

HBI User's guide

So Ozawa

June 17, 2024

1 Introduction

Fortran90 software HBI performs three-dimensional quasi-dynamic numerical simulations of sequences of earthquakes and aseismic slip (SEAS). It can handle arbitrary fault geometry such as branches, step-overs, and rough faults in an elastic half-space. 2D problems can also be solved in the same code. The computation is accelerated by taking advantage of H-matrices. HACApK, open-source software for H-matrices, is called for constructing and operating H-matrices.

The latest versions can be found at <https://github.com/sozawa94/hbi>. Feel free to contact So Ozawa (sozawa@stanford.edu), if you have any questions.

2 License

This software is freely available under the MIT license. If you write a paper using this code, please cite the following manuscript.

So Ozawa, Akihiro Ida, Tetsuya Hoshino, Ryosuke Ando (2023), "Large-scale earthquake sequence simulations of 3D geometrically complex faults using the boundary element method accelerated by lattice H-matrices on distributed memory computer systems", Geophysical Journal International

3 How to run

Download the source codes from github. Type:

```
git clone http://github.com/sozawa94/hbi
```

To compile and link the code, type

```
make
```

You may need to modify `Makefile` depending on your environment.

To run a simulation, type:

```
./lhbiem (inputfile)
```

or use MPI parallelization;

```
mpirun -np 16 ./lhbiem (inputfile)
```

Some examples of input files are included in the directory. To run the code in parallel, **the number of MPI processes must be a squared number**. For 2D case with $N_{cellg} < 5,000$, serial simulations would be fast enough and is recommended because the analysis of output files is simpler.

As an example, `examples/2dp.in` and the corresponding geometry and parameter files are included. Several input files for SEAS project (<https://strike.scec.org/cvws/seas/>) are also available.

3.1 Note on Sherlock users

If you use HBI in Sherlock (cluster in Stanford University), you need to load required modules before make:

```
ml load ifort
ml load impi
```

You also need to switch compiler options by editing `Makefile`.

3.2 Monitoring ongoing jobs

The standard output first shows the information about (lattice) H-matrices (approximated integration kernel). Then, timestep and time (measured by year) are recorded.

`job.log` is a record of the start and normal stop of jobs.

4 Problem setting

4.1 Description of Input file

Various simulations can be performed by modifying the input file, rather than modifying the source code. The input variable list is:

| name | type | meaning |
|-------------------|-----------|--|
| filenumber | integer | job number (name of output files) |
| problem | character | dimension, planar or non-planar, mesh, etc. |
| ncellg | integer | number of elements (not used in 3dp, 3dph, 3dht,3dnt) |
| nstep | integer | maximum number of time steps |
| interval | integer | output & checkpointing every this time steps |
| dtout | real | output every this time (year) |
| dtinit | real | initial time step width (s) |
| geometry_file | character | name of geometry file |
| parameter_file | character | name of parameter file |
| parameterfromfile | logical | true if parameters are read from parameter_file |
| restart | logical | true if restarting a suspended job |
| backslip | logical | true if loading is added by backflip |
| vpl | real | the velocity of backslip (m/s) (used if backflip=T) |
| sr | real | the stressing rate (MPa/s) (used if backslip=F) |
| tmax | real | maximum time (year) |
| sigmaconst | logical | true if normal stress is kept constant |
| ds | real | element size (km) (used in 2dp, 3dp, 3dph) |
| imax | integer | the number of elements in slip-parallel direction (used in 3dp, 3dph) |
| jmax | integer | the number of elements in slip-perpendicular direction (used in 3dp, 3dph) |
| crake | real | rake angle (deg) in 3D problems (not used if parameterfromfile=T) |
| dipangle | real | dip angle (deg) used in 3dph and 2dph |
| evlaw | character | evolution law for RSF ("aging" or "slip", default="aging") |
| a | real | a in RSF (not used if parameterfromfile=T) |
| b | real | b in RSF (not used if parameterfromfile=T) |
| dc | real | dc in RSF (not used if parameterfromfile=T) |
| f0 | real | f0 in RSF (not used if parameterfromfile=T) |
| tauinit | real | initial shear stress (MPa) (not used if parameterfromfile=T) |
| sigmainit | real | initial normal stress (MPa) (not used if parameterfromfile=T) |
| velinit | real | initial slip rate (m/s) (not used if parameterfromfile=T) |
| eps_h | real | error allowance in H matrix (default: 1e-4) |
| eps_r | real | error allowance in Runge-Kutta (default: 1e-4) |
| velmax | real | computation stop if maximum velocity is above this value (default:1e7) |
| velmin | real | computation stop if maximum velocity is below this value (default:1e-16) |
| limitsigma | logical | true if you want to limit normal stress |
| maxsig | real | maximum normal stress (MPa) (used if limitsigma=T) |
| minsig | real | minimum normal stress (MPa)(used if limitsigma=T) |

4.2 Problem

The list of problems are follows:

| problem | description |
|---------|--|
| 2dp | 2D planar fault in full-space (plane strain) |
| 2dph | 2D planar fault in half-space (plane strain) |
| 2dn | 2D nonplanar fault in full-space (plane strain) |
| 2dnh | 2D nonplanar fault in half-space (plane strain) |
| 25d | 2D nonplanar fault in full-space (plane strain) with finite fault width (2.5D approximation) |
| 2dvs | 2D vertical strike-slip fault |
| 3dp | 3D planar squared elements in full-space |
| 3dph | 3D planar squared elements in half-space |
| 3dnr | 3D nonplanar squared elements in full-space |
| 3dhr | 3D nonplanar squared elements in half-space |
| 3dnt | 3D nonplanar unstructured triangular elements in full-space |
| 3dht | 3D nonplanar unstructured triangular elements in half-space |

4.3 Physical variables and constants

The following units are assumed:

| quantity | unit |
|---------------|------|
| time | s |
| location | km |
| displacement | m |
| slip velocity | m/s |
| wave speed | km/s |
| shear modulus | GPa |
| stress | MPa |

The fundamental physical constants are set in `m_const.f90`. The default values are

| quantity | value |
|----------------------------|-------------|
| rigidity | 32.04GPa |
| P-wave speed | 6.000 km/s |
| S-wave speed | 3.464 km/s |
| reference slip rate on RSF | 1 μ m/s |

4.4 Fault Geometry

For 2dp, 2dph, 3dp, and 3dph, fault geometry is set in inputfile. For other problems, an additional `geometry_file` is necessary.

4.4.1 2dp, 2dvs

The fault geometry is uniquely characterized by its element size `ds` and the number of elements `ncellg`.

4.4.2 3dp

The fault geometry is uniquely characterized by its element size `ds` and the number of elements along strike and dip (`imax`, `jmax`).

4.4.3 2dph

The fault geometry is characterized by its element size `ds`, the number of elements `ncellg`, and dip angle in degree `dipangle`. For nonplanar case, use `2dnh`.

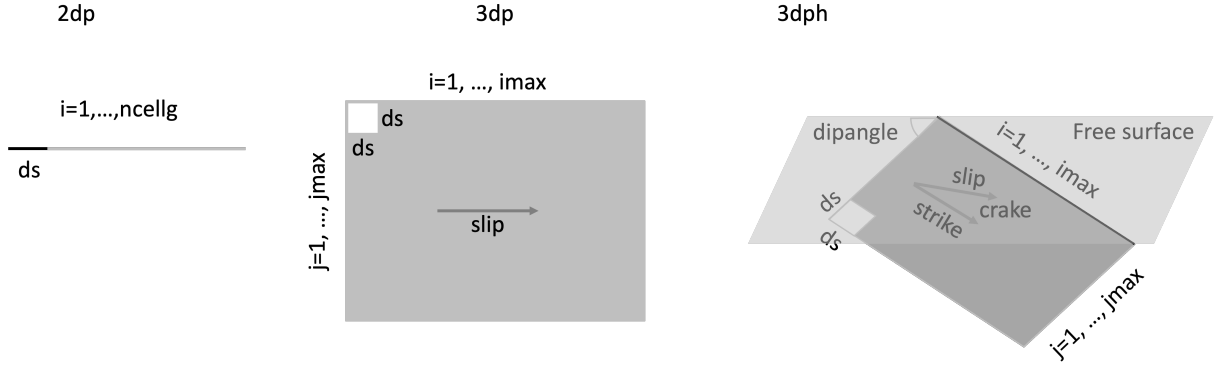


Figure 1: Fault geometry

4.4.4 3dph

The fault geometry is characterized by its element size **ds**, the number of elements along strike and dip (**imax**, **jmax**), and dip angle in degree **dipangle**. See also Figure 1.

4.4.5 2dn, 2dnh, 25d

The coordinates of two edges of a straight line have the geometrical information of an element. **xel**<**xer** is necessary. **geomerty_file** has the following format:

```
xel(1) xer(1) yel(1) yer(1)
xel(2) xer(2) yel(2) yer(2)
...
```

For 2dnh, the structure **geomerty_file** is the same with 2dn. Here, $y = 0$ is assumed to be the free surface and y -axis is downward (i.e., y is always positive).

4.4.6 3dnt, 3dht

The coordinates of the three edges of a triangle have the geometric information of an element. **geometry_file** should have STL (ASCII) format:

```
facet normal normal_vector(1:3)
  outer loop
    vertex xs1 ys1 zs1
    vertex xs2 ys2 zs2
    vertex xs3 ys3 zs3
  endloop
endfacet
```

This structure repeats **ncellg** times. For 3dht, $z = 0$ is the free surface and all elements must be located below the free surface (i.e., $z < 0$).

4.4.7 3dnr, 3dhr

Cell size **ds** must be specified in the inputfile. **geometry_file** should have STL (ASCII) format:

```
x(1) y(1) z(1) strike(1) dip(1)
x(2) y(2) z(2) strike(2) dip(2)
...
```

Here, x,y,z are the center of the square, strike and dip are measured in radian.

4.5 Parameters and Initial conditions

The following parameters and initial values are needed for each element.

4.5.1 Rake angle

In 3D problems, it is also necessary to specify the direction of slip (rake angle) for each element. The angle of rake is defined as 90 = reverse faulting, 0 = right lateral strike-slip faulting, and -90=normal faulting. The code does not allow for a time-varying slip direction. In the uniform case, set **crake** in **inputfile**. 2D problems do not use rake values.

4.5.2 Friction parameters

HBI uses the regularized version of the rate and state friction law (RSF), which is employed in the SEAS benchmark problems (Erickson et al., 2020). The friction coefficient are given by

$$\tau/\sigma = a \sinh^{-1}(V e^{\psi/a}/2V_0). \quad (1)$$

The state evolution is governed either by the aging law

$$\frac{d\psi}{dt} = \frac{bV_0}{d_c} \exp[(f_0 - \psi)/b] - \frac{bV}{d_c}. \quad (2)$$

or the slip law

$$\frac{d\psi}{dt} = -\frac{V_0}{d_c}(f - f_{ss}) \quad (3)$$

$$f_{ss} = f_0 + (a - b) \log \frac{V}{V_0} \quad (4)$$

The evolution law is the aging law as default. Set **evlaw** to **slip** in the inputfile to switch. The values of parameters (a, b, d_c, f_0) are set in **inputfile** in the case of uniform parameters.

4.5.3 Initial conditions

The initial condition is uniquely determined by the value of normal stress, shear stress, and slip rate at each element (the state variable is determined by these three accordingly). In the case of uniform initial values of (σ, τ, V) , these values are set in **inputfile** as **sigmainit**, **tauinit**, and **velinit**.

4.5.4 External loading

To have repeated ruptures on the same fault, external loading is necessary. **taudot** and **sigdot** are the stressing rates for shear and normal stresses, respectively. In the uniform case, **taudot** is **sr** and **sigdot** is zero. the value of **sr** must be specified in **inputfile**.

Alternatively, the code has "backslip" loading option. If **backslip=T**, then the loading is added by slip deficit rate, that is,

$$\dot{\tau}_i = -V_{pl} \sum_j K_{ij} \quad (5)$$

$$\dot{\sigma}_i = -V_{pl} \sum_j L_{ij} \quad (6)$$

where K_{ij} and L_{ij} are kernel matrices for shear and normal stresses, respectively. The value of **vp1** must be specified in **inputfile**. Spatially non-uniform backslip has not been implemented.

4.5.5 Format of parameterfile

If you want to use non-uniform parameters and initial values, set `parameterfromfile T` in `inputfile` and create `parameter_file`. `parameter_file` has following structure:

```
rake(1) a(1) b(1) dc(1) f0(1) tau(1) sigma(1) vel(1) taudot(1) sigdot(1)
rake(2) a(2) b(2) dc(2) f0(2) tau(2) sigma(2) vel(2) taudot(2) sigdot(2)
...
```

4.6 Stop control

The simulation stops when any of the following is satisfied.

1. the time-step reaches `nstep`
2. the maximum slip rate is greater than `velmax`.
3. the maximum slip rate is smaller than `velmin`.
4. the physical time reaches `tmax`.

4.7 Useful functions

If `sigmaconst=T`, then normal stresses do not change over time as assumed in many studies. If `sigmaconst=F` (as the default value), the normal stress changes with time. The computation can fail if the normal stress becomes very large or small. To avoid this, the user can set limits on the value of normal stress by setting `limitsigma=T`. In this case, `maxsig` and `minsig` are the maximum and minimum normal stresses, which should be stated in `inputfile`.

5 Visualization

For those familiar with Python, `Analysis2D.ipynb` and `Analysis3D.ipynb` are some examples for processing and visualizing data. Hereafter, the structure of the output files is briefly shown.

ASCII files `monitorX.dat` and `eventX.dat` have global information.

`monitorX.dat` records the following information at every time step.

```
timestep time max(log10(vel)) mean(displ) mean(tau/sigma) max(sigma) min(sigma)
error-of-RK stepwidth wall-clock-time
```

For example, if you want to view the temporal change of the maximum slip rate in `gnuplot`, type

```
pl 'monitorX.dat' u 2:3 w l
```

The earthquake catalog is saved in `eventX.dat`. This file lists the event number, the onset time, and the moment magnitude. //

The field data are saved in binary stream files. `xyzX.dat` is the coordinates of the elements. `velX.dat`, `tauX.dat`, `sigmaX.dat`, and `slipX.dat` are slip velocity, shear stress, normal stress, and slip every `interval` time steps, respectively. These are binary stream files.

5.1 Paraview

6 HBI-Fluid

This is a different version of HBI that allows fault-zone fluid flow and pore pressure evolution. To switch to the fluid version, you have to compile `main_fv.f90` instead of `main_LH.f90`. The fluid version is only available for 2D planar/nonplanar faults (2dp or 2dn) and parallel computation is not yet supported.

6.1 Governing equations

The fluid version of HBI additionally solves the fluid pressure diffusion equation

$$\beta\phi\frac{\partial p}{\partial t} = \frac{\partial}{\partial x}\left(\frac{k}{\eta}\frac{\partial p}{\partial x}\right). \quad (7)$$

The boundary conditions for both ends are either Dirichlet (fixed pressure) or Neumann (fixed flow rate), which can be specified by parameter `pbcl` and `pbc`.

Permeability k evolves as.

$$\frac{\partial k}{\partial t} = \frac{V}{L}(k_{\max} - k) + \frac{1}{T}(k_{\min} - k), \quad (8)$$

where T is the healing time and L is the permeability enhancement distance. Initial permeability is `kpmin`, so setting `kpmax = kpmin` disables permeability evolution.

The fluid pressure is related to the shear stress with the effective stress law

$$\tau = f(\sigma - p). \quad (9)$$

| name | type | meaning | |
|--------------------|-----------|--------------------------------|--------------------------------------|
| <code>kpmax</code> | real | Maximum permeability | |
| <code>kpmin</code> | real | Initial (minimum) permeability | |
| <code>beta</code> | real | Compressibility | |
| <code>eta</code> | real | Fluid viscosity | |
| <code>phi0</code> | real | Porosity | |
| <code>bcl</code> | character | Left boundary condition | Dirichlet or Neumann |
| <code>bcr</code> | character | Right boundary condition | Dirichlet or Neumann |
| <code>pbcl</code> | real | Fluid pressure at the left BC | Used when <code>bcl=Dirichlet</code> |
| <code>pbc</code> | real | Fluid pressure at the right BC | Used when <code>bcr=Dirichlet</code> |

6.2 Setting

There are several different initial conditions for fluid models. Users specify one of the following for the parameter `setting` in the inputfile.

6.2.1 injectionp

This is fluid injection at the fault center with constant pressure `pinj` (MPa).

6.2.2 injectionq

This is fluid injection at the fault center with constant flow rate `q0` (m/s). The duration of injection is `tinj` (year). SEAS BP6 can be solved with this setting.

6.2.3 swarm

This is constant pressure injection at the left end ($x = 0$) of the fault. The injection pressure is `pbcl` (MPa).

7 References

Tse & Rice (1986) and Rice (1993) are early examples of earthquake cycle simulations on a planar fault with rate and state friction. For nonplanar fault geometry, Segall (2010) is a great reference to the theoretical foundation of the BEM kernel. For the implementation of `2dn`, we use the formula in Ando et al. (2007). `3dnt/3dht` is based on the expression of Nikkhoo & Walter (2015). The time-stepping algorithm is based on "Numerical Recipe for Fortran90" (Press et al., 2007).

8 Acknowledgements

I learned a lot about the method from Nobuki Kame and Makiko Ohtani when I was an undergraduate student. The basic structure of `HBI` is derived from their codes. `HACApK` library used in `HBI` was primarily developed by Akihiro Ida. I thank Ryoya Matsushima for finding many bugs in the code. I also thank Ryosuke Ando, Brittany Erickson, Eric M. Dunham, Pierre Romanet, and Tetsuya Hoshino for their suggestions that improved the performance, accuracy, stability, and readability of the code.