Manual for DRDG3D

Wenqiang Zhang
wqseis@gmail.com
orcid, google scholar, researchgate

Table of Contents

Introduction

This software **DRDG3D** is used for **3D** dynamic rupture (**DR**) modelling, using a discontinuous Galerkin (**DG**) finite element method (Zhang et al., 2022). The program is implemented in Fortran 90 and is only available on CPU platforms, GPUs are currently not supported. NetCDF is used for handling array-oriented scientific data.

License

GNU General Public License (version 3 or higher).

Citation

Cite as: Zhang, Wenqiang, Liu, Yajing, Chen, Xiaofei. A Mixed-Flux-Based Nodal Discontinuous Galerkin Method for 3D Dynamic Rupture Modeling. *Authorea*. November 01, 2022. DOI: 10.1002/essoar.10512657.1

Folder Structure

doc: folder contains the manual for this program
bin: folder contains the binary executable files
src: folder contains all Fortran 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
examples: folder contains examples for several dynamic rupture modelling cases

Requirements

Before building the program, make sure you have the following software installed on your system:

openmpi; gfortran(or **ifort); netcdf; lapack**

Gmsh is used to generate meshes;
meshio is used to convert mesh files *.inp file to *.exo files;
Metis is used for mesh partition for parallel computing;

Cubit/SimModeler is optional. You can use them instead of **Gmsh** for meshing. Export the mesh to Exodus II (*.exo) mesh format.

We will show how to install them.

MacOS

For the macOS, you can use **brew** to install the required packages. **brew** is the package installer for macOS. Refers to https://brew.sh/ for installation.
Optionally, we recommend using **pip** or **conda** to install packages. Refers to https://www.anaconda.com/ or https://docs.conda.io/en/latest/miniconda.html for **conda** installation.

make:

brew install make

gfortran:

brew install gcc

(*installing gcc is time-consuming; gfortran is included in the gcc package)

mpi:

brew install open-mpi

netcdf:

brew install netcdf

brew install netcdf-fortran

BLAS/LAPACK:

brew install lapack openblas

(* macOS already ships with BLAS/LAPACK implementations in its vecLib framework. You don't have to install them if your macOS have vecLib framework)

install meshing software:

gmsh:

pip install gmsh

(or) conda install gmsh

(or) brew install gmsh

(or) directly download from https://gmsh.info/

meshio:

pip install meshio

(or) refers to https://pypi.org/project/meshio/

metis:

pip install metis

(or) brew install metis

Ubuntu

For Ubuntu, use **apt** to install the required packages. **apt** is the automatic package installer for Ubuntu and you don't have to install it.

make:

sudo apt-get install make

gfortran:

sudo apt-get install gfortran

mpi:

sudo apt-get install openmpi-bin

netcdf:

sudo apt-get install netcdf-bin netcdf-dev

BLAS/LAPACK:

sudo apt-get install libblas-dev liblapack-dev

install meshing software:

gmsh:

pip install gmsh

(or) conda install gmsh

(or) sudo apt-get install gmsh

(or) directly download from https://gmsh.info/

meshio:

pip install meshio

(or) sudo apt install meshio-tools

(or) refers to https://pypi.org/project/meshio/

metis:

pip install metis

(or) sudo apt-get install metis

Compute Canada (beluga)

To build the program, “mpif90”, “blas/lapack” and “netcdf-fortran” are required. All three packages have been already installed in *Compute Canada*, just to load the corresponding packages. For example, in *beluga*:

openmpi/4.0.3 is automatically loaded for mpif90
imkl/2020.1.217 is automatically loaded for blas/lapack support
netcdf-fortran is not automatically loaded and can be loaded by:
module load netcdf-fortran

Build the program

We use GNU make to build the program. Some parameters in **Makefile** may need to be modified if you want to build in other environments or use different spatial orders of accuracy.

ENV := mbp (cascadia, beluga, narval)
The program has been tested in macOS 10.14.3 (mbp), Ubuntu 18.04,20.04 (a desktop computer named “Cascadia”), *Compute Canada* (“beluga” and “narval”). Modify it according to different environments.

compiler := gnu (intel)
change to gnu or intel compiler according to your OS.

pOrder := 2
The spatial order of accuracy (O) has the following relationship with pOrder:

$$O = pOrder + 1$$

pOrder can be changed to an arbitrary order. It is recommended that pOrder be 2~4. The program needs to be recompiled after the modification of pOrder, i.e., type in terminal:
make clean
make

NECDF_DIR := /usr/local
The directory of netcdf-fortran, you may need to modify it according to your environment.

Type **make** in the command line to build the programs. The following executable binary files would be generated in the **bin** folder if there is no error during compiling and linking:
bin/exe_get_neigh
bin/exe_part_mesh
bin/exe_solver

Coordinate

Coordinate conversion:
Y-axis is the strike-slip direction
Z-axis is the up-dip direction
X-axis if the fault-normal direction

Unit

Variable	Coord	media	stress	temp	Slip rate	time	displacement	
Unit	km	km/s	MPa	K	m/s	sec	m	

The International System of Units (SI) will be adopted in further releases.

Meshing

We have provided a small-size mesh file "example/tpv5/tpv5.exo" (Figure 1) for a quick test and learn. **Skip** this section when reproducing the example for the first time.

This program only supports the unstructured tetrahedral mesh. The vertex and connectivity information for unstructured meshes is required. Let *Nv* denote the number of vertices, *Nelem* denotes the total number of tetrahedral elements, then the array data of vertices and connectivity should be *Coord*(*Nv*,3) and *Elem*(*Nelem*,4). Additional information for the mesh can also be generated by the meshing software and is not required.

Take "example/tpv5/tpv5.exo" for example. Use **ncdump** in the terminal (or **ncdisp** in Matlab) to dump the information of the Exodus II mesh file:
ncdump –h tpv5.exo
The output information should be something like:

```
netcdf tpv5 {
dimensions:
        num_nodes = 3053 ;
        num_dim = 3 ;
        num_elem = 18891;
        num_el_blk = 24 ;
        num_node_sets = UNLIMITED ; // (0 currently)
        len_string = 33 ;
        len_line = 81 ;
        four = 4 ;
        …
        num_el_in_blk17 = 1070 ;
        num_nod_per_el17 = 3 ;
        num_el_in_blk18 = 128 ;
        num_nod_per_el18 = 3 ;
        num_el_in_blk19 = 126 ;
        num_nod_per_el19 = 3 ;
        num_el_in_blk20 = 1092 ;
        num_nod_per_el20 = 3 ;
        …
        num_el_in_blk24 = 15791 ;
        num_nod_per_el24 = 4 ;
variables:
        float time_whole(time_step) ;
```

```
        char coor_names(num_dim, len_string) ;
        double coord(num_dim, num_nodes) ;
        …
        int64 connect17(num_el_in_blk17, num_nod_per_el17) ;
                connect17:elem_type = "TRI3" ;
        …
        int64 connect24(num_el_in_blk24, num_nod_per_el24) ;
        connect24:elem_type = "TETRA" ;
// global attributes:
                :title = "Created by meshio v5.3.4, 2022-09-08T14:22:12.250971" ;
                :version = 5.1f ;
                :api_version = 5.1f ;
                :floating_point_word_size = 8LL ;
}
```

where **coord** is the vertex array data;
connect17 is the fault connectivity (triangular); **connect20** is the ground surface connectivity (triangular); **connect24** is the volume connectivity (tetrahedron). They are case-dependent, so be careful with the "connectX" variable when using **read_mesh.m** script to read the corresponding data.

Gmsh

Open Examples/tpv5/tpv5.geo file using Gmsh: **gmsh tpv5.geo**
Pay attention! Use Frontal-Delaunay algorithm for surface mesh in Gmsh.
Tools->Options->Mesh->General->Choose Frontal-Delaunay for 2D algorithm).

Click "File->Export->Save as Mesh Abaqus INP (*inp)->save all elements. Then use **meshio** to convert *.inp to Exodus II file (*.exo) by:
meshio convert tpv5.inp tpv5.exo

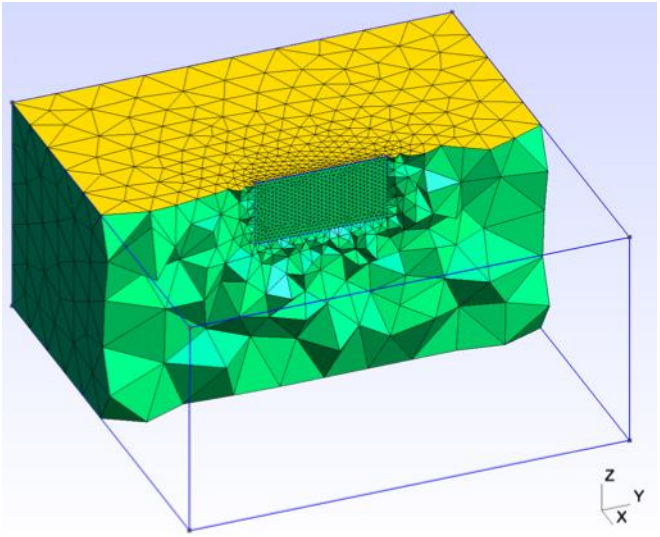

Figure 1. 15791 tetrahedrons generated by Gmsh. Y-Z plane is the fault plane. The yellow surface is the ground surface (Z=0).

Optional meshing software

Coreform Cubit (https://coreform.com/products/coreform-cubit/)

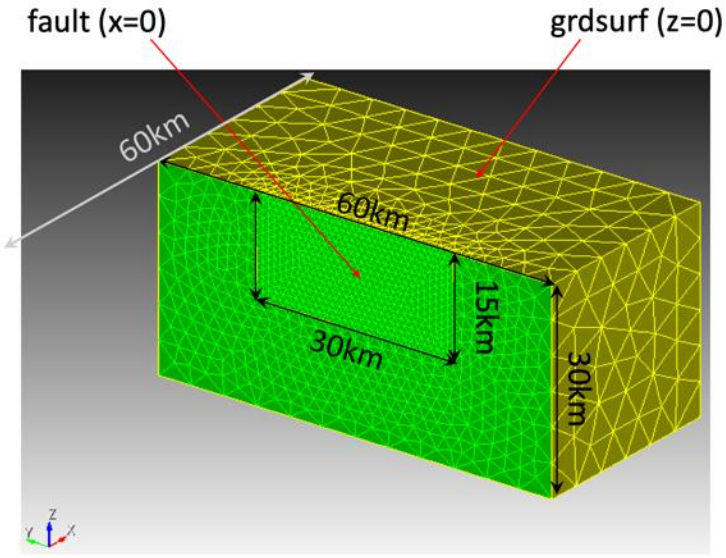

Figure 2. 31216 tetrahedrons generated by CUBIT.

SimModeler (http://www.simmetrix.com/). It can generate the Exodus II (*.exo) mesh file.

Input

Parameter file

parameters.yaml is the parameter configuration file, written in YAML format. YAML is commonly used for configuration files, which can be read by many programming languages, including Matlab, Python, Fortran, C++, etc.

There are many parameters in **parameters.yaml** can be modified. Most of these are optional, which means you don't have to set them in **parameters.yaml**, the program will set them to the corresponding default values. However, if you set them explicitly in the **parameters.yaml** file, new values you set will override the defaults.

nproc:
: the number of parallel processors

simu_time_max:
: the total simulation time (in seconds)

cfl_number:
: <0.5. If the simulation result does not converge, reduce it to a smaller value; the simulation time interval *dt* is not set here (can be overwritten by setting **timestep**), and is automatically set by the program according to the minimum element size, the maximum V_p, and cfl_number

mesh_dir:
: folder name where the mesh files are stored

data_dir:
: folder name where the output data is stored

export_grdsurf_displ: [optional, default value is 0]
export displacement on the ground surface or not
export_grdsurf_strain: [optional, default value is 0]
export strain on the ground surface or not
fault_snap_skip:
time step skip for outputting the snapshot of fault (e.g., output every 10 steps)
grdsurf_snap_skip:
time step skip for outputting the snapshot of the ground surface (free surface)
plasticity: [optional, default value is 0]
Consider Drucker-Prager plasticity when plasticity is 1 (otherwise is 0)
thermalpressure: [optional, default value is 0]
Consider thermal pressurization when thermalpressure is 1 (otherwise is 0)
friction_law: [optional, default value is 0]
0 for slip-weakening;
1 for rate-state, ageing law;
2 for rate-state, slip law;
3 for rate-state with strong velocity-weakening, slip law;

Preprocess

Before running any matlab scripts, run **addmypath.m** first!
Attention: you may need to **modify the directory of metis** if you have installed metis in a different location.
In **addmypath.m,** modify metis_dir = '/usr/local/bin' to your directory of metis if you are not installed metis by "brew install metis"
Find the metis directory by:
which gpmetis
(or) **which pmetis**

conf_mesh.m: modify *tet_id*, *faultsurf_id*, *freesurf_id* according to your mesh file (e.g., tpv5.exo). Use the matlab script **check_surface_id.m** to identify the surface id for the fault surface or the free surface. For example:
check_surface_id('tpv5.exo',17);

conf_media.m: to set homogeneous/heterogeneous velocity structures according to coordinates.

conf_stress.m: the non-uniform initial stress (Tx0, Ty0, Tz0) is set according to the coordinates of the fault plane (e.g., Figure 3).

conf_part.m: we don't need to modify this script.

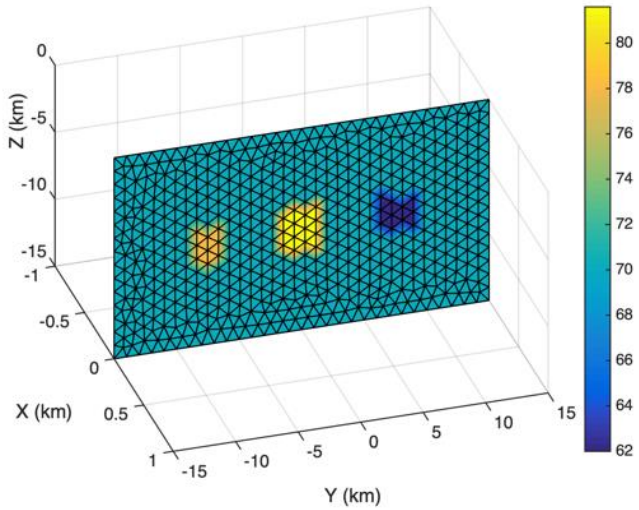

Figure 3. The result of the **conf_stress.m** script, initial shear stress distribution, the left patch is 78 MPa, the right patch is 62 MPa, and the nucleation patch at the center is 81.6 MPa exceeding the shear strength ($\mu_s\sigma_n$=81.24MPa).

run_preprocess.m sequentially calls **conf_mesh.m**, **conf_media.m**, **conf_stress.m**, **conf_part.m** to configure **mesh.nc**
Then call:

../../bin/exe_get_neigh mesh.nc
../../bin/exe_part_mesh mesh.nc

to calculate neighboring information and part the mesh in parallel. The parted mesh is stored as **data/meshVar*.nc**.

Run

Navigate to the directory of examples/tpv5, then type in the command line:
mpirun –np 2 ../../bin/exe_solver

MacBook Pro Retina, 13-inch, Early 2015, 2.7 GHz Intel Core i5, 8 GB 1867 MHz DDR3
Total runtime: ~**20 minutes**

Visualization

draw_fault_snap.m
draw_grdsurf_snap.m
draw_fault_seismo.m
draw_grdsurf_seismo.m
draw_ruptime.m

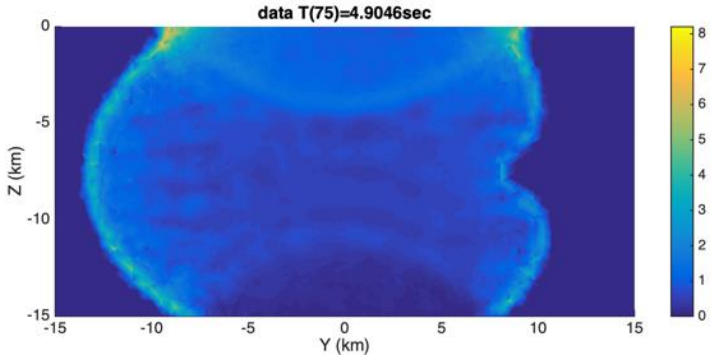

Figure 4. The result of **draw_fault_snap.m**, the snapshot of the fault slip rate (m/s).

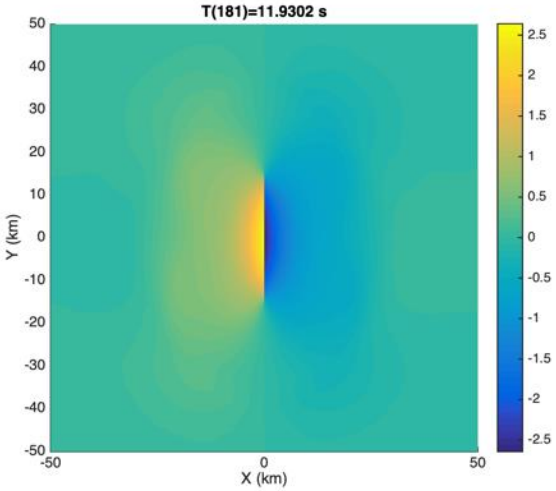

Figure 5. The result of **draw_grdsurf_snap.m**, the snapshot of the displacement **Uy** (m/s) on the ground surface.

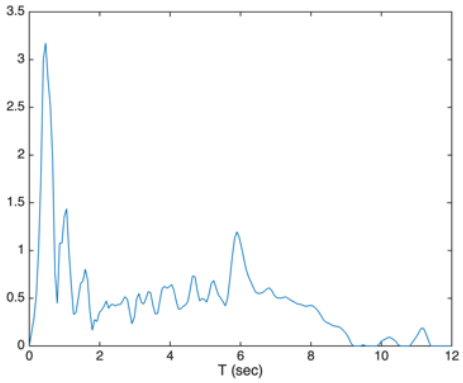

Figure 6. The result of **draw_fault_seismo.m**. The time series of fault slip rate (m/s). Station location (0, 3, -7.5) km.

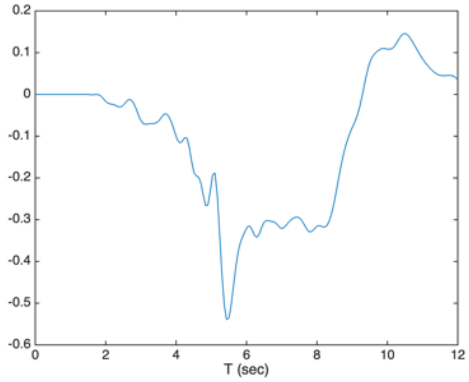

Figure 7. The result of **draw_grdsurf_seismo.m**. The time series of **Vy** (m/s) on the off-fault station (5,5,0) km.

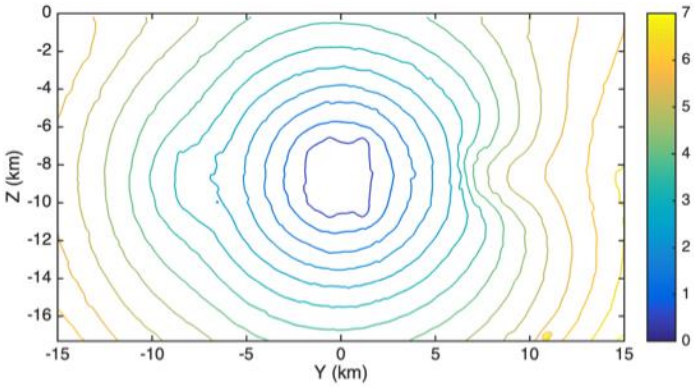

Figure 8. The result of **draw_ruptime.m**. The rupture front contours (every 0.5 s).

Handle NetCDF data

Here we briefly introduce how to handle NetCDF data using Matlab.

Dump the header information of *.nc file:

ncdisp('data/fault_mpi000000.nc');

Read variables from *.nc file:

rate = ncread('data/fault_mpi000000.nc', 'rate');

Refers to Matlab documents (e.g., https://www.mathworks.com/help/matlab/network-common-data-form.html) for more details.

Run jobs on HPC

In the test example, the total number of tetrahedrons is only tens of thousands, which can be run on your laptop to facilitate learning of this software with only 2 CPU cores. However, in actual modelling, the number of elements often reaches several millions, so it is necessary to use a supercomputing cluster, which provides more CPU cores. Here we use *Compute Canada* (https://docs.alliancecan.ca/wiki/Technical_documentation) as an example to show how to submit a job of this program to a supercomputer.

Set "**nproc : 64**" in **parameters.yaml**, then run the Matlab script **run_preprocess.m** to re-generate the data/mesh*.nc, and copy the **data** folder, **job_computecanda.sh**, **parameters.yaml** to *beluga* by:

scp –r data USERNAME@beluga.computecanada.ca:/home/USERNAME/somewhere

or use **Filezilla** (https://filezilla-project.org/) or other FTP/SFTP/SSH software to transfer files between local and remote computers.

Then use the job_computecanada.sh script to submit the job:
sbatch job_computecanada.sh

The runtime of tpv5 using 64 cores in beluga is **98 seconds**.

Check the online document (https://docs.alliancecan.ca/wiki/Running_jobs) to get familiar with running jobs in *Compute Canada*.

Troubleshooting

If you encounter any problems reproducing the examples (or other cases), don't hesitate to contact me (wqseis@gmail.com).

E.g.,
When I run conf_part.m in Matlab, the script gives the following error:
"Error using load
Unable to read file 'tmp.met.dgraph.part.2'. No such file or directory."

Solve: The installation directory of the **metis** software in your computer is different (not /usr/local/bin). Open addmypath.m, and modify "metis_dir = …" according to your environment.