FracG V 1.0.0

FRACTUREGRAPH -

Fault and fracture analysis and meshing software

Ulrich Kelka and Stefan Westerlund

CSIRO

Deep Earth Imaging Future Science Platform

2021
September

Contents

What is FracG?

FracG is a command line Debian GNU/Linux based software that performs analysis of discontinuity data (e.g. faults and fractures). Currently, the framework is only tested for Ubunutu 20.04 LTS.
The input need to be provided as line-vector data in shape-file format. Prior to every analysis FracG will try to clean up the given vector data removing duplicate entries and flaws in the topology. The analysis can be performed with or without obtaining the best-fit models for length distribution, meshing or including raster data. The generic analysis are:

- Models of the length distribution (exponential, log-normal, or power-law) and of the principal orientations (Gaussians fitted to the Kernel density estimation) are obtained by automated fitting
- Statistical parameters including scan line analysis are exported (.csv)
- Box-counting to obtain fractal dimension
- Density maps are generated (frequency, intersections, distances)
- Georeferenced graph is created
- Graph algorithms:
  - *Betweenness Centrality*
  - *Minimum Spanning tree*
  - *Shortest path*
  - *Maximum Flow*
- Classify discontinues (orientation, intersection numbers)
- Analysis can be extended by including raster data (e.g. DEM)
  - Extract raster values for line geometries
  - Build georeferenced graph including raster data as edge and vertex weights
- 2D Finite Element(FE) conforming meshes (discontinuities as side sets)
  - Generate series of randomly sampled FE-meshes (rectangular windows)

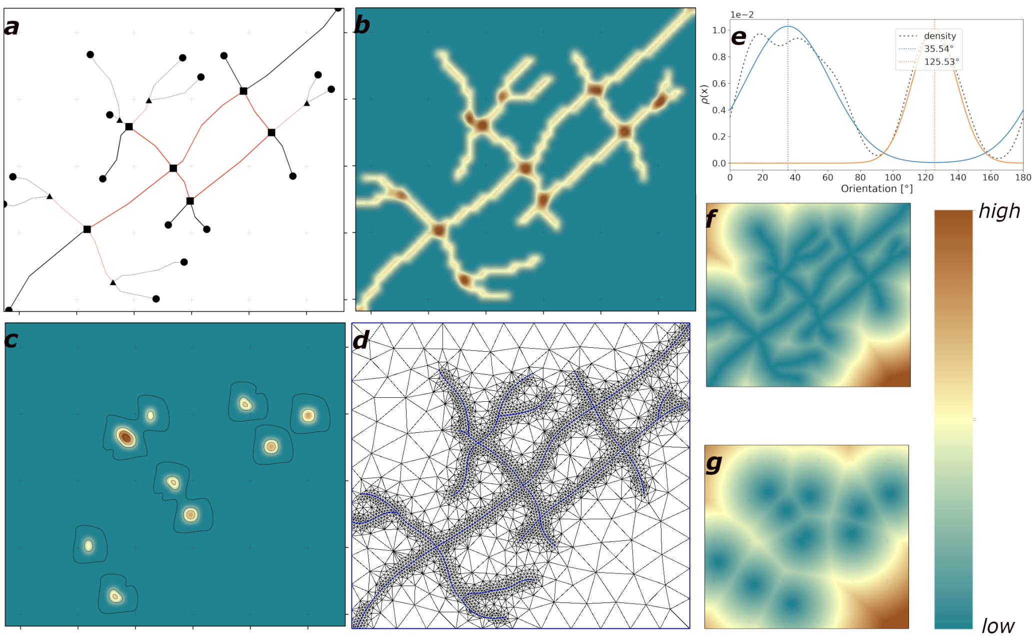

Figure 1: **a** Intersection based graph representation; **b** Resampled line density map (P20) contoured by line intensity (P21); **c** Intersection density (I20) contoured by intersection intensity (I21). **d** 2D FE mesh with refinement around side-sets; **e** Automatic fitting to principal orientation; **f** Line distance map; **g** Centroid distance map

The software can be obtained from: https://bitbucket.csiro.au/scm/fracg/fracg.git.

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 gmsh's open cascade engine is used for the meshing..

cmake

Obtain cmake first.

```
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.

Executing FracG

After installation with cmak 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 defined by the user. To see what options are available type:

```
FracG --help
```

The optional parameters can be set after 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
Folder 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.

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-cellsize multiplied by this factor.

di_raster_spacing |< *double* >
default: 1000
Pixel size of output density/intensity maps. The units are in meters.

dist_raster_spacing |< *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^{th} 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.

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 $|< int >$

gmsh_crop $|< double >$
default: 0
Amount of boarders to cut in percent.

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 out put of gmsh while meshing and the final mesh in the gmsh GUI.

gmsh_point_tol $|< double >$
default: 0.1
Point tolerance used by gmsh.

gmsh_min_cl $|< double >$
default: cl/10
Minimum characteristic length. By default this is will be a 10^{th} 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_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_sw_size |< *double* >
default: -1
If set positive, use a fixed size for all sampling windows.

gmsh_in_meters|< *bool* >
If set true, convert map units into meters.

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.

Skip certain analysis

skip_length_distribution |< *bool* >
Skip calculating the length distribution analysis.

skip_betweenness_centrality |< *bool* >
Skip calculating the betweenness centrality

skip_meshing |< *bool* >
Skip meshing.