

Quake-DFN

beta version

User's Guide (draft)

Kyungjae Im
November 2023

Table of Contents

Installation and test

Installation guide

Test simulation

1. Quake-DFN Introduction

1.1 Brief workflow

1.2 Theoretical background

2. Workflow

2.1 Required Input File: *Input_BulkFaultGeometry.txt*

2.2. Discretize the fault and build stiffness matrices: *RUN_BUILD_INPUT.jl* and *Input_Discretized.jld2*

2.3. Detailed parameter adjustment for discretized elements:
QuickParameterAdjust.jl

2.4. External Stress change: *Input_ExternalStressChange.jld2*

2.4.1 Example – Spherical pressurization:
ExternalStressCalculation/PressureCalculation_Rudnicki.jl

2.5. Conduct Simulation: *RUN_QUAKEDFN.jl*

3. Example Simulations

3.1 BP5QD (SEAS Benchmark)

3.2 Two-Fault System with constant loading

3.3 Single fault with a pressure source

3.4 Two strike-slip faults with a pressure source

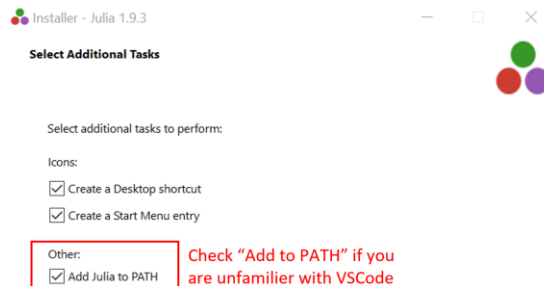
3.5 Two normal faults with a pressure source


Installation and Test

Installation guide

Quake-DFN is written in Julia. VSCode IDE is recommended as it is coded and tested in the VSCode. The required Julia packages are listed below. Note that we need to install the older version of the Matplotlib (3.4.3) Use the following steps.

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



- 2) Open the 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:

In Julia REPL, press “`]`”, then the prompt will become “`(xxx) pkg>`”. At the prompt,

```
add PyPlot                # (this may take a while)
add PyCall
build PyCall
add Conda
add DelimitedFiles
add JLD2
add LinearAlgebra
add Printf
add SpecialFunctions
add HMatrices
add StaticArrays
```

- 5) install Matplotlib

Back to REPL (press “BackSpace”)

```
using PyPlot              #(this may take a while)
using PyCall
using Conda
Conda.add("Matplotlib==3.4.3")    #(this may take a while)
```

- 6) 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 the Quake-DFN and open the QuakeDFN folder in VSCode (file → open folder in VSCode). The root folder has three sub-folders, nine ‘*.jl’ files, one txt file, and this manual.

The txt file (*Input_BulkFaultGeometry.txt*) contains the input file of BP5QD benchmark problem (Jiang et al., 2022). This file alone is sufficient to conduct simulations. The simulation can be implemented by running *RUN_Build_InputFile.jl* and then *RUN_QUAKEDFN.jl*. The result is automatically saved in the “results” folder. The result can be visualized by running *Results/2_3DPlot_Animation.jl* (ensure PlotStep is within the total recorded step).

More detail of the test simulation

The input txt file (*Input_BulkFaultGeometry.txt*) contains the rock properties and fault geometry of 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. The color code of the fault is set to present slip orientation (blue: left lateral, red: right lateral).

The faults can be discretized by running the “*RUN_BUILD_INPUT.jl*”. Once run, it will generate a new 3D geometry plot in which the fault is discretized (figure 1b). The large loading faults are not discretized since there is no benefit as they only slip at constant velocity. Running the *RUN_BUILD_INPUT.jl* generates a file name *Input_Discretized.jld2*. This file is the actual input file that contains the stiffness matrices and initial parameters. Note that the friction parameters and initial conditions can be re-adjusted even after building this input file. But if fault geometry needs to be changed, *Input_Discretized.jld2* should be rebuilt.

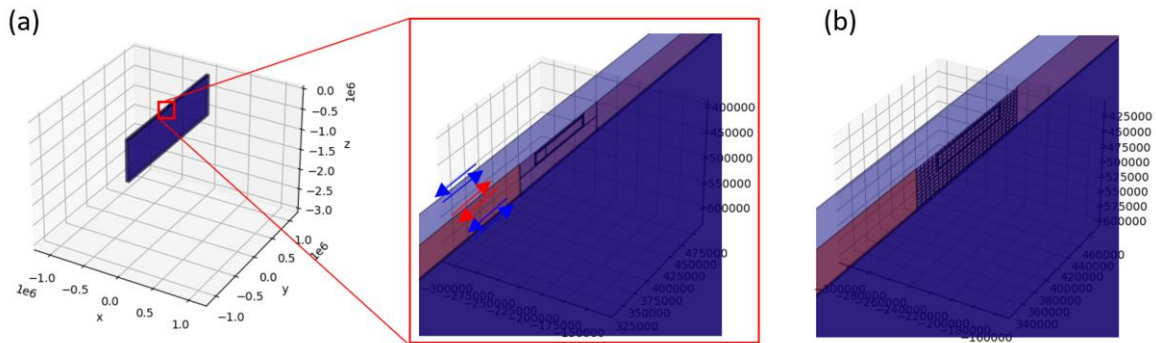


Figure 1. Test simulation geometry. (a) un-discretized fault geometry (*Plot_BulkFaultGeometry.jl*). (b) after discretization (*RUN_BUILD_INPUT.jl*)

Now simulation can be conducted by running the *Run_QUAKEDFN.jl*. Once run, the timestep plot will pop up, which defines the simulation time step size at a given maximum velocity. The time step skim can be redefined in the *Run_QUAKEDFN.jl* (see section 2.5).

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 displacement 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_Animation.jl* presents a 3D snapshot of the simulation result (default: velocity) at a given recorded step. Figure 2 shows the result of with PlotStep = 100, 200, and 300. Note that this simulation does not precisely reproduce the BP5-QD simulation as the grids are coarser. A more precise BP5QD simulation can be done by adjusting the maximum grid length in the *Input_BulkFaultGeometry.txt* (change 4000.0 to 1000.0)

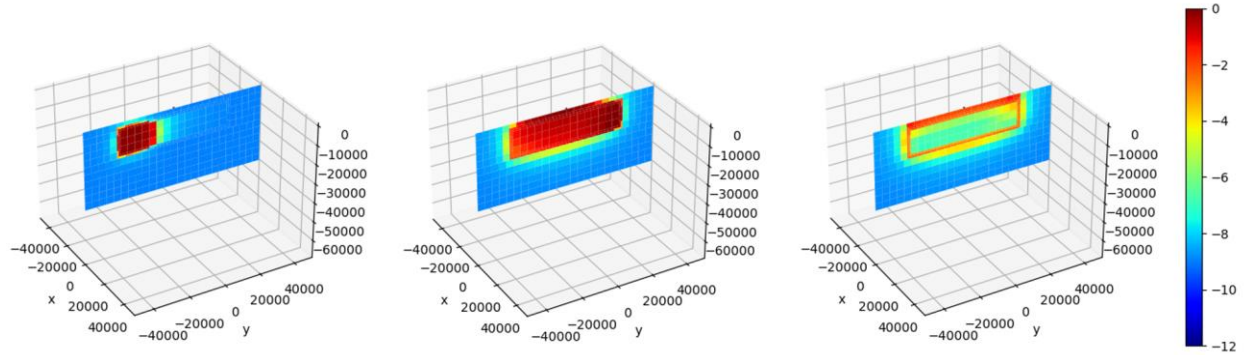


Figure 2. Simulation results plotted by *Result/2_3Dplot_Animation.jl* with PlotStep = 100, 200, and 300.

1. Quake-DFN Introduction

Quake-DFN is the boundary element simulator developed for earthquake rupture simulation of discretely distributed faults governed by rate and state friction law. The simulator formulation is consistent with the widely used quasi-dynamic formulation with radiation damping but with a lumped mass that represents overshoot effects.

1.1 Brief workflow

The simulator works with one essential input that defines un-discretized fault geometry and two supplementary input files that defines detailed parameter in the discretized fault geometry and external stress. The three input files are listed below.

- 1) *Input_BulkFaultGeometry.txt*: This is the essential input file that defines undiscretized fault geometry. This can be discretized by running *RUN_BUILD_INPUT.jl*. (See section 2.1)
- 2) *QuickParameterAdjust.jl*: This file reads the discretized input (*Input_Discretized.jld2*) and change parameters before the simulations. Since it works on the discretized elements, it can apply detailed heterogeneity. (See section 2.3)
- 3) *Input_ExternalStressChange.jld2*: This input defines external shear and normal stress change. It can be any external loading or injection-induced stress change. (See section 2.4)

The simulation has two steps. (1) discretize the input file and (2) run the simulation. The two RUN files are listed below

- 1) *RUN_BUILD_INPUT.jl*: Discretize the *Input_BulkFaultGeometry.txt* and generate the *Input_Discretized.jld2*, which is actual input file for the simulation.
- 2) *RUN_QUAKEDFN.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. Some codes for 2D and 3D plots are provided in the folder.

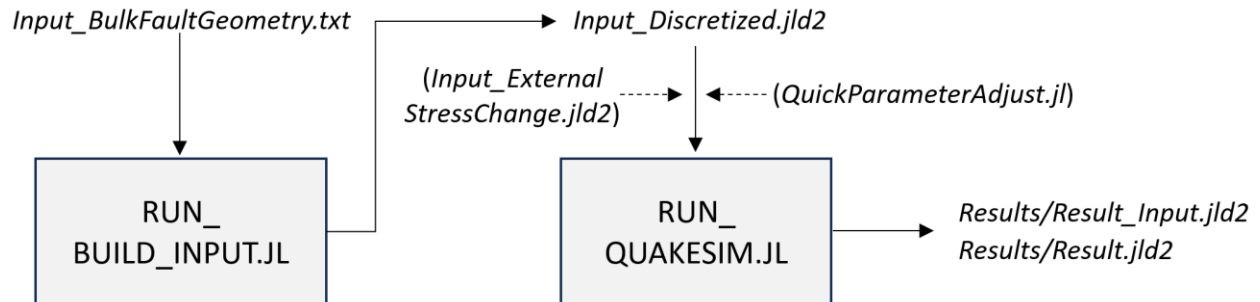


Figure 3. Simulation workflow chart.

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

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

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)

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

with aging state evolution law

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

where V is velocity, θ is the state variable, μ_0 is a reference friction coefficient at reference velocity V_0 , D_c is a critical slip distance, and a and b are empirical constants for the magnitude of direct and evolution effects, respectively.

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 in a finite size of 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.

In the current simulator, $M = 10^6 \text{kg/m}^2$, which roughly represents the overshoot of 400m high rock mass ($\sim M/\rho$). This can be adjusted in the *QuickParameterAdjust.jl*.

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). We define

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

To achieve good resolution, the k_{ii}^{τ}/K_c should be sufficiently larger than 1. In Quake-DFN, this can be checked before running the simulation. Once faults are discretized, run *Plot_Input.jl* with un-commenting *PlotInput=KoverKC* and comment 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 first five lines of the test simulation are as follows.

```

Line 1 → 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
Line 3 → 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 4000.0
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 4000.0
0.0 21500.0 10000.0 1000.0 16000.0 0.0 0.0 1.0 0.021000000000000002 0.02 0.14 1.4e8 1.0e-9 0.6 2.5e7 0.0 0.0 4000.0

```

The first and third lines (texts) are the index of input parameters for the input numbers of second and fourth (and after) lines, respectively. First lines indicates

- | | |
|-------------------------------|---|
| (1) SwitchSS/RN | Indicate if the fault system is Strike-Slip (1) or Normal-Reverse (2) |
| (2) ShearMod | Shear modulus [Pa] |
| (3) PoissonRatio | Poisson's ratio [-] |
| (4) R_Density | Rock Density [kg/m ³] |
| (5) Crit_TooClose | Criteria for “too close” |
| (6) TooCloseNormal_Multiplier | Multiplier for normal interaction for the “too close” calculation |
| (7) Minimum_NS | Minimum normal stress allowed in the simulation |

The index (5) and (6) are introduced for stability. Stability is influenced by the discretization of discrete fault systems. In particular, if a fault tip is too close to the center of another element (e.g., figure 4a purple circle), the interaction becomes too strong since fault tip stress is nearly infinite. The simulator estimates this potential instability by calculating

$$\frac{k_{ij}^{\tau} - \mu k_{ij}^{\sigma}}{k_{ii}^{\tau}} \cdot \frac{k_{ji}^{\tau} - \mu k_{ji}^{\sigma}}{k_{jj}^{\tau}} \quad (6)$$

If this value is greater than the “(5) Crit_TooClose”, the simulator remove one of the fault element for stability. The “(6) TooCloseNormal_Multiplier” is μ in equation 6.

The strong interaction problem can be reduced by separating faults in *Input_BulkFaultGeometry.txt* at intersections, as shown in Figure 4b, so that any element center cannot located at intersections.

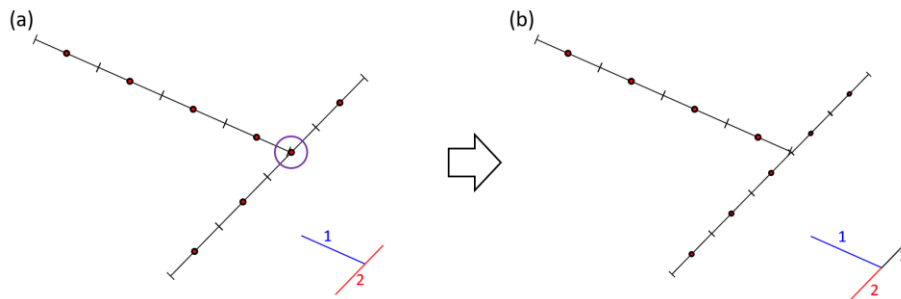


Figure 4. (a): Strong interaction example. (b) strong interaction is avoided. (b) can be achieved in *Input_BulkFaultGeometry.txt* by separating faults at the intersection (bottom right figure in b).

The indexes at the third line define fault geometry, frictional properties, and initial state as denoted as follows.

- (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 (always 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) LR/RN Slip direction (Left lateral (-1) right lateral (1) or reverse (-1) normal (1) slip [-])
- (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]

The friction parameters and initial conditions (9-16) can be adjusted after discretization. Conversely, the geometry (1-8 and 18) cannot be adjusted once the stiffness matrix is built (i.e., after running the *RUN_Build_InputFile.jl*).

The geometry parameters (1-8) should be defined as illustrated in figure 5.

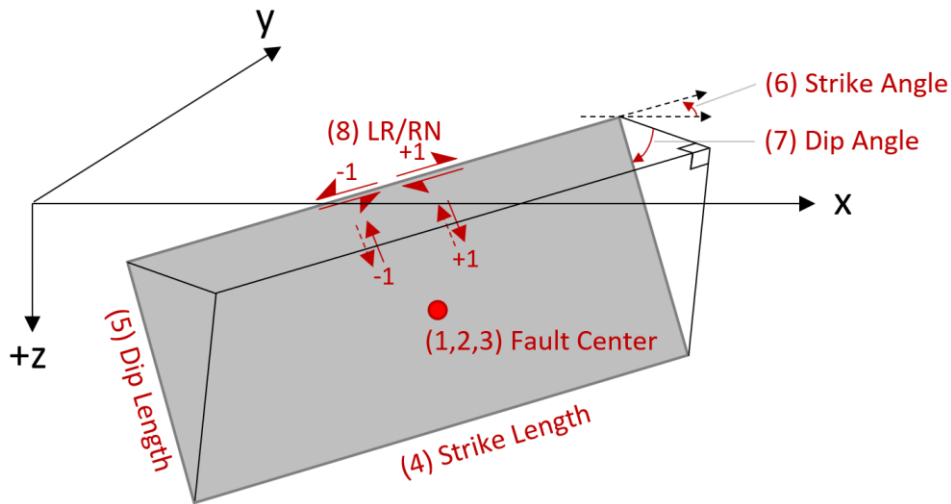


Figure 5. Definitions of the fault geometry parameters.

The initial condition of rate and state friction law is

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

Every variable and initial condition of equation 7 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 7. One may want to set μ_0 as it is a laboratory-measurable quantity. This can be done by using equation 7. For example, if one wants to define μ_0 and make V_i be calculated correspondingly, one can simply calculate

$$V_i = V_0 \exp \left(\left(\mu_i - \mu_0 - b \log \left(\frac{V_0 \theta_i}{D_c} \right) \right) / a \right) \quad (8)$$

and put this initial velocity as a simulation input.

2.2. Discretize the fault and build stiffness matrices: *RUN_BUILD_INPUT.jl* and *Input_Discretized.jld2*

The *RUN_BUILD_INPUT.jl* discretizes the input geometry and builds stiffness matrices based on Okada (1992) formulation. The stiffness matrix defines shear and normal stress change driven by a slip of an element. For N number of elements, one stiffness matrix build requires N×N times of Okada 1992 calculations. This calculation is not yet parallelized, so if N is large, this calculation may take a long time. However, once the stiffness is built for geometry, it can be used for a wide range of parameter sets, as the input parameter and initial condition change do not require re-calculation of the stiffness matrix.

The calculated stiffness matrix and input parameters are saved as *Input_Discretized.jld2*.

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

QuickParameterAdjust.jl can change parameters and initial state at each individual discretized element level. Therefore, complex heterogeneity should be implemented here.

One significant benefit here is that the parameter adjust do not require re-discretization. Sensitivity tests can be easily implemented in this file. Once faults are discretized, one can test a wide range of parameter sets in here. This file is automatically run right before the simulation.

2.4. External Stress change: *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.

ExternalStress_ TimeArray	ExternalStress_ Normal				ExternalStress_ Shear			
	Element 1	Element 2	Element 3	...	Element 1	Element 2	Element 3	...
t_1	$\Delta\sigma(t_1)$	$\Delta\sigma(t_1)$	$\Delta\sigma(t_1)$...	$\Delta\tau(t_1)$	$\Delta\tau(t_1)$	$\Delta\tau(t_1)$...
t_2	$\Delta\sigma(t_2)$	$\Delta\sigma(t_2)$	$\Delta\sigma(t_2)$		$\Delta\tau(t_1)$	$\Delta\tau(t_1)$	$\Delta\tau(t_1)$	
t_3	$\Delta\sigma(t_3)$	$\Delta\sigma(t_3)$	$\Delta\sigma(t_3)$		$\Delta\tau(t_1)$	$\Delta\tau(t_1)$	$\Delta\tau(t_1)$	
t_4	$\Delta\sigma(t_4)$	$\Delta\sigma(t_4)$	$\Delta\sigma(t_4)$		$\Delta\tau(t_1)$	$\Delta\tau(t_1)$	$\Delta\tau(t_1)$	
t_5	$\Delta\sigma(t_5)$	$\Delta\sigma(t_5)$	$\Delta\sigma(t_5)$		$\Delta\tau(t_1)$	$\Delta\tau(t_1)$	$\Delta\tau(t_1)$	
.	.				.			
.	.				.			
.	.				.			

Once the simulation began, the simulator automatically interpolate shear and normal stress change from

2.4.1 *ExternalStressCalculation/PressureCalculation_Rudnicki.jl*

This is an example that generates *Input_ExternalStressChange.jld2*. This file should be run after discretization. Once run, it reads the location and orientation of each discretized element stored in the *Input_Discretized.jld2* file and calculates the normal and shear stress change from spherical pressure diffusion (Rudnicki, 1986; Segall and Lu, 2015). The calculated shear and normal stress are rotated to the fault slip orientation. In the file, we should define several parameters as below.

```
FlowRate=100 # kg/s
PressureOrigin=[0.0, 0.0,-1500]; # Custom Faults
Permeability = 1e-16;
Viscosity = 0.4e-3;
SkemptonCoeff=0.75;
PoissonRatio_Undrained=0.3;
FluidDensity_Ref = 1e3;
```

See Segal and Lu (2015) for more information.

2.5. Conduct Simulation: *RUN_QUAKEDFN.jl*

RUN_QUAKEDFN.jl reads *Input_Discretized.jld2* (and additionally *QuickParameterAdjust.jl* and *Input_ExternalStressChange.jld2* if defined) and run simulation. Once run, it generates the $\log_{10}(dt)$ vs $\log_{10}(V)$ plot. This plot presents the reference timestep size at a given maximum velocity. Simulation time and time step strategy should be defined in this file.

```
##### Simulation Time Set #####
TotalStep = 10000 # Total simulation step
SaveStep = 5000 # Automatically saved every this step
RecordStep = 10 # Simulation sampling rate
```

These variables define total simulation and recording length. The simulation will be conducted for **TotalStep**. The simulator records simulation results in every **RecordStep** in RAM (others will be discarded). The recorded result will be saved to the **Result** folder every **SaveSteps**. In the above example, the simulation will be conducted for 10000 steps, and the simulation progress is saved two times (step 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 result file will contain information of 1000 recorded steps (every 10 simulation steps).

```
##### Time Stepping Setup #####
TimeStepOnlyBasedOnUnstablePatch = 1 # if 1, time step is calculated only based
on the unstable patch
TimeStepPreset = 3 # 1: conservative --> 4: optimistic

# Manually adjust time step below. No change when 0.0
TimeSteppingAdj =
    [0.0 0.0 0.0 0.0; # Time step size
     0.0 0.0 0.0 0.0] # Velocity
```

Time stepping logic (dt velocity dependence) is defined here. Simulation timestep size is mainly dependent on the maximum velocity of all the elements. If

TimeStepOnlyBasedOnUnstablePatch = 1, the time step is only dependent on the unstable ($a-b < 0$) patches. **TimeStepOnlyBasedOnUnstablePatch** = 0 is more stable, but **TimeStepOnlyBasedOnUnstablePatch** = 1 is faster in general.

We set several different scenarios for time stepping. If **TimeStepPreset** = 1, the simulator uses a conservative scenario (smaller step), and if 4, it uses the most optimistic scenario (larger step).

One can manually adjust the time-step skim in the table **TimeSteppingAdj**. The first low represents the step size, and the second low represents the corresponding maximum velocity. This is illustrated in figure 6. Any element with zero uses the preset timestep.

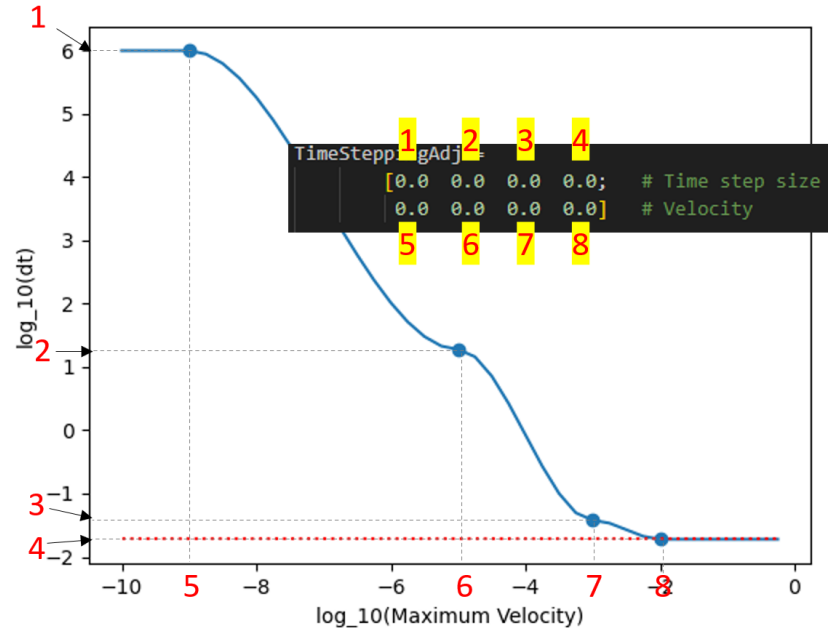


Figure 6. Illustration for *TimeSteppingAdj*.

```
##### H-Matrix ? #####
#####!!!!!! The H-Matrix is still under development !!!!!#####
HMatrixCompress = 0 # 1 for using HMatrix. No HMatrix compression if not 1
HMatrix_eta = 1.0
HMatrix_atol_Shear = 1e-10 # if smaller, better accuracy but slower Calculation
(become unstable if too low)
HMatrix_atol_Normal = 1e-10 # if smaller, better accuracy but slower Calculation
(become unstable if too low)
# The stiffness matrices will be compressed to HMatrix if HMatrixCompress=1.
# We recommend to use only if the element size is larger than 5000
H matrix is still under development. Please keep HmatrixCompress = 0 (no Hmatrix Use).
```

```
##### Plots before run? #####
DtPlot = 1 # 1 will plot dt vs maxV
GeometryPlot = 0 # 1 will plot a-b
AdjustStiffnessPlot = 0 # 1 will plot the elements, of which their stiffness is
changed for stability
```

Setting any of these equals to 1 will give corresponding plot at the beginning of the simulation.

During the simulation, Julia REPL window updates 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 become 1)
- (2) the time in the simulation [day]
- (3) maximum velocity
- (4) timestep size [s]
- (5) Is 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 provided examples in the *InputGeometryExamples* folder. **All 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

(1) Run *InputGeometryExamples/Example1_BuildGeometry_BP5QD.jl*

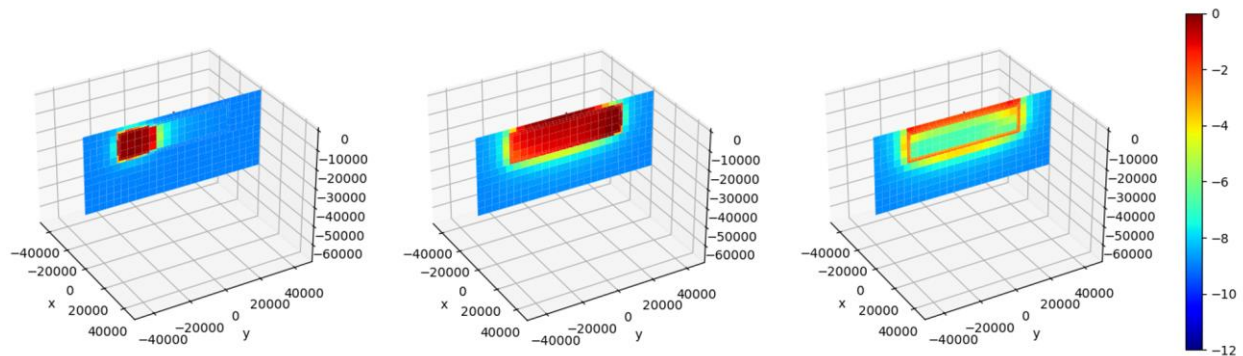
(2) Run *RUN_BUILD_INPUT.jl*

(3) Run *RUN_QUAKEDFN.jl*

```
##### Simulation Time Set #####
TotalStep = 10000 # Total simulation step
SaveStep = 5000 # Automatically saved every this step
RecordStep = 10 # Simulation sampling rate
##### Time Stepping Setup #####
TimeStepOnlyBasedOnUnstablePatch = 1 # if 1, time step is calculated only based
on the unstable patch
TimeStepPreset = 3 # 1: conservative --> 4: optimistic
```

Simulation results (*Results/2_3DPlot_Animation.jl*)

Plotstep: 100, 200, 300



Note that the result does not plot loading faults because they are typically very large. If one want to plot the loading fault, change `LoadingFaultPlot = 1` in *Results/2_3DPlot_Animation.jl*.

To reproduce the BP5QD benchmark test (Jiang et al., 2022) correctly, below two parameters in *InputGeometryExamples/Example1_BuildGeometry_BP5QD.jl* should be changed to 1000.0.

```
UntableMaximumSegLength = 4000.0
StableMaximumSegLength = 4000.0
```


Extra: BP5QD Grid Size Adjustment

- (1) Run *InputGeometryExamples/Example1_BuildGeometry_BP5QD.jl*
- (2) Open the *Input_BulkFaultGeometry.txt* and adjust MaxLeng of first fault (4th line) to 1000.0 as below and save it.

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	4000.0
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	4000.0
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	4000.0

This will make the grid size of the first fault 1000.0.

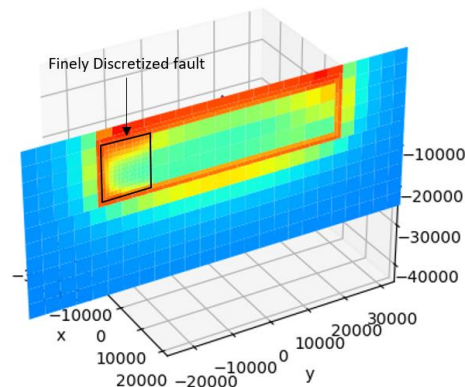
- (3) Run *RUN_BUILD_INPUT.jl*
- (4) Run *RUN_QUAKEDFN.jl*

```
##### Simulation Time Set #####
TotalStep = 10000 # Total simulation step
SaveStep = 5000 # Automatically saved every this step
RecordStep = 10 # Simulation sampling rate

##### Time Stepping Setup #####
TimeStepOnlyBasedOnUnstablePatch = 1 # if 1, time step is calculated only based
on the unstable patch
TimeStepPreset = 3 # 1: conservative --> 4: optimistic
```

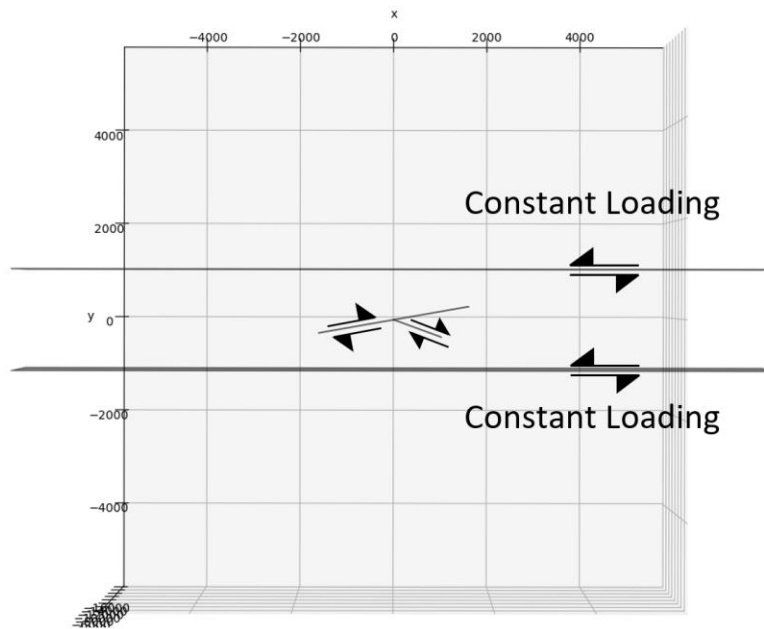
Simulation results (*Results/2_3DPlot_Animation.jl*)

Plotstep: 500



3.2 Two-Fault System with Constant Loading

(1) Run *InputGeometryExamples/ Example2_BuildGeometry_TwoFaults_Loaded.jl*



```
Input_BulkFaultGeometry.txt
```

1	SwitchSS/RN	ShearMod	PoissonRatio	R_Density	Crit_TooClose	TooCloseNormal_Multiplier	Minimum_NS													
2	1.0	2.0e10	0.2	2400.0	1.05	0.6	2.0e6													
3	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		
4	738.605814759156	-130.23613325019778	1500.0	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	
5	-738.605814759156	-130.23613325019778	1500.0	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	
6	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		
7	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		
8	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 does not need to be discretized, hence 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 3 pieces in the *Input_BulkFaultGeometry.txt*. This is due to the “joint separation” discussed in figure 4.

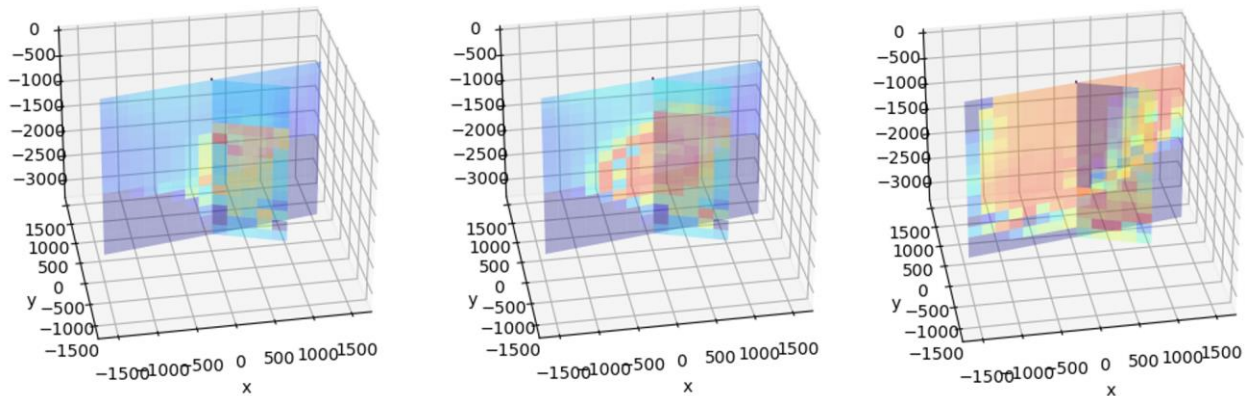
(2) Run *RUN_BUILD_INPUT.jl*

(3) Run *RUN_QUAKEDFN.jl*

```
##### Simulation Time Set #####
TotalStep = 10000 # Total simulation step
SaveStep = 5000 # Automatically saved every this step
RecordStep = 10 # Simulation sampling rate
##### Time Stepping Setup #####
TimeStepOnlyBasedOnUnstablePatch = 1 # if 1, time step is calculated only based
on the unstable patch
TimeStepPreset = 3 # 1: conservative --> 4: optimistic
```

Simulation results (*Results/2_3DPlot_Animation.jl*)

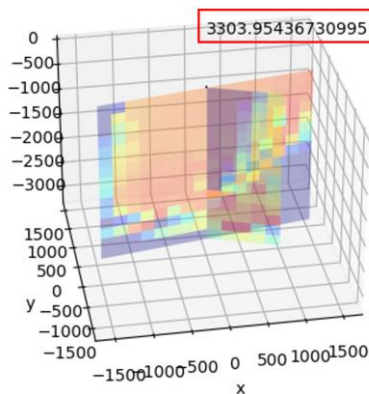
Plotstep: 450, 500, 550 (set “Transparent = 1” for transparent plot)



One can turn on the plot time by setting

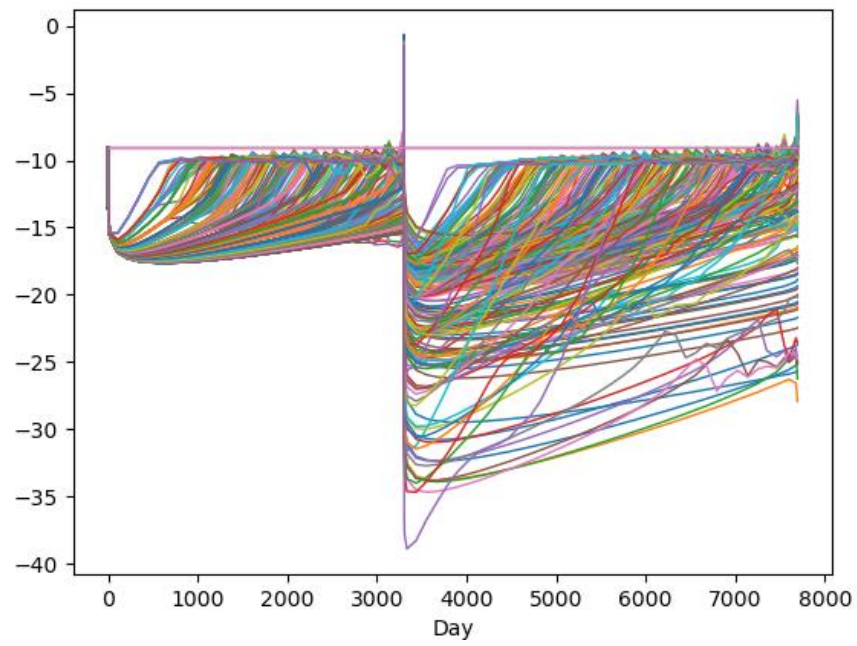
```
ShowDay = 1 # If 1, day is shown in the location
DayLocation = [0,0,1000]
```

Plotstep: 550



Simulation occurred at day 3303.95

The simulation time can be also checked by running *Results/1_2Dplot.jl*. It plots \log_{10} velocity of all elements. The constant velocities at -9 (10^{-9} m/s) is loading faults



3.3 Single fault with a pressure source

(1) Run *InputGeometryExamples/Example3_BuildGeometry_Single_StrikeSlip.jl*

(2) Run *RUN_BUILD_INPUT.jl*

(3) Run *ExternalStressCalculation/PressureCalculation_Rudnicki.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 at the center of the fault. Shear and normal stress change plot will show up at the end of the simulation.

(4) Run *RUN_QUAKEDFN.jl*

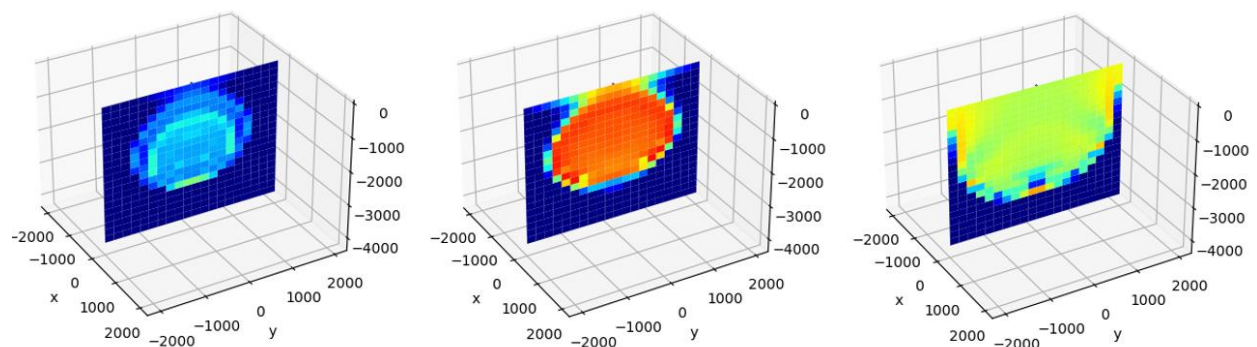
```
##### Simulation Time Set #####
TotalStep = 10000 # Total simulation step
SaveStep = 5000 # Automatically saved every this step
RecordStep = 10 # Simulation sampling rate

##### Time Stepping Setup #####
TimeStepOnlyBasedOnUnstablePatch = 1 # if 1, time step is calculated only based
on the unstable patch
TimeStepPreset = 3 # 1: conservative --> 4: optimistic
```

Simulation results

Velocity

Plotstep: 300, 400, 500

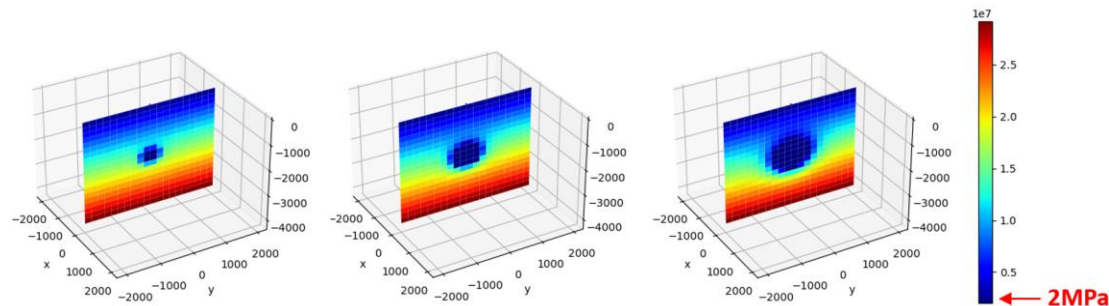


Normal Stress Plot

In *Results/2_3DPlot_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: 100, 300, 1000



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: Fault Property Adjustment

- (1) Run *InputGeometryExamples/Example3_BuildGeometry_Single_StrikeSlip.jl*
- (2) Run *RUN_BUILD_INPUT.jl*
- (3) Run *ExternalStressCalculation/PressureCalculation_Rudnicki.jl*
- (4) Open *QuickParameterAdjust.jl* and put the following and save.

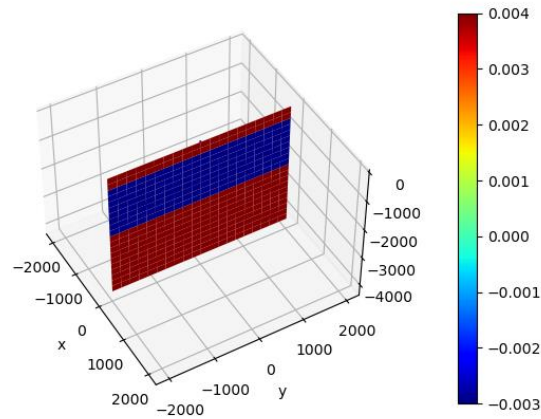
```
#####
##### Direct Adjust #####
for i=1:FaultCount
    if 500 > FaultCenter[i,3] || FaultCenter[i,3] > 2000
        Fault_a[i] = 0.01
    end
end
#####^#####
#####
```

This change the rate and state ‘a’ parameter and will make the fault stable in shallow (<500m) and deep (>2000m) area.

(4-1) Optional Run *Plot_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
```

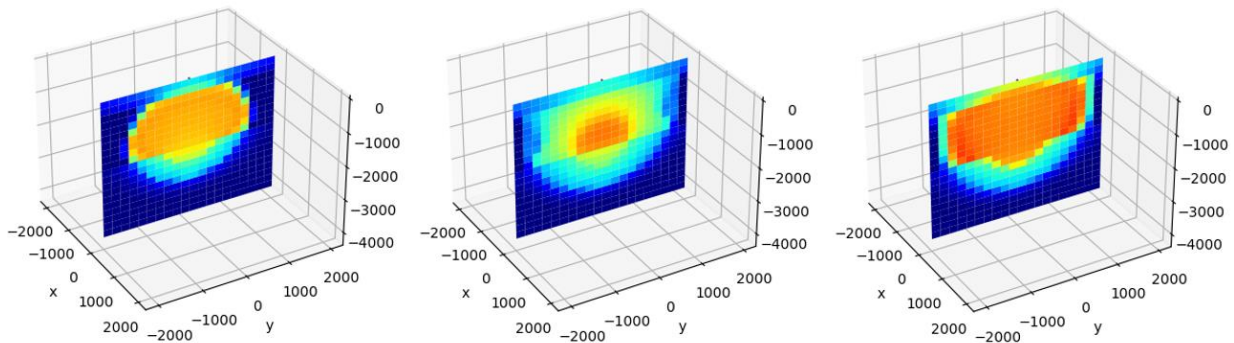
This will show *a-b* value as follows



(5) Run *RUN_QUAKEDFN.jl*

Simulation results

Plotstep: 200, 300, 400



Rupture propagation is blocked at shallow (<500m) and deep (>2000m) areas.

3.4 Two strike-slip faults with a pressure source

- (1) Run *InputGeometryExamples/Example4_BuildGeometry_Two_StrikeSlip.jl*
- (2) Run *RUN_BUILD_INPUT.jl*
- (3) Run *ExternalStressCalculation/PressureCalculation_Rudnicki.jl*

Make sure to Change Pressure origin(fault center of smaller fault)

```
FlowRate=100 # kg/s
PressureOrigin=[-1000.0, 0.0,-1500]; # Custom Faults
Permeability = 1e-16;
Viscosity = 0.4e-3;
SkemptonCoeff=0.75;
PoissonRatio_Undrained=0.3;
FluidDensity_Ref = 1e3;
```

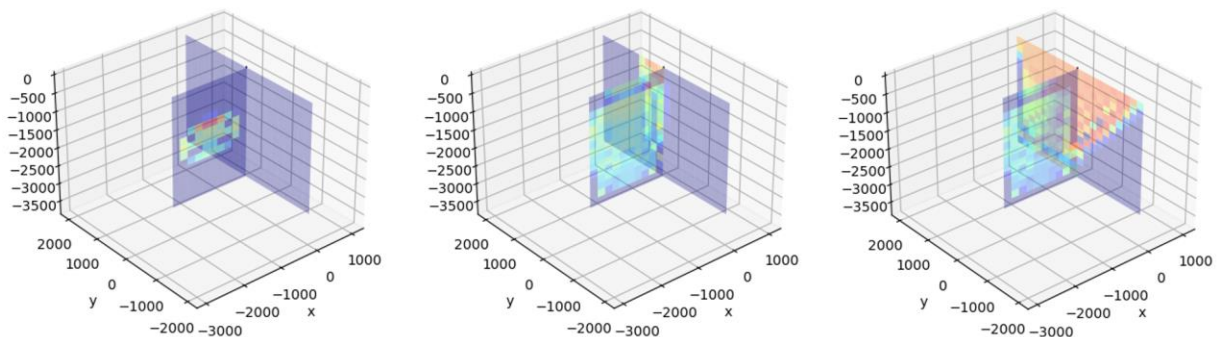
- (4) Run *RUN_QUAKEDFN.jl*

```
##### Simulation Time Set #####
TotalStep = 10000 # Total simulation step
SaveStep = 5000 # Automatically saved every this step
RecordStep = 10 # Simulation sampling rate

##### Time Stepping Setup #####
TimeStepOnlyBasedOnUnstablePatch = 1 # if 1, time step is calculated only based
on the unstable patch
TimeStepPreset = 3 # 1: conservative --> 4: optimistic
```

Simulation results

Plotstep: 100, 300, 400



Extra: Stress Based Initial Condition

One can adjust the initial condition based on the local stress field. Consider rate and state friction law:

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

To conduct the simulation, one should define any three of the following four: μ , μ_0 , V_i , θ_i .

Practically, μ_i should be calculated based on the local stress field. The μ_0 is a measurable parameter (together with V_0). The two initial conditions, V and θ , are relatively hard to be measured. However, we may estimate the initial θ_i since its maximum value is the elapsed time from the last rupture (maximum $d\theta/dt = 1$ s/s), while the initial V has no limit. Hence, one may want to fix the initial θ and determine V correspondingly.

There are two different ways to implement this. (i): before the discretization and (ii) after the discretization. The first method can be implemented as follows.

(1) Run *InputGeometryExamples/Example4_BuildGeometry_Two_StrikeSlip.jl*

(2) Run *InputGeometryExamples/AdjustParameterWithStress*

```
MaxStressOrientation = 135. # between 0-180 degree
StressRatioMaxOverMin = 0.5 # Sig3/Sig1
MaxSigSurface = 2e6 # Stress at surface
MaxSigGrad = 6000.0 # Stress gradient along max stress orientation [Pa/m]
Fault_Theta_i = 1e10 # if zero, automatically calculated based on the velocity
and Mu0
Fault_V_i = 0.0 # if zero, automatically calculated based on the theta and Mu0
Friction_0 = 0.30 # if zero, automatically calculated based on the velocity and
Theta
V0=1e-9
MinFrictionAllowed = 0.1
```

Once run, *Input_BulkFaultGeometry.txt* is changed according to the above input. μ is calculated by maximum stress orientation (135°) and $\sigma_3/\sigma_1=0.5$, $\mu_0=0.3$, $\theta = 10^{10}s$ and initial velocity is set accordingly.

(3) Run *RUN_BUILD_INPUT.jl*

(4) Run *ExternalStressCalculation/PressureCalculation_Rudnicki.jl*

```
FlowRate=100 # kg/s
PressureOrigin=[-1000.0, 0.0,-1500]; # Custom Faults
Permeability = 1e-16;
Viscosity = 0.4e-3;
SkemptonCoeff=0.75;
PoissonRatio_Undrained=0.3;
FluidDensity_Ref = 1e3;
```

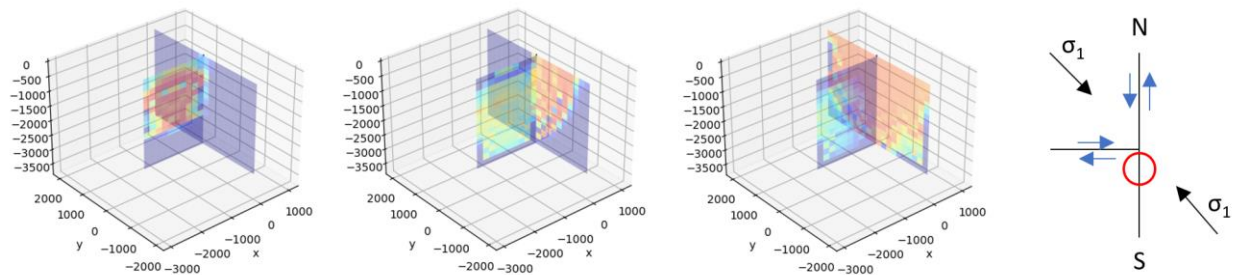
(5) Run *RUN_QUAKEDFN.jl*

```
##### Simulation Time Set #####
TotalStep = 10000 # Total simulation step
SaveStep = 5000 # Automatically saved every this step
RecordStep = 10 # Simulation sampling rate

##### Time Stepping Setup #####
TimeStepOnlyBasedOnUnstablePatch = 1 # if 1, time step is calculated only based
on the unstable patch
TimeStepPreset = 3 # 1: conservative --> 4: optimistic
```

Simulation results

Plotstep: 100, 150, 200



The simulation result is similar to the original simulation except that the mainfault rupture initially propagate toward opposite direction. This is due to the applied stress field. The smaller fault is right lateral now due to the maximum stress orientation is 135°. The smaller fault slip will cause normal strss reduction at south half of the main fault making the rupture propage toward south.

The stress based initial condition change can be also conducted by uncommenting following lines in *QuickParameterAdjust.jl*.

```
#####
##### Calculation of initial state from stress orientation #####
# MaxStressOrientation = 100. # between 0-180 degree
# StressRatioMaxOverMin = 0.5
# MinFrictionAllowed = 0.1 # smaller than this friction is not allowed

# StressGradAtMaxOrientation = 10000.0
# SurfaceStressAtMaxOrientation = 2e6

# Fault_Theta_i .= 1e10
# Fault_V_i .= 0.0
```

```

# Friction_0 = ones(FaultCount) * 0.32
# V0=1e-9;

# Fault_Friction_i, Fault_NormalStress, Fault_V_i, Fault_Theta_i =
#           StressDependentFrictionParameters(MaxStressOrientation,
StressRatioMaxOverMin, MinFrictionAllowed,
#           StressGradAtMaxOrientation, SurfaceStressAtMaxOrientation,
#           FaultStrikeAngle, FaultDipAngle, Fault_V_i, Fault_Theta_i,
Fault_Friction_i, FaultLLRR,
#           Fault_a, Fault_b, Fault_Dc, Fault_NormalStress, Friction_0,
FaultCenter)

#####^#####
#####
#####

```

*If parameters are adjusted here, one can run **RUN_QUAKEDFN.jl** right away.*

*It is convenient to change parameters here since it doesn't require discretization. However, the changes in **QuickParameterAdjust.jl** can not change slip orientation (left/right lateral).*

3.5 Two reverse faults with a pressure source

(1) Run *InputGeometryExamples/Example5_BuildGeometry_TwoFaults_Normal.jl*

```

Input_BulkFaultGeometry.txt
1 SwitchSS/RN ShearMod PoissonRatio R_Density Crit_TooClose TooCloseNormal_Multiplier Minimum_NS
2 2.0 1.0e10 0.2 2400.0 1.05 0.6 2.0e6
3 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
4 -577.3502691896258 -0.0 1000.0 2000.0 2309.401876758503 90.0 60.0 -1.0 0.003 0.006 0.0002 1.0e10 1.0e-15 0.33 2.0e6 4500.0 0.0 200.0
5 363.97023426620234 0.0 1000.0 2000.0 2128.355544951824 90.0 110.0 -1.0 0.003 0.006 0.0002 1.0e10 1.0e-15 0.33 2.0e6 4500.0 0.0 200.0
6 -363.97023426620234 0.0 3000.0 2000.0 2128.355544951824 90.0 110.0 -1.0 0.003 0.006 0.0002 1.0e10 1.0e-15 0.33 2.0e6 4500.0 0.0 200.0

```

In *Input_BulkFaultGeometry.txt*, SwitchSS/RN is now 2.0. If this is 1.0, the system is strike-slip, and if this is 2, the system is reverse-normal slip. Now, both fault has -1 at LR/RN. In the Reverse-Normal system, -1 denotes reverse faulting.

(2) Run *RUN_BUILD_INPUT.jl*

(3) Run *ExternalStressCalculation/PressureCalculation_Rudnicki.jl*

```

FlowRate=100 # kg/s
PressureOrigin=[-577.0, 0.0,-1000]; # Custom Faults
Permeability = 1e-16;
Viscosity = 0.4e-3;
SkemptonCoeff=0.75;
PoissonRatio_Undrained=0.3;
FluidDensity_Ref = 1e3;

```

(4) Run *RUN_QUAKEDFN.jl*

```

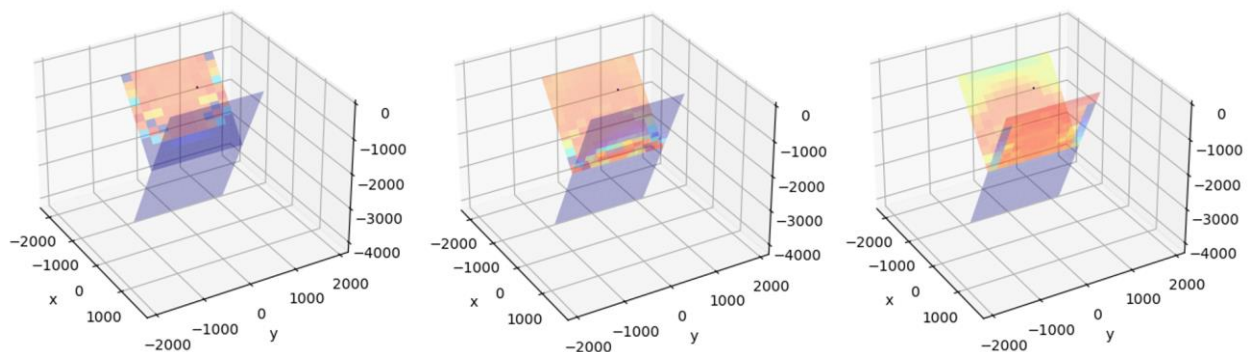
##### Simulation Time Set #####
TotalStep = 10000 # Total simulation step
SaveStep = 5000 # Automatically saved every this step
RecordStep = 10 # Simulation sampling rate

##### Time Stepping Setup #####
TimeStepOnlyBasedOnUnstablePatch = 1 # if 1, time step is calculated only based
on the unstable patch
TimeStepPreset = 3 # 1: conservative --> 4: optimistic

```

Simulation results

Plotstep: 150, 170, 190



References

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