Manual for CGFDM

Zijia Wang 12231211@mail.sustech.edu.cn

Wenqiang Zhang wenqiang.zhang@mcgill.ca

2023.03

Contents

1. Folder Structure

doc: folder contains the manual for this program
bin: folder contains the binary executable files
srcd: folder contains all CUDA source files
obj: folder contains intermediate object files during compiling
Makefile: file for automatically building programs using GNU make
matlab: folder contains Matlab scripts for pre-processing and post-processing
jobs: folder contains examples for several dynamic rupture modelling examples

2. Build the program

The operating environment used by the author to develop the program is:

cuda9.2, cuda10.0 or cuda10.1
openmpi (gcc-4.8.5,with cuda)
netcdf 4.4.4

Some parameters in Makefile may need to be modified if you want to build in other environments.

CUDAHOME: The directory of cuda, you may need to modify it according to your environment.
MPIHOME: The directory of openmpi, you may need to modify it according to your environment.
NETCDFHOME: The directory of netcdf, you may need to modify it according to your environment.

Other optional parameters about numerical format are annotated in Makefile. Generally, I don't recommend modifying them except the **FreeSurface**. If you want to calculate the full-space case, delete the ON after the FreeSurface, and then recompile the program.

3. Running example

3.1 Coordinate and Unit

Coordinate conversion:

Y-axis is the fault-strike direction

Z-axis is the up-dip direction (vertical upward)

X-axis if the fault-normal direction

Unit:

Variable	Coord	Vp, Vs	density	stress	Slip rate	time	Displacement
Unit	m	m/s	kg/m^3	Pa	m/s	sec	m

3.2 Parameter configuration file "params.json"

The params.json file is a parameter file read by the program, and the main parameters are explained as follows:

TMAX:

Total simulation time (in seconds)

DT:

Time step (in seconds)

DH:

Space step (in meters)

NX/NY/NZ:

Number of grids in x/y/z direction

PX/PY/PZ:

Number of MPI divisions in x/y/z direction. **Note that NX/NY/NZ must be divisible by PX/PY/PZ, and PX cannot be an even number.** PX*PY*PZ is the number of GPU cards to be used.

INPORT_GRID_TYPE:

Enter the type of the grid, where 0 is configured in the code, 1 is input by the user, and if it is 1, read the fault data file specified by the "Fault_geometry".

INPORT_STRESS_TYPE:

Enter the type of initial stress, where 0 is configured in the code, 1 is input by the user, and if it is 1, read the data file specified by the "Fault_init_stress".

Friction_type:

Friction law type, 0 for slip-weakening; 1 for rate-state, Ageing law; 2 for rate-state, slip law

EXPORT_WAVE_SLICE_X:

Wavefield y-z plane slice

EXPORT_WAVE_SLICE_Y:

Wavefield x-z plane slice

EXPORT_WAVE_SLICE_Z:

Wavefield x-y plane slice, set to NZ-1 to output the wavefield of the free surface

igpu:

Specifies to run on a GPU card numbered igpu. This is valid only when a single card is running

Fault_grid:

Fault rupturable region

Asp_grid:

Initial stress asperity area. If INPORT_STRESS_TYPE=1, it is invalid.

mu_s, mu_d, Dc, C0:

Parameters of slip-weakening law, If INPORT_STRESS_TYPE=1, it is invalid.

RS_V0, RS_Vini, RS_f0, RS_fw:

Parameters of rate-and-state friction law, If INPORT_STRESS_TYPE=1, it is invalid.

PML_xxx:

PML absorption boundary related parameters, we don’t need to modify them

vp1, vs1, rho1:

Media parameters. If Media1D or Media3D is set, it is invalid.

bi_vp1,bi_vs1,bi_rho1,bi_vp2,bi_vs2,bi_rho2:

Media parameters of bimaterial fault (you can keep the same with vp1, vs1 and rho1 to make it invalid)

OUT:

Output folder

smooth_load_T:

Time for smooth loading of stress perturbation; if it is less than or equal to 0, it will not be smoothly loaded (stress perturbation is instantaneously loaded at moment 0 and remains unchanged).

Additional remarks:

If the MATLAB you are using is a higher version than 2016, get_params.m can't be used

normally. You can visit https://github.com/fangq/jsonlab to download jsonlab-master package and add it to the MATLAB path. Then change "par = jsondecode(textread(parfile,'%c'));" in the get_params.m script to "par = loadjson(parfile);".

3.3 General steps to run the example

All the examples in the jobs folder can be run as follows.

1) **Before running any matlab scripts, run addmypath.m first!**
2) Run **conf_fault_grid.m** to generate fault geometry data specified by the Fault_geometry keyword in params.json file.
3) Run **conf_fault_stress.m** to generate the fault initial stress data specified by the Fault_init_stress keyword in params.json file.
4) Run **run.sh** to submit single-card or multi-card GPU tasks. The path of mpirun in the script should be modified before running. Another you need to modify is **echo "garray3 slots=4 garray4 slots=4" > nodelists**, where garray3/garray4 is cluster node name, 4 is the number of GPU cards used at this node.
5) Visualization: You can use MATLAB or Python scripts to visualize the results.

   draw_snaps.m/draw_fault_snap.py

   Variables in the process of fault rupture

   draw_snaps_wave_xy.m/draw_wave_snap.py

   Wave field propagation on the surface

   plot_fault_seismo.m/draw_fault_seismo.py

   Various seismograms recorded by virtual stations on faults.

   draw_init_t0.m./draw_init_t0.py

   Rupture speed

3.4 Initial stress data file structure

The stress configuration file is a netcdf file. Assuming that the file is init_stress.nc, the result of ncdump–hinit _ stress.nc is:

netcdf init_stress {

dimensions:

ny = 800 ;

nz = 400 ;

variables:

float x(nz, ny) ;
float y(nz, ny) ;
float z(nz, ny) ;
float Tx(nz, ny) ;
float Ty(nz, ny) ;
float Tz(nz, ny) ;
float dTx(nz, ny) ;
float dTy(nz, ny) ;
float dTz(nz, ny) ;
float mu_s(nz, ny) ;
float mu_d(nz, ny) ;
float Dc(nz, ny) ;
float C0(nz, ny) ;
float a(nz, ny) ;
float b(nz, ny) ;
float L(nz, ny) ;
float Vw(nz, ny) ;
float State(nz, ny) ;
}

Where Tx, Ty, Tz are the components of the initial fault traction in the (x, y, z) coordinate system, and dTx, dTy, dTz are the perturbation of the initial fault traction, namely:

$$\tau_i^0(t) = \tau_i^0 + \Delta\tau_i \cdot G(t), (i = x, y, z)$$

Where

$$G(t) = \begin{cases} \exp\left[\dfrac{(t-T)^2}{t(t-2T)}\right], 0 < t < T \\ 1, \qquad t \geq T \end{cases}$$

The loading time T is specified by the keyword smooth_load_T in params.json. If you don't want to use this nucleation method, just set them all to 0.

3.5 Description of output data file

We take fault_mpi000201.nc as an example.

The six digits in the file name are the number of mpi block, separated by two digits, that is, the first two digits 00 indicate the first mpi block in X direction, the middle two digits 02 indicate the third mpi block in Y direction, and the last two digits 01

indicate the second mpi block in Z direction. These six digits indicate the position of the data on the fault plane. In the visualization script, the data output by each mpi will be put together. Use the ncdump -h command to view the file.

Commonly used variables:

init_t0: Rupture initiation time, which is used to calculate the rupture speed

Vs1: Strike-slip rate

Vs2: Dip-slip rate

Ts1: Shear stress of strike-slip component

Ts2: Shear stress of dip-slip component

Tn: Normal stress

Us1: Strike-slip displacement

Us2: Dip-slip displacement

Us0: Displacement

rake: Rake

4. Additional notes on media settings

1) Homogenous media is set directly in params.json.
2) If layered media is used, add "Media1D": "media1d.dat" to the params.json. File example:

```
# layernum
4
#depth(m) rho vp vs
2500      2100      4500      2600
3000      2750      6050      3500
10500     2800      6230      3600
19000     3050      6750      3900
```

3) If 3-D media is used, add "Media3D": "media3d.nc" to the params.json. Media3d.nc needs to contain Vp, Vs and rho of each grid point. File example:

```
netcdf media3d {
dimensions:
        nx = 160 ;
        ny = 920 ;
        nz = 300 ;
variables:
        double vs(nx, nz, ny) ;
        double vp(nx, nz, ny) ;
        double rho(nx, nz, ny) ;
```

References

Zhang, Z., Zhang, W., & Chen, X. (2014). Three-dimensional curved grid finite-difference modelling for non-planar rupture dynamics. Geophysical Journal International, 199(2), 860–879. https://doi.org/10.1093/gji/ggu308

Zhang, W., Zhang, Z., Li, M., & Chen, X. (2020). GPU implementation of curved-grid finite-difference modelling for non-planar rupture dynamics. Geophysical Journal International, 222(3), 2121–2135. https://doi.org/10.1093/gji/ggaa290