actual discretized input (*Input_Discretized.jld2*) and changes parameters before the simulations. It can apply detailed heterogeneity instantly right before the simulation. (See section 2.3)

3) *Input_ExteralStressChange.jld2*: (Optional) This input defines external shear and normal stress change. It can be any external loading, back slip, or injection-induced stress change. (See section 2.4)

Workflow

Quake-DFN 2.0 uses H-Matrix by default. The main workflow has three steps. (1) Build Hmatrix Structure. (2) Build the input file, and (3) run the simulation. The three RUN flies are listed below

1) *RUN1_HmatrixStructure.jl*: Build Hmatrix Structure. It generates *Input_HmatrixStructure.jld2*, which is required for Hmatrix compression.

2) *RUN_BUILD_INPUT.jl*: Discretize the *Input_BulkFaultGeometry.txt* with *Input_HmatrixStructure.jld2*and generate the *Input_Discretized.jld2*, which is actual input file for the simulation.

3) *RUN_QUAKEDFN_Hmatrix.jl*: Read *Input_Discretized.jld2*. and conduct a simulation. The total simulation steps and timestep size are defined here. (See section 2.5)

The simulation result is saved in the *Results* folder as a jld2 file. The folder also provides codes for 2D and 3D plots and Catalog generation.

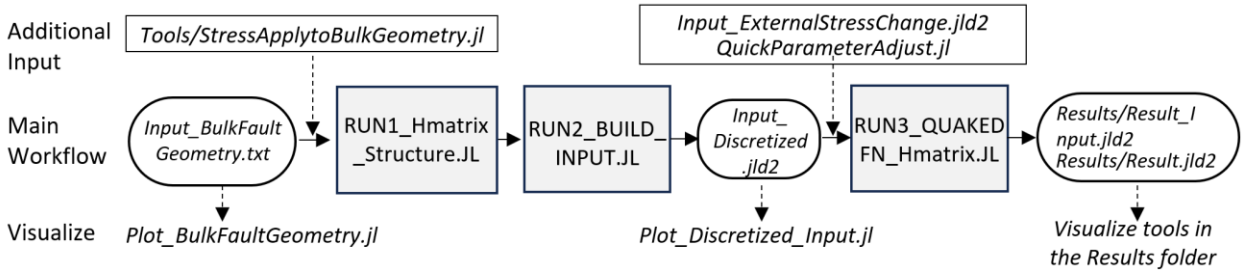


Figure 3. Simulation workflow chart.

1.2 Theoretical background

Quake-DFN considers discretely distributed boundary element faults in 3D elastic half-space with quasi-static stress transfer. A lumped mass (Im and Avouac, 2021) and radiation damping (Rice, 1993) approximate the inertia effect. With these assumptions, the momentum balance equation at i^{th} boundary element becomes

$$M_i\ddot{\delta}_i = \tau_{0i} + \sum_j k_{ij}^{\tau}(-\delta_j) - \mu_i(\sigma_{0i}^{'} + \sum_j k_{ij}^{\sigma}\delta_j + \sigma_i^{'E}) - \frac{G}{2\beta}\dot{\delta}_i + \tau_i^E, \quad (1)$$

where M is the lumped mass per unit contact area for each element, δ_{0j} is the initial displacement of element j, δ_j is the shear displacement of element j, σ'_{0i} is the initial effective normal stress of element i, G is shear modulus, β is shear wave speed, and k_{ij} is a stiffness matrix that defines the elastic stress change imparted on element i due to slip of element j (k^{τ} and k^{σ} represent shear and normal stiffness matrix, respectively). The stiffness matrices are calculated by assuming quasistatic stress transfer (Okada, 1992). The τ^E and σ'^E are shear and effective normal stress changes driven by external stress, such as tectonic loading or poro-elastic stress change.

We used the rate and state friction (Dieterich, 1979; Marone, 1998)

$$\mu = \mu_0 + a\log\left(\frac{V}{V_0}\right) + b\log\left(\frac{V_0\theta}{D_c}\right) \quad (2)$$

and the aging law with the normal stress-dependent evolution (Linker and Dieterich, 1992)

$$\frac{d\theta}{dt} = 1 - \frac{V\theta}{D_c} - \alpha\frac{\theta\dot{\sigma}}{b\sigma}, \quad (3)$$

where V is velocity, θ is the state variable, μ_0 is a reference friction coefficient at reference velocity V_0, Dc is a critical slip distance, and a and b are empirical constants for the magnitude of direct and evolution effects, respectively. The α term defines normal stress-dependent state evolution.

The lumped mass approximation allows for inertial effects not captured by radiation damping alone, such as inertial overshoot or friction-induced vibrations (Im & Avouac 2021). The lumped mass per unit area M approximates the equivalent mass in a rupture process. If rupture size is fixed (as can happen in a repeating earthquake on a finite-sized fault), M can be defined as

$$M \sim \frac{\rho L}{(1-\nu)\pi^2}, \quad (4)$$

where L is the length scale of the rupture size, ρ is rock density, and ν is Poisson's ratio. Conversely, if the rupture size is not fixed, L may be approximated by the expected rupture size in the simulations. By default, M is set to be h_{min} × density / 2, where h_{min} is the minimum size of all elements. This setting makes M/k_{ij} sufficiently low and suppresses the overshoot effect, making the simulation close to radiation damping alone.

The governing equation (equation 1) is a widely used quasi-dynamic formulation using radiation damping (e.g., Erickson et al., 2020) with an added overshoot effect. Therefore, our simulator is compatible with the other simulators that employ radiation damping (see section 3.1). We utilize two methods to solve equations 1-3: (i) a typical iterative method that is applied to a low-velocity system and (ii) the method of Im et al. (2017), which is stable at high velocity. The two solvers are automatically switched for each element based on their velocities. The timestep is dependent on the maximum velocity but automatically adapts if it fails to find a converged solution.

To resolve the earthquake rupture process, the shear stiffness of an element should be sufficiently larger than the critical stiffness (Rice, 1993). The ratio

$$\frac{k_{ii}^{\tau}}{K_c} = \frac{k_{ii}^{\tau} D_c}{(b-a)\sigma} \tag{5}$$

is required to achieve a minimum resolution, the k^{τ}_{ii}/K_c should be sufficiently larger than 1. In Quake-DFN, this can be checked by plotting the distribution of this ratio (equation 5) before running the simulation. Once faults are discretized, run *Plot_Discretized_Input.jl* with un-commenting *PlotInput=KoverKC* and commenting out others (see below).

```
##########################################################################
######################## Which parameter want to plot? ############
    ColorMinMax = 0
    # PlotInput = log10.(Fault_Theta_i); ColorMinMax = 0
    # PlotInput = log10.(Fault_V_i); ColorMinMax = 0
    # PlotInput =Fault_NormalStress; ColorMinMax = 0
    PlotInput =KoverKC ;ColorMinMax=[0,5]
    # PlotInput =UnderResolved ;ColorMinMax=[0,1]
    # PlotInput = Fault_a - Fault_b; ColorMinMax = 0
    # PlotInput =  Fault_BulkIndex; ColorMinMax = 0
    # PlotInput = Fault_Dc; ColorMinMax = 0
    # PlotInput = FaultLLRR; ColorMinMax = 0
    # PlotInput = FaultLLRR .* Fault_Friction_i; ColorMinMax = 0
    # PlotInput = Fault_Friction_i; ColorMinMax = 0
```

2. Workflow

2.1 Required Input File: *Input_BulkFaultGeometry.txt*

The *Input_BulkFaultGeometry.txt* file contains information on rock properties, fault geometry, frictional properties, and element size after the discretization. The input file formats are different between the rectangular and triangular grid cases. The first four lines of rectangular grid input appear to be

```
SwitchSS/RN ShearMod    PoissonRatio    R_Density   Crit_TooClose   TooCloseNormal_Multiplier   Minimum_NS
1.0 2.0e10  0.2 2400.0  1.05    0.6 2.0e6
Ctr_X   Ctr_Y   Ctr_Z   St_L    Dip_L   StAng   DipAng  Rake    a       b       Dc      Theta_i V_i Fric_i  Sig0    SigGrad V_Const MaxLeng
-1000.0 -1000.0 1500.0  2000.0  3000.0  50.0    70.0    0.0     0.003   0.006   0.0002  1.0e10  1.0e-15 0.6 2.0e6   6000.0  0.0 200.0
```

And the first four lines of the triangular grid should be

```
1  SwitchSS/RN ShearMod    PoissonRatio    R_Density   Crit_TooClose   TooCloseNormal_Multiplier   Minimum_NS
2  0.0 2.0e10  0.25    2670.0  1.05    0.6 2.0e6
3  P1_x        P1_y        P1_z        P2_x        P2_y        P2_z        P3_x        P3_y                P3_z                Rake    a       b       Dc      Theta_i     V_i     Fric_i  Sig0    SigGrad V_Const MaxLeng
4  -2492.1392  -315.6315   -681.4651   -2635.3689  -296.6863   -416.3537   -2634.577   -546.6224000000001  -416.12330000000003 0.0     0.005   0.01    0.0003  100000.0    1.0e-15 0.6     1.0e7   0.0     0.0     100000.0
```

The first and third lines (texts) are the headers of input parameters for the input numbers of the second and fourth (and after) lines, respectively. The first and second lines are the same in both the rectangular and triangular systems, as

(1) SwitchSS/RN — Input system of sense of slip:
(0): Rake Angle, (1): Strike-Slip, (2): Normal-Reverse.
*) option (1) and (2) only apply to a rectangular grid system

(2) ShearMod — Shear modulus [Pa]

(3) PoissonRatio — Poisson's ratio [-]

(4) R_Density — Rock Density [kg/m^3]

(5) Crit_TooClose — **Not used**

(6) TooCloseNormal_Multiplier: **Not used**

(7) Minimum_NS — Minimum normal stress allowed in the simulation

The items (5) and (6) were introduced to improve numerical stability in very complex fault systems, but they are not used in the current version.

Sometimes complex geometry results in very strong interactions. This problem can be alleviated by separating faults at intersections, as shown in Figure 4b. In this way, the element center cannot be located at intersections.

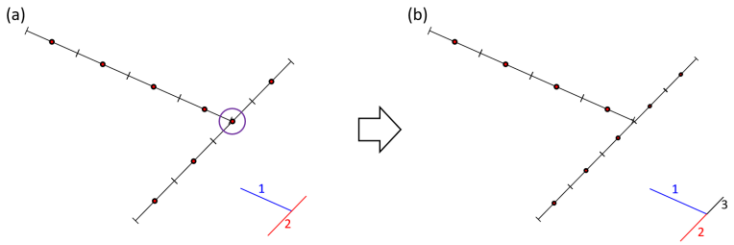


Figure 4. (a): Strong interaction example. (b) Strong interaction is avoided. (b) can be achieved in *Input_BulkFaultGeometry.txt* by separating faults at the intersection (bottom right figure in b).

The headers of the third line define fault geometry, frictional properties, and the initial state. For a rectangular element system,

(1) Ctr_X x component of Fault Center [m]
(2) Ctr_Y y component of Fault Center [m]
(3) Ctr_Z depth of Fault Center (positive) [m]
(4) St_L fault length along strike [m]
(5) Dip_L fault length along dip [m]
(6) StAng Strike Angle (0-180) [Degree]
(7) DipAng Dip Angle (0-180) [Degree]
(8) Rake Rake angle (if SwitchSS/RN = 0) [Degree] or sense of slip (if SwitchSS/RN /=0)
SwitchSS/RN = 1: Left lateral (-1) right lateral (1)
SwitchSS/RN = 2: reverse (-1) normal (1)
(9) a Rate-and-State Parameter "a" [-]
(10) b Rate-and-State Parameter "b" [-]
(11) Dc Rate-and-State Parameter "D_c" [m]
(12) Theta_i Initial value of Rate-and-State Parameter θ_i [s]
(13) V_i Initial velocity V_i [m/s]
(14) Fric_i Initial Friction μ_i [-]
(15) Sig0 Normal stress at the surface [Pa]
(16) SigGrad Normal stress gradient with depth [Pa/m]. Zero for uniform normal stress
(17) V_Const Constant velocity (if non-zero, it will only slip at this velocity) [m/s]
(18) MaxLeng Discretized element length [m]

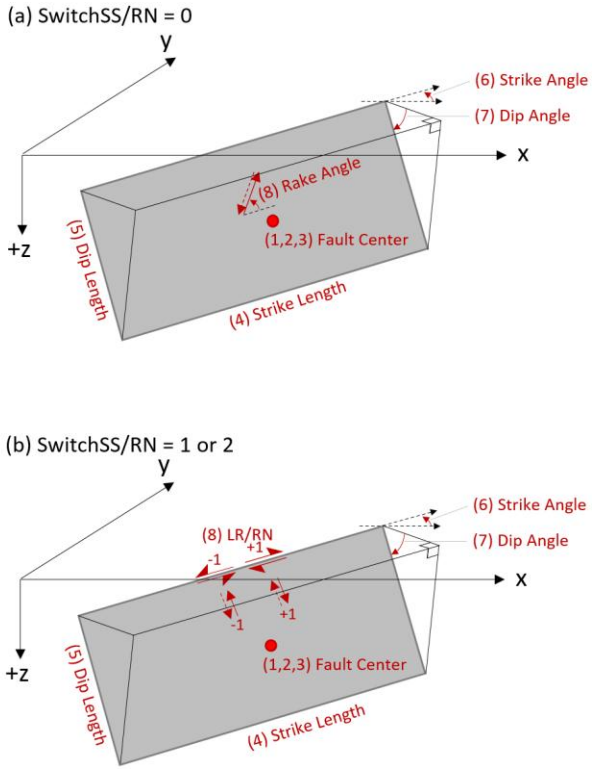


Figure 5. Definitions of the fault geometry parameters for a rectangular mesh system. (a) in case SwitchSS/RN = 0. (b) in case SwitchSS/RN = 1 (strike-slip mode) or 2 (normal and reverse mode)

In a triangular grid system,

(1) P1X x component of vertex 1 [m]
(2) P1Y y component of vertex 1 [m]
(3) P1Z z component of vertex 1 [m]
(4) P1X x component of vertex 2 [m]
(5) P1Y y component of vertex 2 [m]
(6) P1Z z component of vertex 2 [m]
(7) P1X x component of vertex 3 [m]
(8) P1Y y component of vertex 3 [m]
(9) P1Z z component of vertex 3 [m]
(10) Rake Rake angle [Degree]
(11) a Rate-and-State Parameter "a" [-]
(12) b Rate-and-State Parameter "b" [-]
(13) Dc Rate-and-State Parameter "D_c" [m]
(14) Theta_i Initial value of Rate-and-State Parameter θ_i [s]
(15) V_i Initial velocity V_i [m/s]
(16) Fric_i Initial Friction μ_i [-]
(17) Sig0 Normal stress at the surface [Pa]
(18) SigGrad Normal stress gradient with depth [Pa/m]. Zero for uniform normal stress
(19) V_Const Constant velocity (if non-zero, it will only slip at this velocity) [m/s]
(20) MaxLeng Discretized element length [m] (**Not used**)

The rake angle is zero degrees for left lateral slip and 180 degrees for right lateral slip. It should be assessed in a counterclockwise direction from the viewpoint of the xz plane (see figure 6). For more intricate fault geometries, stress fields can be applied to the geometry by using the *Tools/StressApplytoBulkGeometry.jl* function (section 2.1.1). This approach will automatically compute the initial friction and slip orientation and update *Input_BulkFaultGeometry.txt* accordingly.

The input parameters for the triangular grid system are 20 folds. The simulator automatically distinguishes between rectangular and triangular input based on the variation in length. Unlike the rectangular grid system, the triangular system cannot be further discretized in the current version so the MaxLeng parameter is not used in this version.

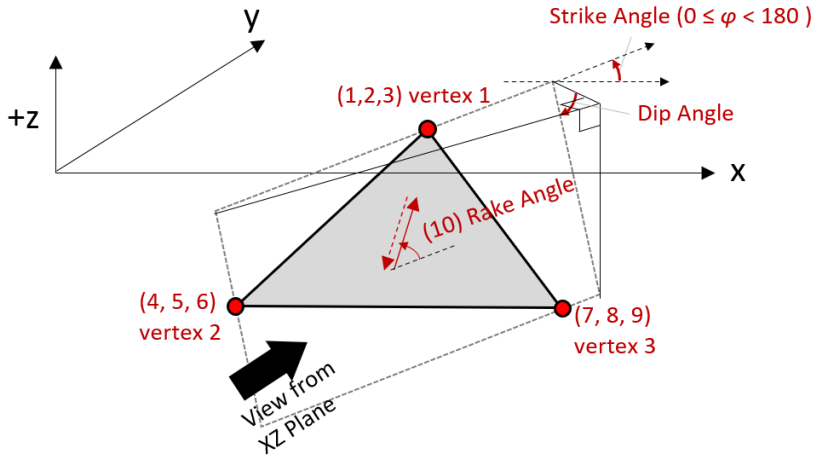


Figure 6. Definitions of the fault geometry parameters for the triangular mesh system.

The friction parameters and initial conditions ((9-16) for rectangular and (11-18) for triangular systems) can be adjusted after discretization. Conversely, the geometry and rake angle cannot be adjusted once the stiffness matrix is built (i.e., after running the *RUN_BUILD_INPUT.jl*).

The initial condition of the rate and state friction law is

$$\mu_i = \mu_0 + a \log\left(\frac{V_i}{V_0}\right) + b \log\left(\frac{V_0 \theta_i}{D_c}\right). \quad (6)$$

Every variable and initial condition of equation 6 is defined in the *Input_BulkFaultGeometry.txt*, except μ_0 and V_0. Note that these two parameters are interdependent, and therefore, one can set V_0 arbitrarily. In Quake-DFN, $V_0 = 10^{-9}$m/s, and μ_0 is determined by equation 6.

Instead of V_i, one may want to set μ_0 as an initial value since it is a laboratory-measurable quantity. This can be done by using equation 6. For example, if one wants to fix μ_0 and make V_i be calculated correspondingly, one can calculate

$$V_i = V_0 exp\left(\left(\mu_i - \mu_0 - b \log\left(\frac{V_0 \theta_i}{D_c}\right)\right)/a\right) \quad (7)$$

with the desired μ_0. Putting this initial velocity in the input will set μ_0 to the desired value. Note that this treatment can also be performed by QuickParameterAdjust.jl, on the discretized-input-file as well.

2.1.1 Stress application to complex fault systems: *Tools/StressApplytoBulkGeometry.jl*

Each bulk element is characterized by a distinct rake angle that corresponds to the prevailing regional stress field. Quake-DFN provides a function to compute and implement the initial stress (friction) and rake angle based on any applied stress field. This process can be executed using the *Tools/StressApplytoBulkGeometry_Homogeneous.jl* after the geometry has been built. This code reads the *Input_BulkFaultGeometry.txt* and modifies it based on the applied stress field. An illustrative example can be found in the Example Simulations section.

2.2. Build H-Matrix Structure: *RUN1_Hmatrix_Structure.jl*

Quake-DFN utilizes H-Matrix by default. Large-scale simulation requires a large number of elements, N. Since memory consumption and simulation time increase with N^2, if N is large, the simulation requires an impractical amount of memory and simulation time. This problem can be alleviated by using H-Matrix compression (Hackbusch, 1999). Quake-DFN allows the H-Matrix compression for large-scale simulations (figure 7). See Figure 7 for memory saving and simulation speed improvement.

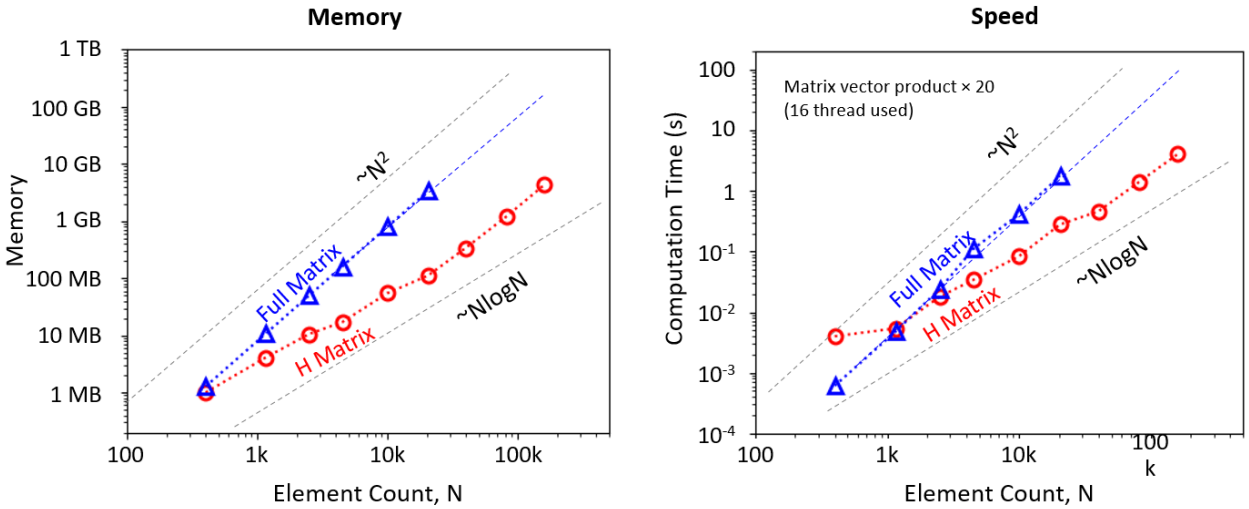
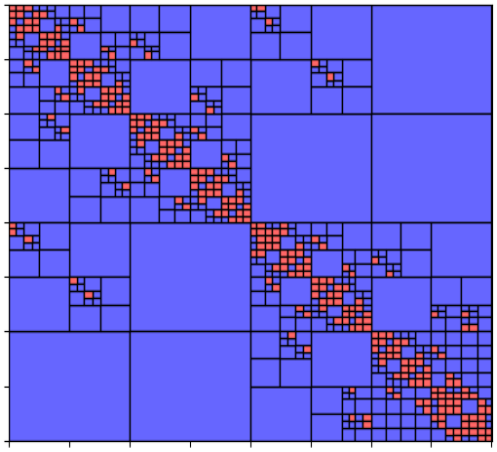


Figure 7. Memory compression and speed change by using H-Matrix approximation.

Once the Input_Bulk Fault Geometry is ready, the first step of the simulation is running *RUN1_Hmatrix_Structure.jl*. Once run, it will provide the HMatrix structure as follows.



This structure file will be saved as *Input_HmatrixStructure.jld2*.

2.3. Discretize the fault and build stiffness matrices: *RUN2_BUILD_INPUT.jl*.

The *RUN_BUILD_INPUT.jl* discretizes the bulk fault (rectangular element only) and builds the stiffness matrix and compress matrix according to the Hmatrix structure. For the rectangular system, Okada's (1992) formulation is used, and for the triangular system, Nikkhoo and Walter (2015) is used. The stiffness matrix defines shear and normal stress change driven by a slip of an element. The calculated stiffness matrix and input parameters are saved as *Input_Discretized.jld2*.

```
################################################################################
############## Hmatrix with Distributed discretization ####################
Hmatrix = true
HowManyDistribution = 5
################################################################################
```

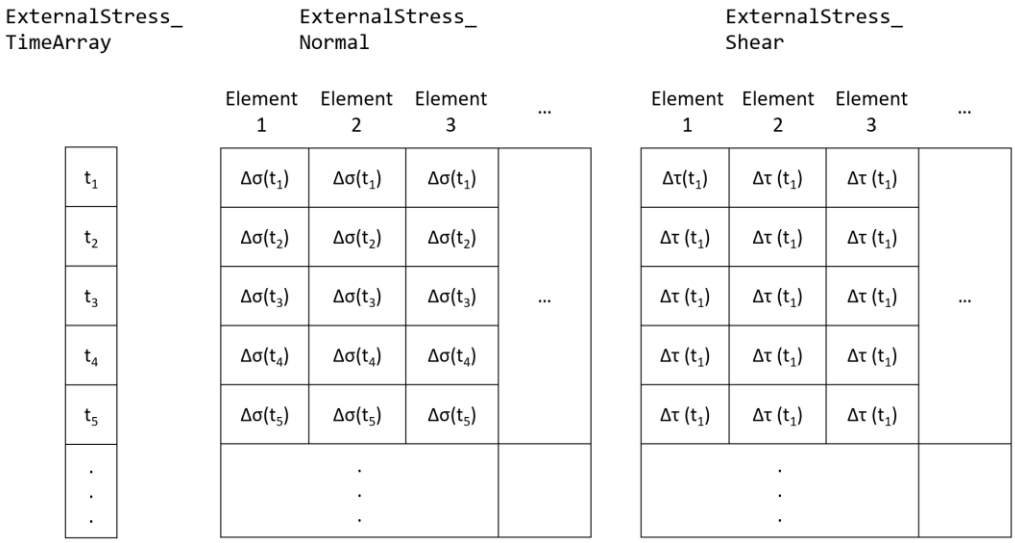
In the code, there are two parameter inputs. The option [Hmatrix = true] indicates that the simulation will use Hmatrix. If you prefer to use a full matrix, which may consume a large amount of memory, you can simply set [Hmatrix = false]. The [HowManyDistribution] parameter determines how the matrix-building task is divided. For example, if you set [HowManyDistribution = 5], the task will be split into five different terminals. When executed, five new terminals will pop up and conduct the matrix building separately.

2.3.1 Parameter adjustment for discretized elements: *QuickParameterAdjust.jl*

QuickParameterAdjust.jl can change parameters and initial state at each individual discretized element level. Therefore, complex heterogeneity can be implemented here. This parameter adjustment does not require re-discretization. Hence, sensitivity tests can be easily implemented in this file. Once faults are discretized, one can test a wide range of parameter sets here. This file is automatically run right before the simulation.

2.3.2. External Stress change (Optional): *Input_ExternalStressChange.jld2*

This file is optional. With this file, any time-dependent external stress can be applied to each element. The element count in this file should be equal to the discretized element count in *Input_Discretized.jld2*. Otherwise, the simulation will neglect the file. The file should contain three variables (variable name should be identical to): (1) ExternalStress_TimeArray, (2) ExternalStress_Normal, and (3) ExternalStress_Shear. Each variable should contain the following information.



Once the simulation begins, the simulator automatically interpolates shear and normal stress change from the variables.

2.4. Conduct Simulation: *RUN3_QUAKEDFN.jl*

RUN3_QUAKEDFN.jl reads *Input_Discretized.jld2* (and additionally *QuickParameterAdjust.jl* and *Input_ExternalStressChange.jld2* if defined) and runs the simulation.

```
########################## Simulation Time Set ##############################
TotalStep = 10000 # Total simulation step
SaveStep = 5000 # Automatically saved every this step
RecordStep = 10 # Simulation sampling rate !! should be a factor of SaveStep !!

ThreadCount = 5 # Only used for HMatrix Simulations. 5~8 seems to be most
efficient in most of the computer
```

These variables define total simulation and recording length. The simulation will be conducted for ***TotalStep***. The simulator stores the simulation results of every ***RecordStep*** in memory (others will be discarded). The recorded result will be saved to the *Result* folder every ***SaveStep***s. In the above example, the simulation will be conducted for 10000 steps, and the simulation progress is saved twice (at steps 5000 and 10000) in the *Result* folder. The SaveStep should be smaller than TotalStep (otherwise, no result will be saved) and better be a divisor of the TotalStep. The resulting file will contain information on a total of 1000 recorded steps (every 10 simulation steps).

The Hmatirx will be solved in parallel. The [ThreadCount] defines how many threads are to be used for the simulation.

```
########################## Time Stepping Setup ################################
DtCut = 5
SwitchV = 1e-2
RuptureTimeStepMultiple = 3
MaximumDt = 1e7
VerticalLengthScaleforM = 0
```

Time stepping logic (*dt* velocity dependence) is defined here. Simulation timestep size is mainly dependent on the maximum velocity of all the elements and the acceleration. SwitchV delineates the time step at which the numerical solver transitions. If the maximum velocity of any element exceeds this threshold, a "rupture solver" is employed to solve the equation. The "rupture solver" utilizes the solution method outlined in Im et al. (2017 JGR). This approach is numerically more stable during rupture events while still addressing the same equation. In this case, the time step is set by ***RuptureTimeStepMultiple***/10 of the natural period of each element. During the static phase, the time step is determined by the velocity (V) and acceleration (a) of the previous time step, given by $\Delta t = (V/a)/$***DtCut***. The simulation selects the smallest value of Δt among all elements.

During the simulation, Julia REPL window updates the simulation status as follows.

(1)	(2)	(3)	(4)	(5)
0.00100	0.00002	1.556e-02	1.898e-02	1
0.00200	0.00002	1.673e-02	1.898e-02	1
0.00300	0.00002	1.733e-02	1.898e-02	1
0.00400	0.00002	1.785e-02	1.898e-02	1
0.00500	0.00002	1.836e-02	1.898e-02	1
0.00600	0.00003	1.889e-02	1.898e-02	1
0.00700	0.00003	1.942e-02	1.898e-02	1
0.00800	0.00003	1.997e-02	1.898e-02	1

Each column represents:
(1) Simulation progress (Simulation will be over when this becomes 1)
(2) the time in the simulation [day]
(3) maximum velocity of all elements [m/s]
(4) timestep size [s]
(5) Is a high-velocity solver being used? (if 1, high-velocity solver (Im et al., 2017) is being used for high-velocity elements

3. Example Simulations

Here are some brief instructions for the provided examples in the *Tools/InputGeometryExamples* folder. **Some examples here are coarsely discretized for quick testing**. To resolve nucleation length correctly, one can make it finer by adjusting "MaxLeng" in the *Input_BulkFaultGeometry.txt*.

3.1 BP5QD (Jiang et al., 2022)

(1) Run *Tools/InputGeometryExamples/Example1_BuildGeometry_BP5QD.jl*

(2) Run *RUN1_Hmatrix_Structure.jl*

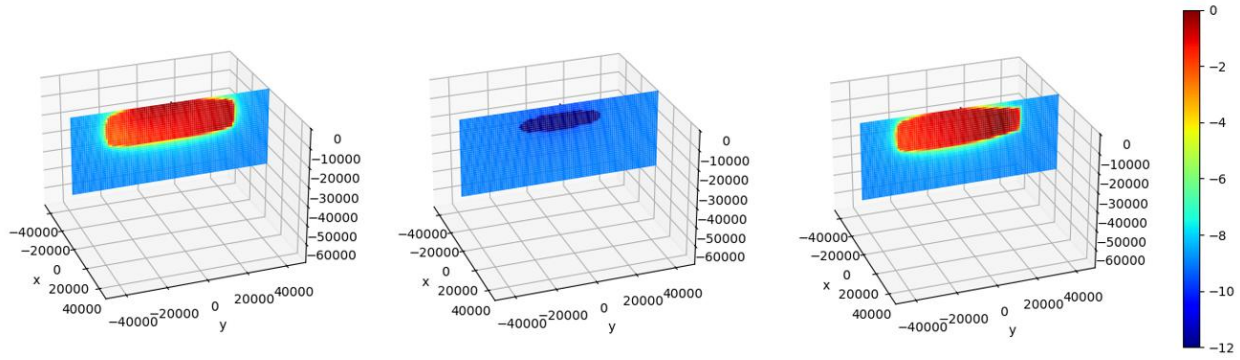
(3) Run *RUN2_BUILD_INPUT.jl*

(4) Run *RUN3_QUAKEDFN.jl*

```
########################## Simulation Time Set ###############################
TotalStep = 10000 # Total simulation step
SaveStep = 5000 # Automatically saved every this step
RecordStep = 10 # Simulation sampling rate !! should be a factor of SaveStep !!
```

Simulation results (*Results/2_3DPlot_and_Animation.jl*)

Plotstep: 100, 400, 600



Direct Application of Long-Term Slip Rate (Back Slip)

In this case, we remove the loading faults. Instead, we apply the same effect by back-slip loading.

(1) Run *Tools/InputGeometryExamples/Example1_BuildGeometry_BP5QD.jl*

In the *Input_BulkFaultGeometry.txt*, delete the last five faults (loading faults)

Input_BulkFaultGeometry.txt

```
SwitchSS/RN ShearMod    PoissonRatio    R_Density   Crit_TooClose   TooCloseNormal_Multiplier   Minimum_NS
1.0 3.2038e10   0.25    2670.0  1.05    0.6 2.0e6
Ctr_X   Ctr_Y   Ctr_Z   St_L    Dip_L   StAng   DipAng  LR/RN   a   b   Dc  Theta_i V_i Fric_i  Sig0    SigGrad V_Const MaxLeng
0.0 -24000.0    10000.0 12000.0 12000.0 90.0    90.0    1.0 0.004   0.03    0.13    1.3e8   0.03    0.6 2.5e7   0.0 0.0 1000.0
0.0 6000.0  10000.0 48000.0 12000.0 90.0    90.0    1.0 0.004   0.03    0.14    1.4e8   1.0e-9  0.6 2.5e7   0.0 0.0 1000.0
0.0 31500.0 10000.0 1000.0  16000.0 90.0    90.0    1.0 0.031000000000000003    0.03    0.14    1.4e8   1.0e-9  0.6 2.5e7   0.0 0.0 1000.0
0.0 -31500.0    10000.0 1000.0  16000.0 90.0    90.0    1.0 0.031000000000000003    0.03    0.14    1.4e8   1.0e-9  0.6 2.5e7   0.0 0.0 1000.0
0.0 0.0 2500.0  62000.0 1000.0  90.0    90.0    1.0 0.031000000000000003    0.03    0.14    1.4e8   1.0e-9  0.6 2.5e7   0.0 0.0 1000.0
0.0 0.0 17500.0 62000.0 1000.0  90.0    90.0    1.0 0.031000000000000003    0.03    0.14    1.4e8   1.0e-9  0.6 2.5e7   0.0 0.0 1000.0
0.0 30500.0 10000.0 1000.0  14000.0 90.0    90.0    1.0 0.013000000000000001    0.03    0.14    1.4e8   1.0e-9  0.6 2.5e7   0.0 0.0 1000.0
0.0 -30500.0    10000.0 1000.0  14000.0 90.0    90.0    1.0 0.013000000000000001    0.03    0.14    1.4e8   1.0e-9  0.6 2.5e7   0.0 0.0 1000.0
0.0 0.0 3500.0  60000.0 1000.0  90.0    90.0    1.0 0.013000000000000001    0.03    0.14    1.4e8   1.0e-9  0.6 2.5e7   0.0 0.0 1000.0
0.0 0.0 16500.0 60000.0 1000.0  90.0    90.0    1.0 0.013000000000000001    0.03    0.14    1.4e8   1.0e-9  0.6 2.5e7   0.0 0.0 1000.0
0.0 41000.0 9000.0  18000.0 18000.0 90.0    90.0    1.0 0.04    0.03    0.14    1.4e8   1.0e-9  0.6 2.5e7   0.0 0.0 1000.0
0.0 -41000.0    9000.0  18000.0 18000.0 90.0    90.0    1.0 0.04    0.03    0.14    1.4e8   1.0e-9  0.6 2.5e7   0.0 0.0 1000.0
0.0 0.0 29000.0 100000.0    22000.0 90.0    90.0    1.0 0.04    0.03    0.14    1.4e8   1.0e-9  0.6 2.5e7   0.0 0.0 1000.0
0.0 0.0 1000.0  64000.0 2000.0  90.0    90.0    1.0 0.04    0.03    0.14    1.4e8   1.0e-9  0.6 2.5e7   0.0 0.0 1000.0
0.0 525000.0    20000.0 950000.0    40000.0 90.0    90.0    1.0 0.04    0.03    0.14    1.4e8   1.0e-9  0.6 2.5e7   0.0 1.0e-9  1.0e10
0.0 -525000.0   20000.0 950000.0    40000.0 90.0    90.0    1.0 0.04    0.03    0.14    1.4e8   1.0e-9  0.6 2.5e7   0.0 1.0e-9  1.0e10
0.0 0.0 520000.0    2.0e6   960000.0    90.0    90.0    1.0 0.04    0.03    0.14    1.4e8   1.0e-9  0.6 2.5e7   0.0 1.0e-9  1.0e10
20000.0 0.0 500000.0    2.0e6   1.0e6   90.0    90.0    -1.0    0.04    0.03    0.14    1.4e8   1.0e-9  0.6 2.5e7   0.0 1.0e-9  1.0e10
-20000.0    0.0 500000.0    2.0e6   1.0e6   90.0    90.0    -1.0    0.04    0.03    0.14    1.4e8   1.0e-9  0.6 2.5e7   0.0 1.0e-9  1.0e10
```

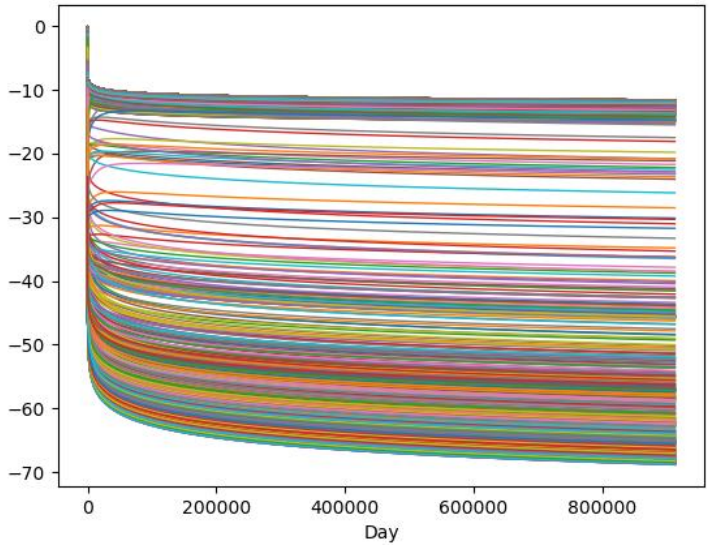
Delete these lines

(2) Run *RUN1_Hmatrix_Structure.jl*

(3) Run *RUN2_BUILD_INPUT.jl*

(4) Run *RUN3_QUAKEDFN.jl*

Plot *Results/1_2DPlot.jl*

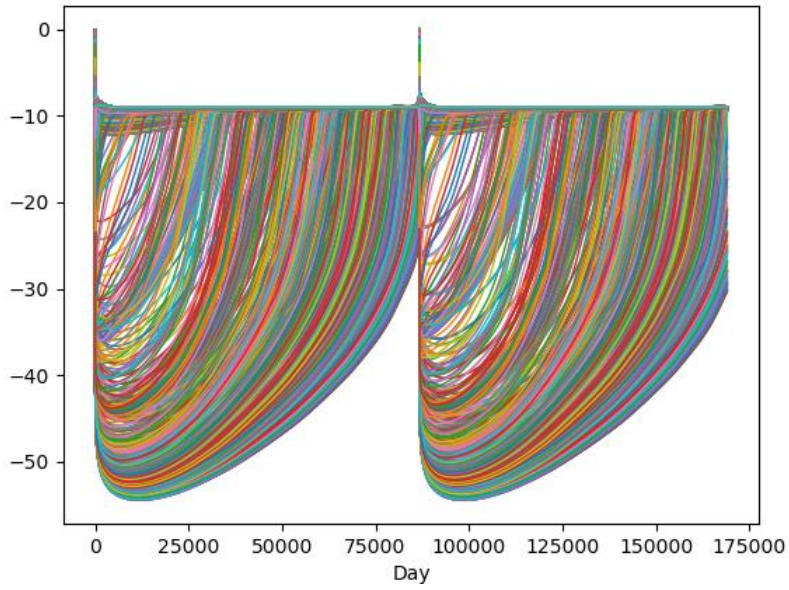


The first rupture occurs due to the initial strong criticality. However, no earthquake recurrence is simulated since we removed the loading faults. Instead of applying the loading fault, we can

achieve the same effect by applying a direct long-term slip rate to each fault. This is often called back-slip. We can apply backslip loading by *Tools/ExternalStress/UniformBackslip.jl*

(5) Run *Tools/ExternalStress/UniformBackslip.jl*

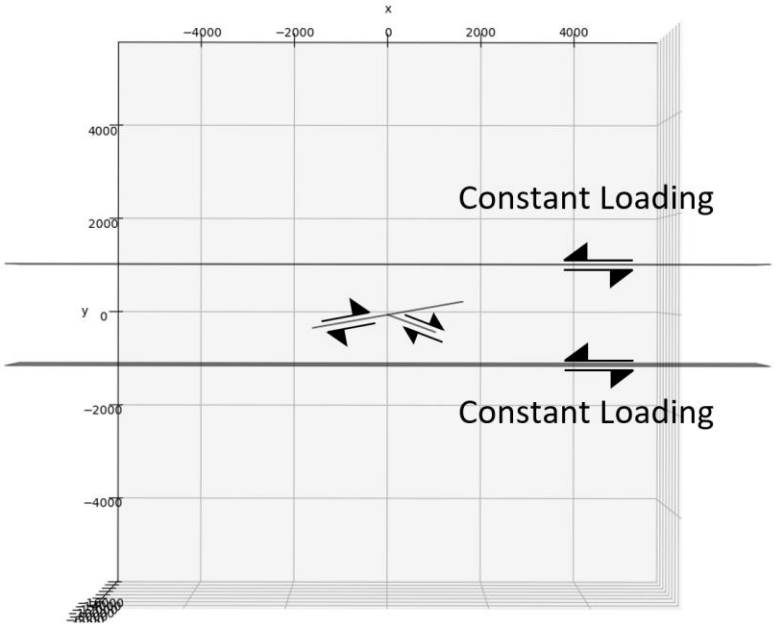
Plot *Results/1_2DPlot.jl*



Now we have cyclic earthquakes. The result is identical to the original simulation with loading faults (section 3.1).

3.2 Two-Fault System with Constant Loading

(1) Run *Tools/InputGeometryExamples/ Example2_BuildGeometry_TwoFaults.jl*



Input_BulkFaultGeometry.txt

```
SwitchSS/RN ShearMod    PoissonRatio    R_Density   Crit_TooClose   TooCloseNormal_Multiplier   Minimum_NS
1.0 2.0e10  0.2 2400.0  1.05    0.6 2.0e6
Ctr_X   Ctr_Y   Ctr_Z   St_L    Dip_L   StAng   DipAng  LR/RN   a   b   Dc  Theta_i V_i Fric_i  Sig0    SigGrad V_Const MaxLeng
738.605814759156    130.23613325019778  1500.0  1500.0  3000.0  10.0    90.0    1.0 0.003   0.006   0.0005  1000.0  1.0e-9  0.6 2.0e6   6000.0  0.0 200.0
-738.605814759156   -130.23613325019778 1500.0  1500.0  3000.0  10.0    90.0    1.0 0.003   0.006   0.0005  1000.0  1.0e-9  0.6 2.0e6   6000.0  0.0 200.0
469.8463103929542   -171.01007166283435 1500.0  1000.0  3000.0  160.0   90.0    1.0 0.003   0.006   0.0005  1000.0  1.0e-9  0.6 2.0e6   6000.0  0.0 200.0
0.0 1000.0  3500.0  15000.0 7000.0  0.0 90.0    -1.0    0.01    0.005   0.001   1.0e6   1.0e-9  0.6 3.0e6   0.0 1.0e-9  1.0e10
0.0 -1000.0 3500.0  15000.0 7000.0  0.0 90.0    -1.0    0.01    0.005   0.001   1.0e6   1.0e-9  0.6 3.0e6   0.0 1.0e-9  1.0e10
```

Input_BulkFaultGeometry.txt shows that the two horizontal faults have constant velocities. If constant velocity is assigned, the faults will only slip at the velocity, with all the other parameters ignored. These two faults are the loading fault that applies constant loading to the two smaller faults in between. These loading faults do not need to be discretized, hence a very large MaxLeng value is assigned.

While the fault geometry appears to have only two faults in between the loading faults, the non-loading faults are three pieces in the Input_BulkFaultGeometry.txt. This is due to the "joint separation" discussed in figure 4.

(2) Run *RUN1_Hmatrix_Structure.jl*

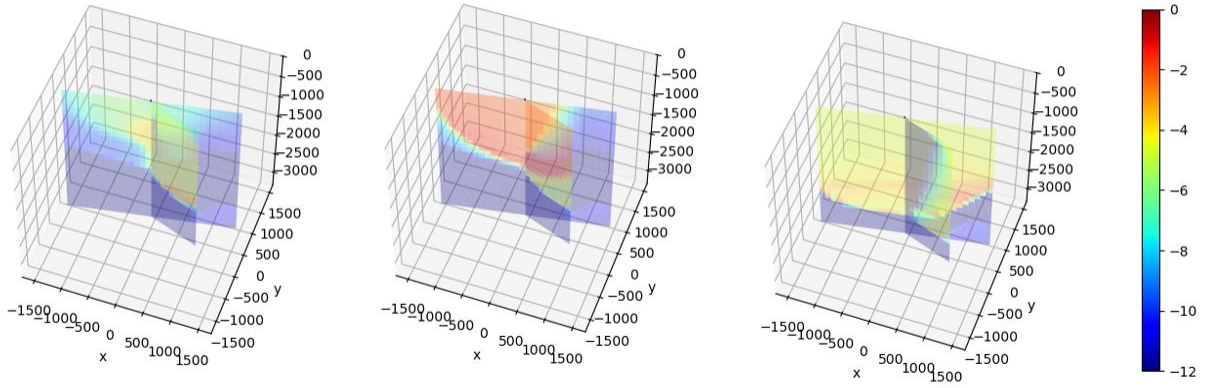
(3) Run *RUN2_BUILD_INPUT.jl*

(4) Run *RUN3_QUAKEDFN.jl*

```
########################## Simulation Time Set ###################################
TotalStep = 10000 # Total simulation step
SaveStep = 10000 # Automatically saved every this step
RecordStep = 10 # Simulation sampling rate
```

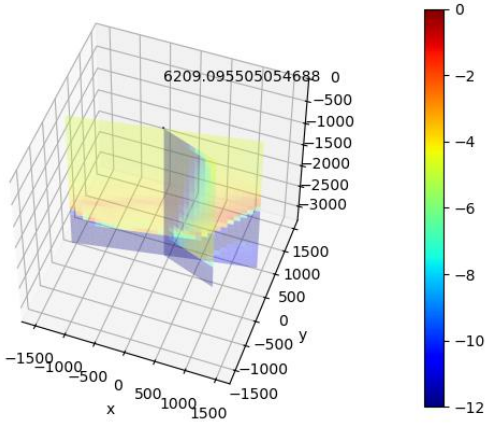
Simulation results (*Results/2_3DPlot_and_Animation.jl*)

Second Rupture



One can turn on the plot time by setting

```
ShowDay = 1 # If 1, day is shown in the location
DayLocation = [0,0,1000]
```



The rupture occurred on day 6209.

3.3 Single fault with a pressure source

(1) Run *InputGeometryExamples/Example3_BuildGeometry_Single_StrikeSlip.jl*

(2) Run *RUN1_Hmatrix_Structure.jl*

(3) Run *RUN2_BUILD_INPUT.jl*

(4) Run *Tools/ExternalStress/PressureCalculation_Rudnicki_RectangleOnly.jl*

```
FlowRate=100 # kg/s
PressureOrigin=[0.0, 0.0,-2000]; # Custom Faults
Permeability = 1e-16;
Viscosity = 0.4e-3;
SkemptonCoeff=0.75;
PoissonRatio_Undrained=0.3;
FluidDensity_Ref = 1e3;
```

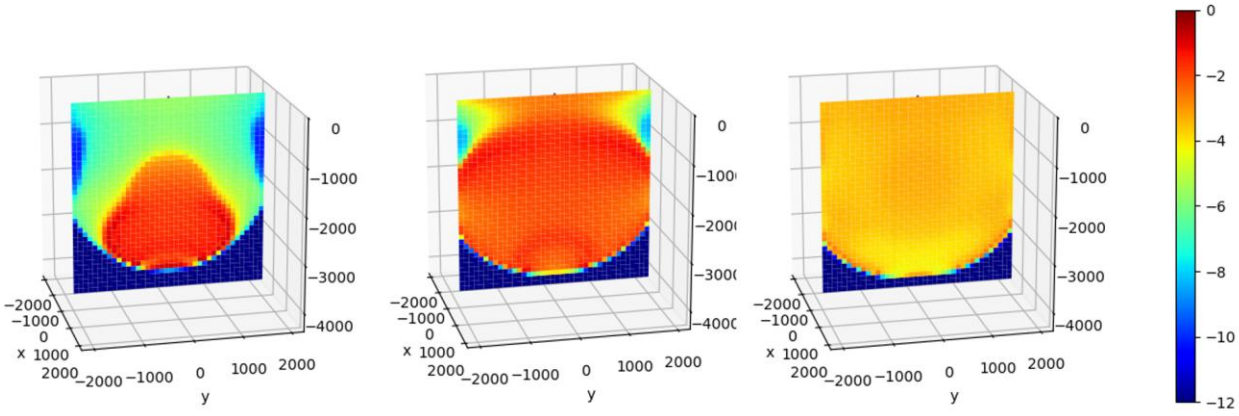
This applies a spherical pressure source to the center of the fault. Shear and normal stress change plot will show up at the end of the simulation.

(5) Run *RUN_QUAKEDFN.jl*

```
########################## Simulation Time Set ################################
TotalStep = 10000 # Total simulation step
SaveStep = 10000 # Automatically saved every this step
RecordStep = 10 # Simulation sampling rate
```

Simulation results

Sequence of the second rupture

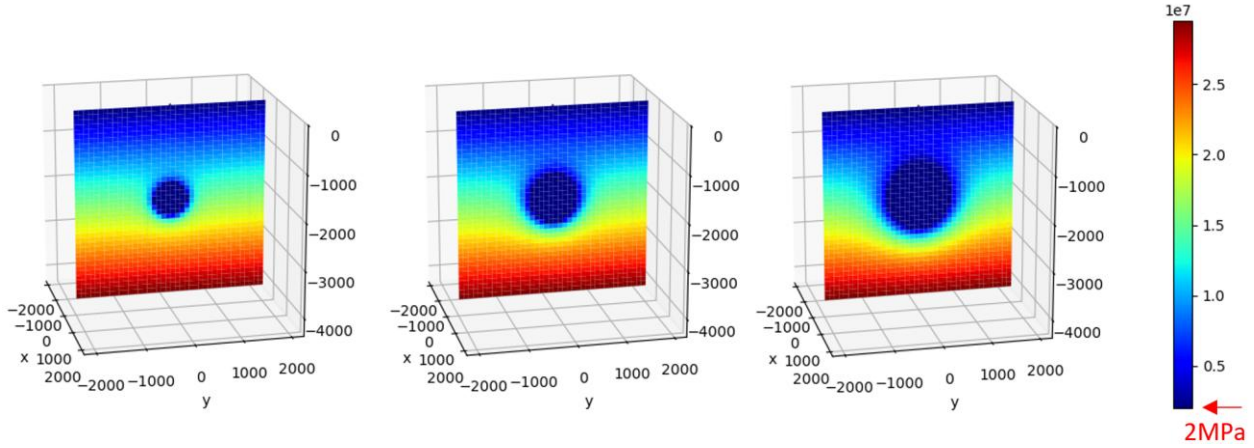


Normal Stress Plot

In *Results/2_3DPlot_and_Animation.jl*,

```
################### What to Plot ? ####################
# PlotInput=log10.(ResultV[PlotStep,:]); ColorMinMax=[-12,0]
PlotInput= Result_NormalStress[PlotStep,:] ; ColorMinMax=0
# PlotInput=log10.(ResultPressure[PlotStep,:]); ColorMinMax=[3,6]
# PlotInput= Result_NormalStress[PlotStep,:] -  Fault_NormalStress;
ColorMinMax=[-1e6,1e6]
# PlotInput=ResultDisp[PlotStep,:]; ColorMinMax=0
```

Plotstep: 200, 400, 600



Effective normal stress decreases due to the pressurization. Note that the minimum normal stress is 2Mpa due to the "(7) Minimum_NS" in Input_BulkFaultGeometry.txt (section 2.1).

Extra: heterogeneous fault properties

(1) Run *InputGeometryExamples/Example3_BuildGeometry_Single_StrikeSlip.jl*

(2) Run *RUN1_Hmatrix_Structure.jl*

(3) Run *RUN2_BUILD_INPUT.jl*

(4) Run *Tools/ExternalStress/PressureCalculation_Rudnicki_RectangleOnly.jl*

(5) Open *QuickParameterAdjust.jl* and put the following and save.

```
#########################################################################
########################### Direct Adjust ###############################
for i=1:FaultCount
   if 500 > FaultCenter[i,3] || FaultCenter[i,3] > 3000
     Fault_a[i] = 0.01
   end
end
###^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^###
```

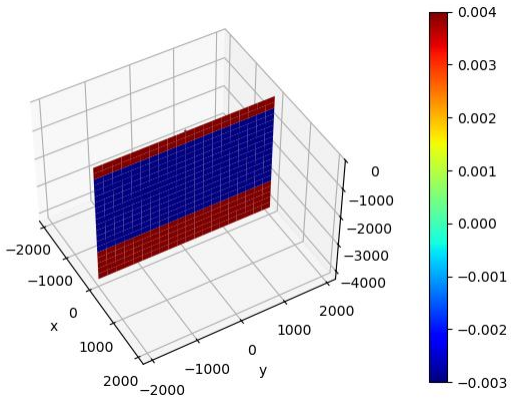
```
##########################################################################
```

This changes the rate and state 'a' parameter and will make the fault stable in shallow (<500m) and deep (>3000m) areas.

(4-1) *Optional* Run *Plot_Discretized_Input.jl* to check a-b

```
    ################## Which parameter want to plot? ########################
        ColorMinMax = 0
        # PlotInput = log10.(Fault_Theta_i); ColorMinMax = 0
        # PlotInput = log10.(Fault_V_i); ColorMinMax = 0
        # PlotInput =Fault_NormalStress; ColorMinMax = 0
        # PlotInput =KoverKC ;ColorMinMax=[0,5]
        # PlotInput =UnderResolved ;ColorMinMax=[0,1]
        PlotInput = Fault_a - Fault_b; ColorMinMax = 0
        # PlotInput =  Fault_BulkIndex; ColorMinMax = 0
```

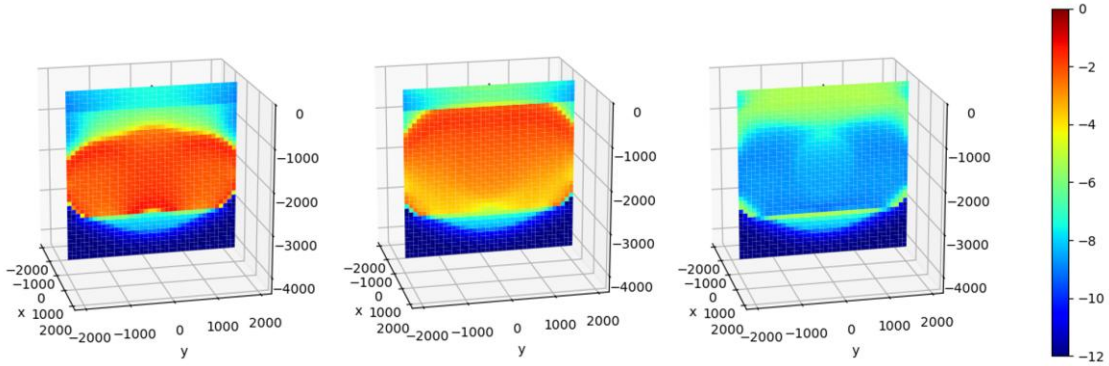
This will show *a-b* value as follows



(5) Run *RUN_QUAKEDFN.jl*

Simulation results

Plots: sequence of the second rupture



Rupture propagation is blocked at shallow (<500m) and deep (>3000m) areas.

3.4 Ridgecrest Earthquake

We provide a simplified Ridgecrest earthquake rupture geometry in the folder *Tools/InputGeometryExamples*.

(1) Move *Tools/InputGeometryExamples/Input_BulkFaultGeometry_Ridgecrest.txt* to the root directory and change the name to *Input_BulkFaultGeometry.txt*.

(2) Run *RUN1_Hmatrix_Structure.jl*

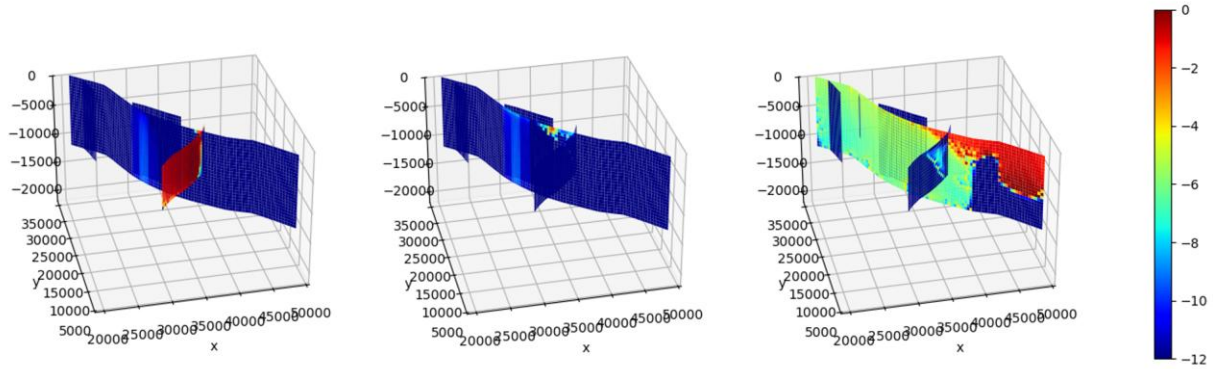
(3) Run *RUN2_BUILD_INPUT.jl*

(4) Run *RUN3_QUAKEDFN.jl*

```
########################## Simulation Time Set ##############################
TotalStep = 15000 # Total simulation step
SaveStep = 5000 # Automatically saved every this step
RecordStep = 10 # Simulation sampling rate !! should be a factor of SaveStep !!
```

Simulation results

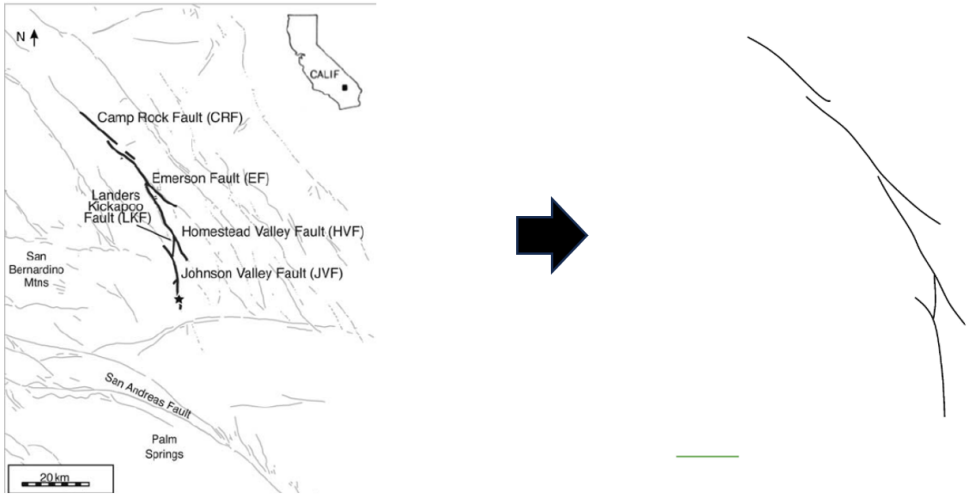
Plot Velocity



3.5 Complex fault geometry – Complex fault network in homogeneous stress field

Quake-DFN offers a useful tool for simulating earthquake rupture in actual fault geometries by converting JPG images into input files for vertical faults. While we are using PowerPoint for this process, any graphic tool can suffice for generating images.

(1) The initial step involves creating a JPG file that depicts the fault geometry. Utilizing graphic tools in PowerPoint, one can trace an image of the surface rupture from the Landers earthquake (Olsen et al., 1997), as demonstrated in the right image below. Note that the green horizontal line will be used as a scale initially set at 10 km. The right image then should be saved as "*Faultimage.jpg*" and stored in the *Tools/ImageReader* folder in Quake-DFN.



When the image is ready, run *Fault_segmentation_entire_process.py* first. Next, execute BuildGeometry_givencoordinates_shallowfault.jl. An input file, *InputFaultGeometry.jl*, will be created in the Quake-DFN root folder. You can verify the built geometry by running *Plot_BulkFaultGeometry.jl*.

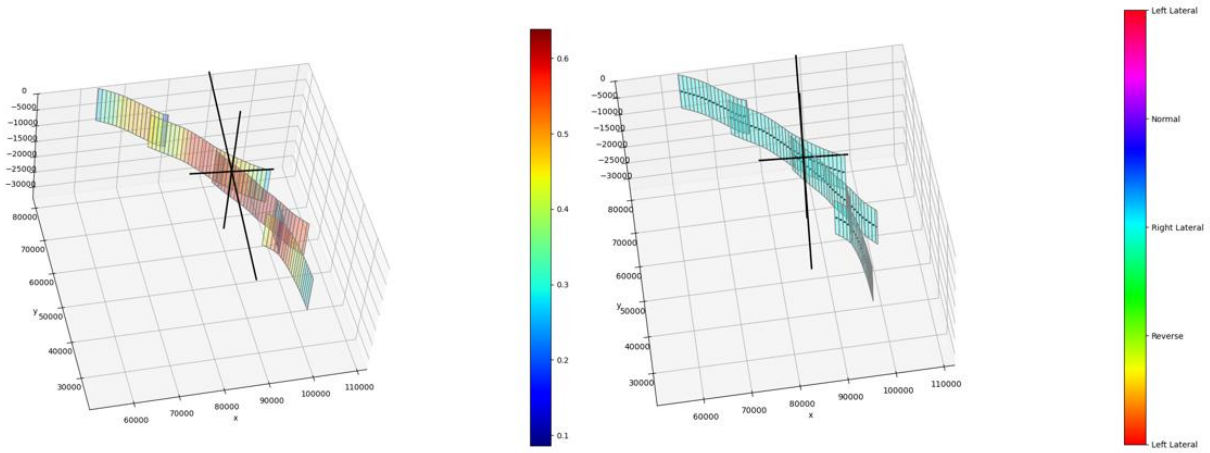
(2) Run *Tools/StressApplytoBulkGeometry.jl*.

```
######################################## Inputs ##################################
###### build Principal Stress. Compression Positive. Only Ratio Matters! ########
    PrincipalStressRatioX = 0.3
    PrincipalStressRatioY = 1.0
    PrincipalStressRatioZ = 0.6
    StressRotationStrike = -5 # degree
    StressRotationDip = 0   # degree

    MaximumTargetVelocity = 1e-11
    ConstantMu0 = 0.0
    ConstantTheta = 1e10
```

```
Fault_a = 0 # if not zero, RSF "a" value will be revised to this uniformly
Fault_b = 0 # if not zero, RSF "b" value will be revised to this uniformly
Fault_Dc = 0 # if not zero, RSF "Dc" value will be revised to this uniformly

MinFrictionAllowed = 0.05
ShearModulus = 30e9
PoissonRatio = 0.25
Rock_Density = 2700.0
MinimumNormalStressAllowed = 1e6
StressOnSurface_Sig1Orientation = 2e6 # pascal
StressGredient_Sig1Orientation = 6000 # pascal/m

FaultSegmentLength = 0
LoadingFaultInvert = 1
```

By setting `MaximumTargetVelocity = 1e-11`, the initial slip velocity of the highest friction element is set to 10^{-11} m/s. And μ_0 in rate and state friction is calculated for that element. This μ_0 is then applied to all the other elements. The initial velocity of the other elements are calculated accordingly. Given that the initial velocity is high ($V\Theta/D_c$ >1), the highest friction element is critically stressed (rupture will be initiated without any loading)

The applied stress field calculates initial friction and a sense of slip.



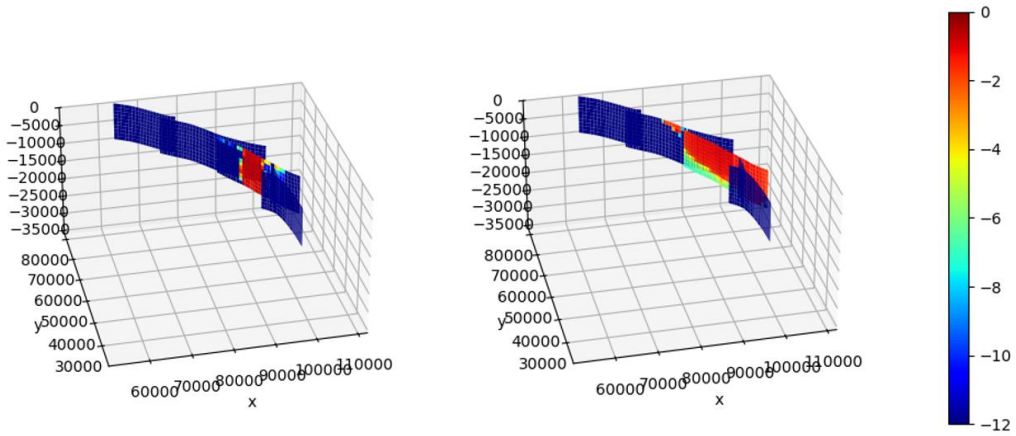
(3) Run *RUN1_Hmatrix_Structure.jl*

(4) Run *RUN2_BUILD_INPUT.jl*

(5) Run *RUN3_QUAKEDFN.jl*

```
########################## Simulation Time Set ##############################
```

```
TotalStep = 5000 # Total simulation step
SaveStep = 5000 # Automatically saved every this step
RecordStep = 10 # Simulation sampling rate !! should be a factor of SaveStep !!
```

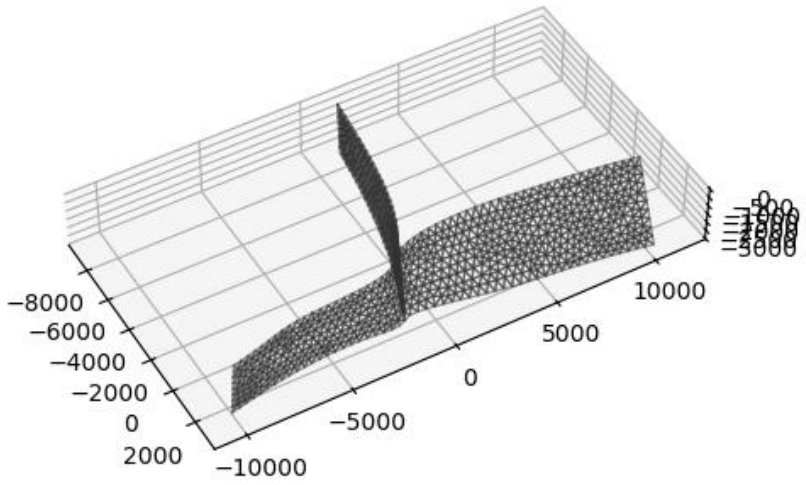


3.6 Triangular Meshing - Triple Junction

Triangular meshing is useful because it can accommodate complex geometries. Building such complex geometry may be done by external software. Quake-DFN provides a tool to read such externally built geometry (obj) files. This code and some examples are in *Tools/TriangleMeshReader*.

(1) Run *Tools/TriangleMeshReader/ObjReader.jl* with the following file name in the code.

```
FileName = "TripleJunction.obj"
```



The *TripleJunction.obj* is a meshed geometry file build in free CAD software (FreeCAD).

(2) Run *Tools/StressApplytoBulkGeometry.jl* with the following input

```
    ##########################################################################
    ######################### Inputs ##################################
    ###### build Principal Stress. Compression Positive. Only Ratio Matters!
    PrincipalStressRatioX = 0.3
    PrincipalStressRatioY = 1.0
    PrincipalStressRatioZ = 0.4
    StressRotationStrike = 40 # degree
    StressRotationDip = 10   # degree

    MaximumTargetVelocity = 1e-11 # if this is non-zero, the maximum velocity is
set to this value. And Mu0 will be adjusted accordingly.
    ConstantMu0 = 0.0
```

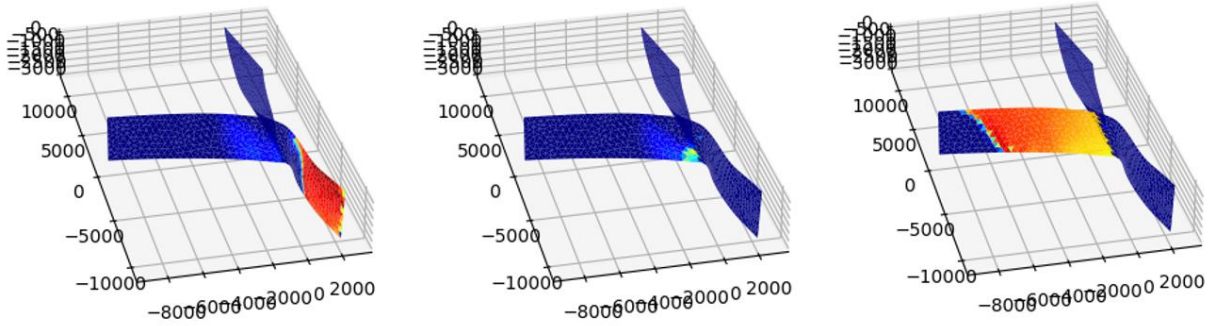
```
    ConstantTheta = 1e10 # if not zero, initial theta will be revised to this
uniformly
    Fault_a = 0 # if not zero, RSF "a" value will be revised to this uniformly
    Fault_b = 0 # if not zero, RSF "b" value will be revised to this uniformly
    Fault_Dc = 0 # if not zero, RSF "Dc" value will be revised to this uniformly

    MinFrictionAllowed = 0.05
    ShearModulus = 30e9
    PoissonRatio = 0.25
    Rock_Density = 2700.0
    MinimumNormalStressAllowed = 1e6
    StressOnSurface_Sig1Orientation = 2e6 # pascal
    StressGredient_Sig1Orientation = 6000 # pascal/m

    FaultSegmentLength = 0 # if 0, segment length will be unchanged (Only For
Rectangular Grid)
    LoadingFaultInvert = 1 # if 1, loading fault sense of slip becomes inverted
```

(3) Run *RUN1_Hmatrix_Structure.jl*.

(4) Run *RUN2_BUILD_INPUT.jl*.

(5) Run *RUN3_QUAKEDFN.jl*.

Result Velocity.



References

Dieterich, J. H. (1979). Modeling of rock friction: 1. Experimental results and constitutive equations. *J. Geophys. Res.*, *84*(9), 2161–2168.

Hackbusch, W. (1999). A sparse matrix arithmetic based on-matrices. Part I: Introduction to-matrices. *Computing*, *62*(2), 89–108

Im, K., & Avouac, J.-P. (2021). Tectonic tremor as friction-induced inertial vibration. *Earth and Planetary Science Letters*, *576*, 117238. https://doi.org/10.1016/j.epsl.2021.117238

Im, K., Avouac, J.-P. (2024). Quake-DFN: A Software for Simulating Sequences of Induced Earthquakes in a Discrete Fault Network. https://doi.org/10.1785/0120230299

Jiang, J., Erickson, B. A., Lambert, V. R., Ampuero, J., Ando, R., Barbot, S. D., Cattania, C., Zilio, L. D., Duan, B., & Dunham, E. M. (2022). Community-driven code comparisons for three-dimensional dynamic modeling of sequences of earthquakes and aseismic slip. *Journal of Geophysical Research: Solid Earth*, *127*(3), e2021JB023519.

Marone, C. (1998). Laboratory-Derived Friction Laws and Their Application To Seismic Faulting. *Annual Review of Earth and Planetary Sciences*, *26*(1), 643–696. https://doi.org/10.1146/annurev.earth.26.1.643

Nikkhoo, M., & Walter, T. R. (2015). Triangular dislocation: an analytical, artefact-free solution. *Geophysical Journal International*, *201*(2), 1119–1141. https://doi.org/10.1093/gji/ggv035

Okada, Y. (1992). Internal deformation due to shear and tensile faults in a half-space. *Bulletin of the Seismological Society of America*, *82*(2), 1018–1040.

Rice, J. R. (1993). Spatio-temporal complexity of slip on a fault. *J. Geophys. Res.*, *98*(B6), 9885. https://doi.org/10.1029/93JB00191

Rudnicki, J. W. (1986). Fluid mass sources and point forces in linear elastic diffusive solids. *Mechanics of Materials*, *5*(4), 383–393.

Segall, P., & Lu, S. (2015). Injection-induced seismicity: Poroelastic and earthquake nucleation effects. *Journal of Geophysical Research: Solid Earth*, *120*(7), 5082–5103.