PyQuake3D

3D BOUNDARY ELEMENT METHODS FOR
EARTHQUAKE MODELING

PyQuake3D Documentation

Rongjiang TANG[1] & **Luca DAL ZILIO**[2]

February 17, 2025

[1]rongjiang.igp@hotmail.com
[2]luca.dalzilio@ntu.edu.sg

Contents

1 Introduction

PyQuake3D is a **Python**-based Boundary Element Method (BEM) code for simulating sequences of seismic and aseismic slip (SEAS) on a complex 3D fault geometry governed by rate- and state-dependent friction. It can handle arbitrary 3D fault geometries in either uniform elastic half-space or full-space scenarios, including simulations of non-planar, branches, step-over, and rough faults. This document provides an overview of how to use the script, as well as a detailed description of the input parameters.

2 Contribution

2024.7.1 Rongjiang Tang developed the code framework and the Quasi-dynamic BIEM Seismic cycle model on a complex 3D fault geometry governed by regularized aging law.

2024.10.5 Rongjiang Tang and Luca Dal Zilio implemented the H-matrix Matrix-Vector Multiplication and developed the Cascadia model.
We welcome contributions to PyQuake3D. Please ensure that you follow the contribution guidelines and maintain the consistency of the codebase.

3 Dependencies

$$\text{Python} \geq 3.8$$
$$\text{NumPy} \geq 1.2$$
$$\text{ctypes} == 1.1$$

Generally, the numpy library comes pre-installed with Python. The ctypes is a library used for calling C code from Python, and it can be installed using:
pip install ctypes

4 How to run

Download the source codes from github. Type:
To run the PyQuake3D script, use the following command

```
python -g --inputgeo <input_geometry_file> -p --inputpara <input_parameter_file>
```

For example,To execute benchmarks like BP5-QD at Current Directory

```
python src/main.py -g examples/bp5t/bp5t.msh -p examples/bp5t/parameter.txt
```

To run cascadia model, use:

```
python src/main.py -g examples/cascadia/cascadia35km_ele4.msh -p
examples/cascadia/parameter.txt
```

Ensure that the 'parameter.txt' file in each example is appropriately configured as described in the parameter setting description.

5 Code Structure and file description

All the python codes are located in the *src* directory, and parameters and models can be found in the examples folder. H2lib is a C language H-matrix library used to compile the dynamic library hm.so, which allows Python to call the H2lib C programs to use H-Matrix. Generally, you do not need to recompile the library, as we have already compiled the dynamic library in a Linux environment and placed it in the src directory. However, if you are using macOS or need to modify the C code, you will need to recompile and link the code in the H2lib-master directory using make, and then replace the original hm.so file in the *src* directory with the newly compiled version.

The examples directory contains three subdirectories, each with an example. The bp5t directory contains the benchmark half-space fault model published by the Southern California Earthquake Center (https://strike.scec.org/cvws/seas/download/). The Cascadia directory simulates the Cascadia subduction zone, based on the publicly available slab2 model (https://www.sciencebase.gov/catalog/item/5aa1b00ee4b0b1c392e86467). Surface0 is a simple half-space non-planar fault surface model used for testing simulation results for non-planar condition.

The code consists of multiple modules as the following functions:

- **main.py**: Executing the quasi-dynamic model program, allow the user to code for output
- **QDsim.py**: Class for establishing a quasi-dynamic model
- **DH_Greenfunction.py**: Calculate stress green functions in homogeneous Elastic Half-Space
- **SH_Greenfunction.py**: Calculate stress green functions in homogeneous Elastic Full-Space.
- **Readmsh.py**: Reading External Data and parameters
- **Hmatrix.py**: Load dynamic library

6 Parameters setting

The simulation parameters are implemented by modifying the *parameter.txt* file, rather than by changing the source code. The input variable list is in $Table$ 1. If $InputHetoparamter$ in $Table$ 1 is True, heterogeneous stress and friction parameters are imported from external files. The external filename is defined in *parameter.txt* and must remain in the same directory as *parameter.txt*. In this case, you only need to appropriately set parameter of $Table$ 1 and $Table$ 4. Otherwise, you need to appropriately set the parameters of stress and frictional initial condition shown in $Table$ 2 and $Table$ 3.

7 External file format

In addition to *parameter.txt*, the program requires external files, primarily the .msh and $Inputparameterfiles$. The filename for $Inputparameterfiles$ is defined within *parameter.txt*.

Table 1: General Parameters

Parameter	**Default**	**Description**
`Corefunc directory`		The storage path for the kernel function matrix composed of stress Green's functions
`Input Hetoparamter`	False	If 'True', the heterogeneous stress and friction parameters are imported from external files.
`Inputparamter file`		The file name of imported heterogeneous stress and friction parameters
`Processors`	50	The number of processors which can be scheduled by the operating system to run on different CPU cores.
`Batch_size`	1000	The number of processes which can be scheduled by the operating system to run on different CPU cores.
`H-matrix`	False	If 'True', The kernel function will be approximated using H-Matrix. Note that in this case, the code must be run under Linux or MacOS system. The H-Matrix approximation is implemented using the open-source C library *H2Lib*, and the compilation of the dynamic library is done based on Makefile in $H2Lib-master$ directory.
`Lame constants`	0.32e11 Pa	The first Lame constant
`Shear modulus`	0.32e11 Pa	Shear modulus
`Rock density`	2670 kg/m^3	Rock mass density
`Reference slip rate`	1e-6	Reference slip rate.
`Reference friction coefficient`	0.6	Reference friction coefficient.
`Plate loading rate`	1e-6	Plate loading rate.

The *.msh* file contains necessary mesh data such as node coordinates, element numbers, and other relevant information. It is exported from *Gmsh* software. Please ensure to select the $Version2ASCII$ format to match the code's reading program.

When $InputHetoparameter == True$, the initial condition can be imported externally (via the Inputparameter file):

```
rake(0) a(0) b(0) dc(0) f0(0) tau(0) sigma(0) vel(0) taudot(0) sigdot(0)
rake(1) a(1) b(1) dc(1) f0(1) tau(1) sigma(1) vel(1) taudot(1) sigdot(1)
...
```

The total number of rows in the *InputHetoparameter* file is equal to the number of cells. The first column represents the rake angle, which is currently kept constant in the calculations. Columns 2, 3, 4, and 5 represent parameters related to the rate-state friction law. Columns 6 and 7 denote shear and normal tractions, respectively. Column 8 is the initial slip rate, and the last two columns represent the loading rates for shear and normal tractions. Note that this loading refers to additional loading other than the slip deficit rate loading, with a default value of 0.

8 Initial Condition

The initial values for frictions, slip and stress can be imported from an external file (section 7), or directly set (section 6) under simple fault geometry conditions. However,when the initial shear stress information is missing, the scalar pre-stress τ_0 is chosen as the steady-state stress:

$$\tau_0 = \overline{\sigma}_n \cdot a \cdot \sinh^{-1}\left(\frac{V_{\text{init}}}{2V_0} \cdot \exp\left(\frac{f_0 + b\ln\left(\frac{V_0}{V_{\text{init}}}\right)}{a}\right)\right) \tag{1}$$

9 Stop Control

The simulation stops when any of the following is satisfied.
1.The time-step reaches the *totaloutputsteps*.
2.RungeKutta_solve_Dormand_Prince iteration attempted 20 times without meeting the error tolerance requirements.

10 Visualization

The current program supports output in VTK format, which can be used for 3D visulization in ParaView. We encourage users to develop their own code within the main function to flexibly output data uing sim0 object from QDsim class. The sim0 object contains all variables required for the simulation, and you can access them by using sim0.variables. Code framwork is shown as lstlisting 1.

```
# Create class

sim0=QDsim.QDsim()
nodelst,elelst=readmsh.read_mshV2(fnamegeo)
sim0=QDsim.QDsim(elelst,nodelst,fnamePara)

# Forward modelling of earthquake cycle
for i in range(totaloutputsteps):
        if(i==0):
                dttry=sim0.htry
        else:
                dttry=dtnext
        dttry,dtnext=sim0.simu_forward(dttry)

# print sim0.variables
print(sim0.__dict__)

...user write code to ouput sim0.variables...
```

Listing 1: Forward implementation and user-defined ouput code

11 Common issues and potential solutions

If computer memory is insufficient, parallel computing of the Green's function may lead to errors. In such cases, it may be necessary to run $self.A1s, self.A2s, self.Bs = self.get_coreAB_mulproess1()$ and $self.A1d, self.A2d, self.Bd = self.get_coreAB_mulproess2()$ separately to prevent memory overflow. We are currently optimizing this part of the code. If the calculation diverges, such as in cases where the slip rate becomes excessively large or the Runge-Kutta iteration error is too high, reducing the grid size may solve this problem.

12 Testing and Validation

PyQuake3D is the first Python-based three-dimensional quasi-dynamic earthquake cycle numerical simulation program in the geophysical open-source community. It serves as a pivotal tool for advancing Spontaneous Earthquake System (SEAS) models, contributing to a deeper understanding of earthquake system dynamics. Python's simple and readable syntax makes PyQuake3D accessible to both beginners and experienced developers, facilitating code development, comprehension, and maintenance. As researchers strive to develop more advanced and detailed SEAS models, it is crucial to validate PyQuake3D against other numerical codes to ensure reliable and reproducible results, thereby supporting sustained scientific progress.

Jiang et al. (2021) provide comprehensive benchmark descriptions, including quasi-dynamic formulations for earthquake ruptures. These benchmark results are open-source and incorporated into the SCEC/USGS Spontaneous Rupture Code Verification Project. In this study, PyQuake3D is evaluated using the BP5-QD benchmark for 3D SEAS simulations, and its results are compared with other numerical codes (Figure 2). The BP5-QD benchmark features a strike-slip vertical fault in a three-dimensional elastic half-space. Full descriptions of the benchmark are available on the SEAS code comparison platform (https://strike.scec.org/cvws/seas/) and in Jiang et al. (2021).

The initial conditions of the fault need to be carefully set. The initial slip rate is set to $V_{\text{init}} = 10^{-9}\,\text{m/s}$, and the effective normal stress is $\bar{\sigma}_n = 25\,\text{MPa}$ the scalar pre-stress τ^0 is chosen as the steady-state stress

$$\tau^0 = \bar{\sigma}_n \sinh^{-1}\left[\frac{V_{\text{init}}}{2V_0}\exp\left(\frac{f_0 + b\ln\left(\frac{V_0}{V_{\text{init}}}\right)}{a}\right)\right]$$

Where reference slip rate $V_0 = 10^{-9}$. To break the symmetry of the problem and enable comparisons across different simulations, a higher pre-stress τ^i is applied within the square nucleation region, which has a width of 12 km:

$$\tau^i = \bar{\sigma}_n\,\text{asinh}^{-1}\left[\frac{V_i}{2V_0}\exp\left(\frac{f_0 + b\ln\left(\frac{V_0}{V_{\text{init}}}\right)}{a}\right)\right] + \frac{\mu}{2c_s}V_i$$

This initial condition should lead to an immediate initiation of the first seismic event.

It is noteworthy that the computation of the BPQ5 model, which consists of 9,200 elements and spans 21,500 time steps, took approximately 49 minutes in total, with an average computation time of about 8.5 seconds per 100 time steps. PyQuake3D can be run on a laptop, although computational efficiency may be somewhat lower compared to

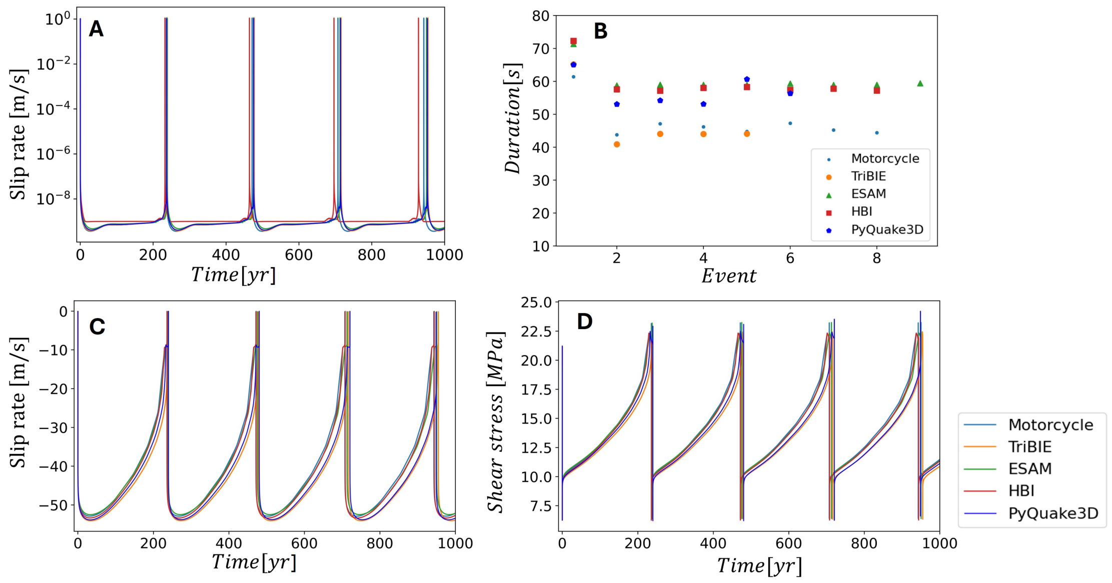

Figure 1: Coseismic rupture of the first event in BP5-QD simulations. The time evolution of (a, c, and e) slip rates and (b, d, and f) shear stresses during the first earthquake is shown at different locations on the fault. Panels (a) and (b) correspond to a point within the initial nucleation zone (x = 24 km; x = 10 km). Panels (c) and (d) correspond to a point at the free surface (x = 0 km; x = 0 km). Panels (e) and (f) correspond to a point within the propagation zone (x = 0 km; x = 10 km).

high-performance systems. However, its portability and reduced research costs make it a convenient option for many users.

For the simulations, we selected a total simulated time long enough to generate up to six large earthquakes. Model behavior was assessed through various types of simulation outputs from different codes, with the grid resolution for all models set to 1000 m. Figure 4 presents a comparison of the time series for global source properties, which include the evolving maximum slip rates and moment rates across the entire seismogenic fault area. These properties are crucial for accurately determining the initiation and cessation times of earthquakes. The results, including those from PyQuake3D, show good agreement. In terms of rupture duration, there was a discrepancy of less than 20 seconds across the different codes, with the rupture duration computed by PyQuake3D falling within the middle range, further supporting the accuracy of the model.

Figures 5 and 6 present the coseismic time series at two local points on the fault. We compare the coseismic slip rate and traction evolution for the first and fourth earthquakes. Although discrepancies are observed among the results from different codes, with these discrepancies generally amplifying over time, the results from PyQuake3D are found to be in good agreement with those from the majority of the other codes.

We compare the coseismic rupture fronts during the first event in simulations with a grid spacing of 1000 m (Figure 7). The rupture time data record the moments when each point on the fault reaches a specific threshold slip rate ($V_{\text{th}} = 0.03\, m/s$) during the first earthquake. Although the rupture front contours exhibit some discrepancies, they maintain qualitative agreement. We observe that the BEM-based PyQuake3D produces rupture speeds that are close to the average values within the group, particularly in comparison to the SBEM/BEM simulations. The largest discrepancies are found with the

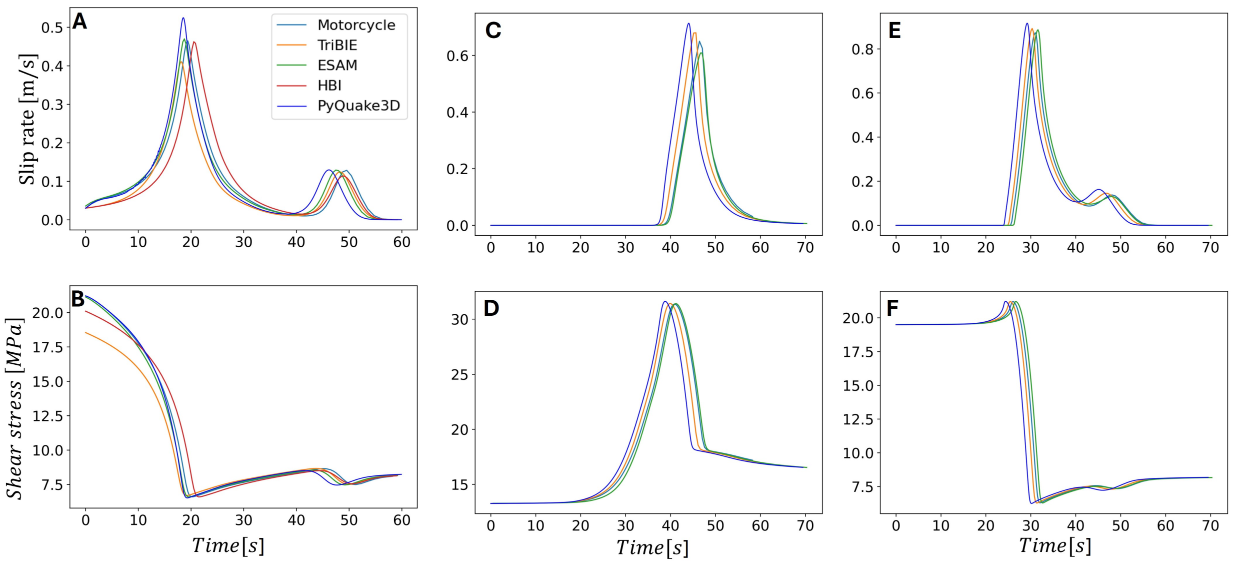

Figure 2: Long-term fault behavior and earthquake characteristics in BP5-QD simulations. (a) Time evolution of maximum slip rates in the seismogenic zone and (b) rupture duration

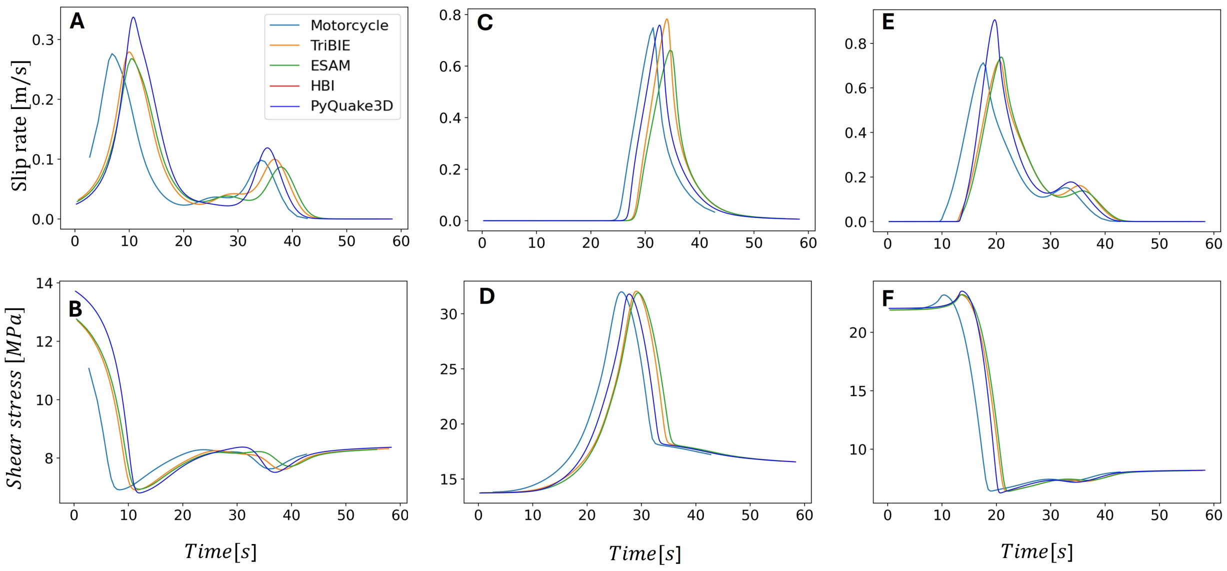

Figure 3: Coseismic rupture of the fourth event in BP5-QD simulations. Time evolution of (a, c, and e) coseismic slip rates and (b, d and f) are the same with Figure 5 but for the fourth event.

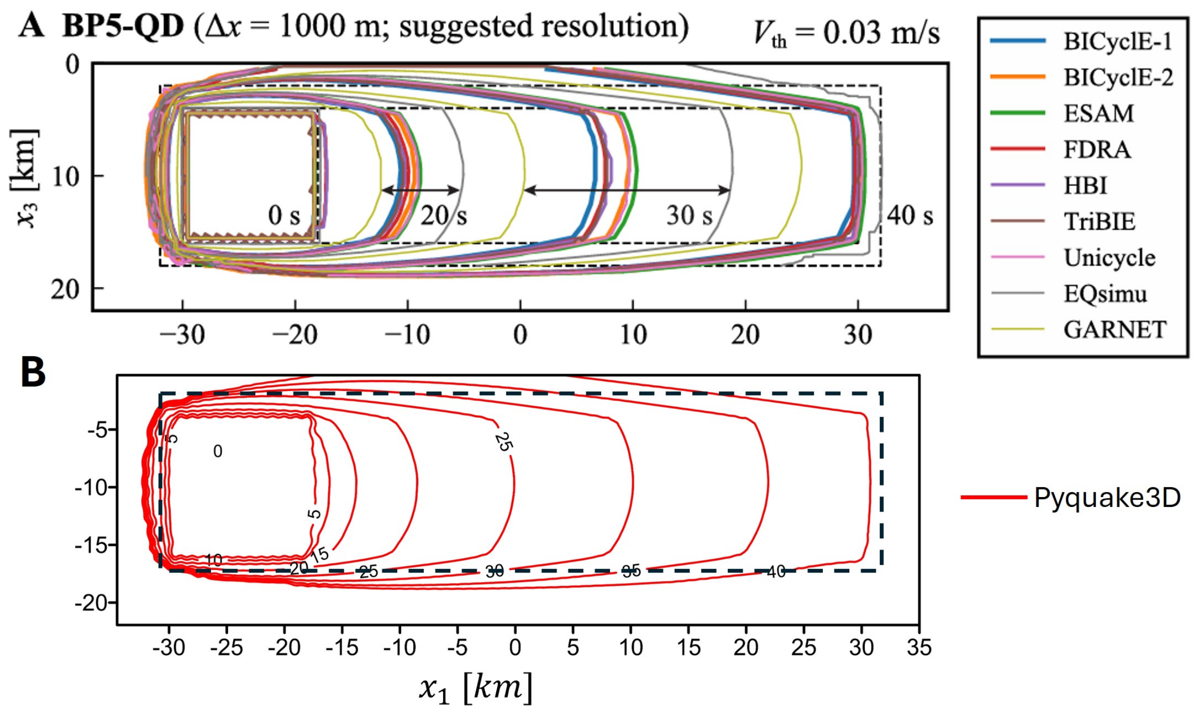

Figure 4: Rupture fronts of the first earthquake in BP5-QD simulations with different codes. The contours of rupture fronts indicate 0, 20, 30, and 40 s after the earthquake initiation time in simulations with (a) Other codes and (b) PyQuake3D. The threshold slip rate for the coseismic phase, ($V_{\mathrm{th}} = 0.03\, m/s$).

two volume-discretized codes.

Next, we present a comparison of the overall earthquake patterns in the BP5-QD simulations (Figure 8). We compare the fault slip evolution profiles along the strike direction obtained from three codes: FDRA, BICyclE, and PyQuake3D. Both FDRA and PyQuake3D utilize the Boundary Element Method (BEM), while BICyclE employs the Semi-Analytical Boundary Element Method (SBEM). The results indicate that, after the first earthquake, subsequent events exhibit similar recurrent slip patterns. Coseismic slip consistently initiates near the predefined nucleation zone and propagates into the shallow velocity-strengthening (VS) region. At the same time, postseismic and interseismic slip occur in the adjacent VS regions at both ends of the fault, displaying symmetry on either side.

It is noteworthy that PyQuake3D shows a slight deviation in the nucleation location for certain events, such as the third earthquake, where the nucleation point is positioned closer to the middle region of the velocity-weakening (VW) zone. This discrepancy has been observed by Jiang et al. (2021). The nucleation phase of later earthquakes is particularly sensitive to numerical resolution and domain-size-dependent loading. Moreover, we find that the nucleation phase of subsequent earthquakes is highly sensitive to different adaptive time-step strategies, as illustrated in Figure 7d.

13 License

This project is licensed under the MIT License.

14 Acknowledgments

We referred to the HBI code to develop original python-based BIEM algorithm:
`Ozawa, S., Ida, A., Hoshino, T., Ando, R. (2023). Large-scale earthquake sequence simulations on 3-D non-planar faults using the boundary element method accelerated by lattice H-matrices. Geophysical Journal International, 232(3), 1471-1481.`

We referred to MATLAB code to develop the kernel function:
`Nikkhoo, M., Walter, T. R. (2015). Triangular dislocation: an analytical, artefact-free solution. Geophysical Journal International, 201(2), 1119-1141.`

The implementation of the hierarchical matrix is mainly based on open-source code H2lib. http://www.h2lib.org/.

We would like to thank Associate Prof. Ryosuke Ando and Dr.So Ozawa for their help in the code development, and Professor Steffen Börm for his assistance with HMatrix programming.

For further details, please refer to the official documentation or contact the development team.
We referred to MATLAB code from Nikkhoo and Walter (2015) to develop the kernel function:
`Nikkhoo, M., Walter, T. R. (2015). Triangular dislocation: an analytical, artefact-free solution. Geophysical Journal International, 201(2), 1119-1141.`

The implementation of the hierarchical matrix is mainly based on open-source code *H2lib*, which is developed by Professor Börm. `http://www.h2lib.org/`. We would like to thank Professor Steffen Börm for his assistance with HMatrix programming.

15 Contact

For further details, please refer to the official documentation or contact the development team.
Rongjiang Tang: rongjiang.igp@hotmail.com
Luca Dal Zilio: luca.dalzilio@ntu.edu.sg

Table 2: Stress and Frition Settings

Parameter	Default	Description
`Vertical principal stress (ssv)`	1.0	The vertical principal stress scale: the real vertical principal stress is obtained by multiplying the scale and the value
`Maximum horizontal principal stress (ssh1)`	1.6	Maximum horizontal principal stress scale.
`Minimum horizontal principal stress(ssh2)`	0.6	Minimum horizontal principal stress scale
`Angle between ssh1 and X-axis`	30°	Angle between maximum horizontal principal stress and X-axis.
`Vertical principal stress value`	4e7 Pa	Vertical principal stress value
`Vertical principal stress value varies with depth`	True	If True, Vertical principal stress value varies with depth
`Vertical principal stress value varies with depth`	15000 m	If vertical principal stress value, it maintains a constant value at the conversion depth, and the horizontal principal stress value also changes with depth simultaneously
`Shear traction solved from stress tensor`	False	If ‘True‘, the non-uniform shear stress is projected onto the curved fault surface by the stress tensor
`Rake solved from stress tensor`	True	If ‘True‘, the non-uniform rakes are solved from the stress tensor.The angle of rake is defined as 90 =reverse faulting,0=right lateral strike slip faulting, and -90=normal faulting.
`Fix_rake`	30°	If ‘True‘, Set fixed rakes if ‘Rake solved from stress tensor‘ is ‘False‘.
`Shear traction in VW region`	0.53	The ratio of shear traction to normal traction in the velocity strengthening region.
`Shear traction in VS region`	0.78	The ratio of shear traction to normal traction in the velocity weakening region.
`Shear traction in nucleartion region`	1.0	The ratio of shear traction to normal traction in the velocity nucleation region.
`Widths of VS region`	10000.0 m	The width of the velocity weakening region.
`Transition region ratio from VS to VW region`	0.4	The size ratio of transition to VS region

Table 3: Nucleation and Friction Setting

Parameter	Default	Description
Set_nucleation	True	If True, sets a patch whose shear stress and sliding rate are significantly greater than the surrounding area to meet the nucleation requirements.
Radius of nucleation	8000 m	The radius of the nucleation region
Nuclea_posx	34000 m	Posx of Nucleation
Nuclea_posy	15000 m	Posy of Nucleation
Nuclea_posz	-15000 m	Posz of Nucleation
Rate-and-state parameters a in VS region	0.04	Rate-and-state parameters a in VS region
Rate-and-state parameters b in VS region	0.03	Rate-and-state parameters a in VS region
Characteristic slip distance in VS region	0.13 m	Characteristic slip distance in VS region
Rate-and-state parameters a in VW region	0.004	Rate-and-state parameters a in VW region
Rate-and-state parameters a in VW region	0.03	Rate-and-state parameters a in VW region
Characteristic slip distance in VW region	0.13 m	Characteristic slip distance in VW regioN
Rate-and-state parameters a in nucleation region	0.004	Rate-and-state parameters a in nucleation region
Rate-and-state parameters a in nucleation region	0.03	Rate-and-state parameters a in nucleation region
Characteristic slip distance in nucleation region	0.14 m	Characteristic slip distance in nucleation regioN
Initial slip rate in nucleation region	3e-2	Initial slip rate in nucleation region

Table 4: Output Setting

Parameter	**Default**	**Description**
`totaloutputsteps`	2000	The number of calculating time steps.
`outsteps`	10	The time step interval for outputting the VTK files.
`outputstv`	True	If True, the VTK files will be saved in out directroy.
`outputmatrix`	False	If True, the matrix format txt files will be saved in out directroy.