

# **FracG V 0.1**

FractureGraph - Fault and fracture analysis and meshing software

Ulrich Kelka and Stefan Westerlund

CSIRO Deep Earth Imaging

Future Science Platform

2021  
April



# Contents

<b>What is FracG?</b>	<b>2</b>
<b>Installation</b>	<b>3</b>
xutils	3
cmake	3
Manual compilation	3
<b>Executing FracG</b>	<b>4</b>
Options	4
Input files	4
Output	4
Correction parameters and distances	4
Statistical parameters	5
Maximum flow options	6
Gmsh options	6
<b>Trouble shooting</b>	<b>7</b>

## IMPORTANT – PLEASE READ CAREFULLY

This document contains the terms under which CSIRO agrees to licence its Software to you. This is a template and further information relevant to the licence is set out in the Supplementary Licence specific to the Software you are licensing from CSIRO. Both documents together form this agreement. The Software is copyright (c) Commonwealth Scientific and Industrial Research Organisation (CSIRO) ABN 41 687 119 230. Except where otherwise indicated, including in the Supplementary Licence, CSIRO grants you a licence to the Software on the terms of the GNU General Public Licence version 3 (GPLv3), distributed at: <http://www.gnu.org/licenses/gpl.html>. The following additional terms apply under clause 7 of GPLv3 to the licence of the Software that is granted by CSIRO:

EXCEPT AS EXPRESSLY STATED IN THIS AGREEMENT AND TO THE FULL EXTENT PERMITTED BY APPLICABLE LAW, THE SOFTWARE IS PROVIDED "AS-IS". CSIRO MAKES NO REPRESENTATIONS, WARRANTIES OR CONDITIONS OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY REPRESENTATIONS, WARRANTIES OR CONDITIONS REGARDING THE CONTENTS OR ACCURACY OF THE SOFTWARE, OR OF TITLE, MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE, NON-INFRINGEMENT, THE ABSENCE OF LATENT OR OTHER DEFECTS, OR THE PRESENCE OR ABSENCE OF ERRORS, WHETHER OR NOT DISCOVERABLE.

TO THE FULL EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL CSIRO BE LIABLE ON ANY LEGAL THEORY (INCLUDING, WITHOUT LIMITATION, IN AN ACTION FOR BREACH OF CONTRACT, NEGLIGENCE OR OTHERWISE) FOR ANY CLAIM, LOSS, DAMAGES OR OTHER LIABILITY HOWSOEVER INCURRED. WITHOUT LIMITING THE SCOPE OF THE PREVIOUS SENTENCE THE EXCLUSION OF LIABILITY SHALL INCLUDE: LOSS OF PRODUCTION OR OPERATION TIME, LOSS, DAMAGE OR CORRUPTION OF DATA OR RECORDS; OR LOSS OF ANTICIPATED SAVINGS, OPPORTUNITY, REVENUE, PROFIT OR GOODWILL, OR OTHER ECONOMIC LOSS; OR ANY SPECIAL, INCIDENTAL, INDIRECT, CONSEQUENTIAL, PUNITIVE OR EXEMPLARY DAMAGES, ARISING OUT OF OR IN CONNECTION WITH THIS AGREEMENT, ACCESS OF THE SOFTWARE OR ANY OTHER DEALINGS WITH THE SOFTWARE, EVEN IF CSIRO HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH CLAIM, LOSS, DAMAGES OR OTHER LIABILITY.

APPLICABLE LEGISLATION SUCH AS THE AUSTRALIAN CONSUMER LAW MAY APPLY REPRESENTATIONS, WARRANTIES, OR CONDITIONS, OR IMPOSES OBLIGATIONS OR LIABILITY ON CSIRO THAT CANNOT BE EXCLUDED, RESTRICTED OR MODIFIED TO THE FULL EXTENT SET OUT IN THE EXPRESS TERMS OF THIS CLAUSE ABOVE "CONSUMER GUARANTEES". TO THE EXTENT THAT SUCH CONSUMER GUARANTEES CONTINUE TO APPLY, THEN TO THE FULL EXTENT PERMITTED BY THE APPLICABLE LEGISLATION, THE LIABILITY OF CSIRO UNDER THE RELEVANT CONSUMER GUARANTEE IS LIMITED (WHERE PERMITTED AT CSIRO'S OPTION) TO ONE OF FOLLOWING REMEDIES OR SUBSTANTIALLY EQUIVALENT REMEDIES:

- (a) THE REPLACEMENT OF THE SOFTWARE, THE SUPPLY OF EQUIVALENT SOFTWARE, OR SUPPLYING RELEVANT SERVICES AGAIN;
- (b) THE REPAIR OF THE SOFTWARE;
- (c) THE PAYMENT OF THE COST OF REPLACING THE SOFTWARE, OF ACQUIRING EQUIVALENT SOFTWARE, HAVING THE RELEVANT SERVICES SUPPLIED AGAIN, OR HAVING THE SOFTWARE REPAIRED.

IN THIS CLAUSE, CSIRO INCLUDES ANY THIRD PARTY AUTHOR OR OWNER OF ANY PART OF THE SOFTWARE OR MATERIAL DISTRIBUTED WITH IT. CSIRO MAY ENFORCE ANY RIGHTS ON BEHALF OF THE RELEVANT THIRD PARTY.

## What is FracG?

FRACG is a command line based software that performs analysis of discontinuity data (e.g. faults and fractures). The input need to be provided as line-vector data in shape-file format. The software performs a statistical analysis on the data that can be extended by providing relevant raster data in GeoTIFF format. The main analysis derives metrics from a graph representation. The line data can also be converted into 2D and 3D finite element meshes. Note that the 3D meshing is considered experimental at the moment and only a vertical extrusion to a specified depth can be performed at the moment.

## Installation

*FracG* is designed for Debian GNU/Linux and requires third-party libraries as outlined below. First, get the latest GDAL/OGR version, add the PPA to your sources:

```
sudo add-apt-repository ppa:ubuntugis/ppa && sudo apt-get update
```

Now the necessary libraries can be installed from the terminal:

```
sudo apt-get install build-essential \  
libgdal-dev \  
libboost-all-dev \  
liblapack-dev \  
libblas-dev \  
libgsl-dev \  
libgmsh-dev \  
libarmadillo-dev
```

Export the environmental variables for gdal

```
export CPLUS_INCLUDE_PATH=/usr/include/gdal  
export C_INCLUDE_PATH=/usr/include/gdal
```

To obtain *gmsh* visit <http://gmsh.info/>. Note that open cascade is used for creating the mesh. Three options are available for compilation:

### xutils

This options uses a shell script and and Imake file.

```
sudo apt-get install xutils-dev
```

If the libraries are all installed correctly the permission for the file “*install.sh*” need to be changed.

```
cd ...FracG  
sudo chmod -x install.sh
```

Now the installation script can be executed:

```
sudo ./install.sh
```

*FracG* can now be executed from the command line.

### cmake

This option is recommended

```
sudo apt-get install cmake
```

In the *FracG* directory, type;

```
mkdir build \  
cd build \  
cmake .. \  
sudo make install
```

*FracG* can now be executed from the command line.

## Manual compilation

You can compile *FracG* without the install script or cmake.

```
cd ...FracG/src  
g++ -o FracG main.cpp graph.cpp GeoRef.cpp geometrie.cpp stats.cpp model.cpp  
-larmadillo -lgsl -lgdal -lgmsh
```

# Executing FracG

After installation with cmake FracG will be set as a global executable in your environment. In a terminal choose your current directory in which files you wish to analyse are located. The first argument is a shape file containing the fault or fracture traces. This is always required. The second and thirds optional arguments are the raster file in GeoTiff format and a point shape file containing source and target node for shortest path.

```
FracG <.shp> <.tif> <.shp>
```

## Options

FracG has several parameters that can be defined by the user. To see what options are available type:

```
FracG --help
```

The optional parameters can be set after the name of input file. They do not have to be in a specific order and you can add as many of them as you want:

```
FracG <.shp> --option1 --option2 ...
```

## Input files

The input files can be defined on the command line. This might be necessary if several input parameters should be user-defined. In the following, we list the optional parameters defined by a keyword, their data-type and their default value.

**shapefile** |< *std :: string* >

*default:* na

Path/name of the line-shape-file including extension.

**raster\_file** |< *std :: string* >

*default:* na

Path/name of the raster-file in GeoTIFF format including extension. This file has to be in the same reference system as the line-shape file.

**source\_file** |< *std :: string* >

*default:* na

Path/name of the point-shape-file including extension. This file needs to contain two points that will be used for computing the shortest path and maximum flow between them. If not source file is given the shortest path will not be computed. This file has to be in the same reference system as the line-shape file.

## Output

**out\_dir** |< *std :: string* >

*default:* fracg\_output\_ + <name-of-shapefile>

The main directory in which all results will be written.

**graph\_results\_file** |< *std :: string* >

*default:* graph.vertices & graph.branches

Filename to save graph analysis results to.

**graph\_results\_folder** |< *std :: string* >

*default:* graph

Foldername to save graph analysis results to.

## Correction parameters and distances

**dist\_thresh** |< *double* >

*default:* 1

Distances under this distance threshold will be considered the same location. Used for merging line segments and as distance in the point index map of the graph. The units are meters.

**angl\_threshold** |< double >

*default:* 25

Maximum orientation difference in degrees for merging two segments whose tips are with the critical distance.

**dfd\_threshold** |< double >

*default:* 1 Threshold of cumulative discrete frechet distance of two line-strings. If the cumulative distance is below this threshold the lines will be considered duplicates and one of them will be removed.

**split\_dist\_thresh** |< double >

*default:* = dist\_thresh

Distance threshold to use in splitting faults into segments, for considering nearby but separate faults to actually overlap. Used for fixing flaws in digitisation leading to false intersection classification. The units are in meters.

**spur\_dist\_thresh** |< double >

*default:* = dist\_thresh

Distance threshold to use in removing spurs, remove spurs which are shorter than this distance. Used to correct for false intersection classification. The units are in meters.

**classify\_lineaments\_dist** |< double >

*default:* = dist\_thresh

Distance used in to classify lineaments in terms of intersection number along their trace. This distance represents the buffer width around the line and intersections within this distance are counted for the classification. The units are in meters.

**raster\_stats\_dist** |< double >

*default:* = 1.25

Distance used for analysing raster data for the line-strings. The distance is the buffer width around the traces for computing mean values, the length of the profile lines for computing cross gradient, parallel gradients and cumulative cross-gradients along the line-strings. The unit is the number of pixels of the raster of the input raster. Note that the distance in meters is derived as the mean of the x- and y-cellsizes multiplied by this factor.

**raster\_spacing** |< double >

*default:* 1000

Pixel size of output density/intensity maps. The units are in meters.

**raster\_spacing2** |< double >

*default:* 500

Pixel size of output distance maps. The units are in meters.

**isect\_search\_size** |< double >

*default:* = raster\_spacing = 1000

Search for intersections within this distance. Using a circular sampling window this is the radius of the window. The units are in meters.

**resample** |< bool >

*default:* false

Resampling all created raster files to a 10<sup>th</sup> of the initial cell size using cubic spline interpolation.

## Statistical parameters

**angle\_param\_penalty** |< double >

*default:* 2

Penalty per parameter, when fitting Gaussian distributions to the angle distribution.

**scanline\_count** |< int >

*default:* 100

Number scanlines to construct.

**scanline\_spaceing** |< double >

*default:* 10

Minimum spacing of scanlines in meters.

**graph\_min\_branches** |< int >

*default:* 100

Number of branches that build a connected component. Connected components with branch numbers above this threshold will be analysed separately in addition to the analysis of the entire dataset.

**component** |< int >

*default:* -1

If greater than zero extract this connected component from the graph and build a line shape-file from it.

### Maximum flow options

**max\_flow\_cap\_type** |< std :: string >

*default:* l

Type of capacity to use in maximum flow calculations, l for length, o for orientation, lo for both.

**max\_flow\_gradient\_flow\_direction** |< std :: string >

*default:* right

Target direction of the gradient-based maximum flow (towards left, right, top, or bottom).

**max\_flow\_gradient\_pressure\_direction** |< std :: string >

*default:* right

Target direction of the gradient-based maximum flow pressure (towards left, right, top, or bottom).

**max\_flow\_gradient\_border\_amount** |< double >

*default:* 0.05; For gradient-based maximum flow, the border features are those that intersect with the bounding box that is reduced by this amount (0 to 1).

### Gmsh options

**gmsh\_cell\_count** |< int >

*default:* 10

Target element count in x and y direction. For rectangular domains this will be the target mean element size along x and y. This will yield the usual characteristic length (cl) of the model

**gmsh\_show\_output** |< bool >

*default:* false

Show output of gmsh while meshing and the final mesh in the gmsh GUI.

**gmsh\_min\_cl** |< double >

*default:* cl/10

Minimum characteristic length. By default this will be a 10<sup>th</sup> of the usual characteristic length (cl).

**gmsh\_max\_dist** |< double >

*default:* cl/2

Maximum distance for refinement around side-sets in 2D.

**gmsh\_min\_dist** |< double >

*default:* cl/4

Minimum distance for refinement around side-sets in 2D.

*default:* 100

**gmsh\_ext\_depth** |< double >

Extrusion depth for 3D mesh in meters.

**gmsh\_name\_ss** |< bool >

*default:* false

Name sideset individually.

**gmsh\_sample\_cell\_count** |< *int* >

*default:* 2

Number of sampling windows from which 2D-meshes should be generated.

**gmsh\_sample\_count** |< *int* >

*default:* 10

Target element count in x and y direction for the sampling windows. For rectangular domains this will be the target mean element size along x and y. This will yield the usual characteristic length (cl) of the model.

**gmsh\_sample\_show\_output** |< *bool* >

*default:* false

Show out put of gmsh while meshing and the final mesh in the gmsh GUI for every sampling window.

**gmsh\_in\_show\_meters** |< *bool* >

*default:* false

Convert coordinates into meters. This can be necessary fro small scale models.

## Trouble shooting

FracG comes without any warranty. If you encounter any bugs please let us know and we will do our best to fix them as soon as possible. In some cases it might be necessary to install Aramdillo manually. In that case please follow the instruction on the web page: [ARMADILLO](#)

Please send any inquires or bug-reports to us via [email](#).