MODFRAC, Fracture Network Modeler and Mesher

Technical Description

G. Pichot[1] H. Borouchaki[2] P. Laug[3]

1. Inria Paris & Université Paris-Est, CERMICS (ENPC), geraldine.pichot@inria.fr
2. Université de Technologie de Troyes, houman.borouchaki@utt.fr
3. Inria Saclay (retired)

1 Introduction

It is well known that fractures play a major role in many applications of geology (water resources, deep storage, geothermal energy, among others) and cannot be neglected in subsurface modeling as they are preferential flow paths. Fractures are ubiquitous and have a large range of sizes (from a few centimeters to several kilometers) and apertures (which determine their transmissivity).

A possible strategy for modeling these fracture networks is the *Discrete Fracture Network* (DFN) approach. Fractures are then modeled as ellipses or polygons whose properties (orientation, size, position, transmissivity) are governed by statistical laws derived from field observations. In numerical simulations, the three-dimensional domain Ω is typically a parallelepiped (here called a *cube* for the sake of brevity) containing several thousand or even **more than a million fractures**.

The MODFRAC software, from such a DFN model, calculates the intersections of the fractures with each other and with the faces of the *cube*, completes this model with geometric and topological information, and generates a mesh of this model.

The execution takes place in several stages:

1. Reading the software control parameters.
2. Reading the fracture network.
3. Optionally, reading the description of the wells and other geometric constraints.
4. Possible conversion of ellipses into polygons (on user request).
5. Selection of all fractures inside the *cube*.
6. Calculation of the intersections of the fractures with each other and with the faces of the *cube*. These intersections are straight segments on the fractures and faces of the *cube*, and eventually produce arcs of ellipses at the boundary of the fractures.
7. In the case of a mesh performed on the DFN backbone only (the surrounding porous media is not meshed), selection of fractures connected to given faces of the *cube*, using a graph structure. Only these fractures will be useful for later numerical simulations.
8. Calculation of the intersections of the preceding straight segments with each other. These intersections are points at the intersection of 3 faces which are either fractures or faces of the *cube*. These intersection points subdivide the segments into sub-segments. It is essential to manage this data in a strictly consistent manner.
9. Transmission of the preceding information to an "indirect" surface mesher, where the three-dimensional mesh results from the construction of planar meshes of the parameter domains. The meshing methodology is based on a combined frontal-Delaunay approach in a Riemannian context.
10. Improving minimum mesh quality.

11. Writing the surface mesh, which could possibly be used by an independent volume mesher.
12. Writing files that can be used for numerical simulations.

The software makes maximum use of the available computing power and most of the steps are parallelized using an arbitrary number of *threads*, which greatly increases processing speed on common machines comprising one or more multi-core processors.

2 Features

MODFRAC software is driven by the ***modfrac.env*** text file. In this file, the user can modify each parameter (or environment variable) by means of a keyword and a value (or several values). It is appropriate here to distinguish between two main classes of parameters. The first, called generic parameters, are used in a wide variety of mesh applications. We give below the generic parameters used in the context that concerns us, namely meshing fracture networks. The second, called specific parameters, have been added to control certain particular aspects of the MODFRAC software.

Generic parameters

The following list gives, for each generic parameter, its keyword, its type, its description and its default value.

- ***angle_mesh*** : real (0 to 90) indicating the angular tolerance for the geometric discretization of the ellipses bounding the fractures (if *hgeo_flag* is set). This is the angle (in degrees) between an edge of the discretization and the tangent to the ellipse at each end. The default value is 8°.
- ***export_amadeus***: integer (0 to 4) indicating the content of the mesh file *amadeus.mesh* which will be generated: 0 no file, 4 file readable by the GHS3D volume mesh generator, otherwise visualization file (with Edges and Normals fields). If *FRACporous* = 1 and *export_amadeus* ≠ 4, two files are generated: *amadeus.mesh* for the volume mesher and *amadeus-visu.mesh* for the visualization (it contains edges information). The default value is 0.
- ***gradation***: real in the interval [1.1, 1.e+30] indicating the maximum ratio of lengths between any two adjacent edges (1.e+30 means that no gradation is desired). The default value is 1.e+30.
- ***hgeo_flag***: boolean (0, 1) indicating that a geometric mesh is desired (you can simultaneously set *hgeo_flag* and *hphy_flag*). In our context, the mesh is controlled by the curvature of the ellipses bounding the fractures. The default value is 0.
- ***hgeomin***: real indicating the minimum size of edges for a geometric mesh (*hgeo_flag* ≠ 0). The default is *diag* * 0.001, where *diag* is the length of the diagonal of the *cube*.
- ***hgeomax***: real indicating the maximum size of edges for a geometric mesh (*hgeo_flag* ≠ 0). The default is *diag* * 0.1, where *diag* is the length of the diagonal of the *cube*.
- ***hphy_flag***: boolean (0, 1) indicating that a physical mesh is desired, i.e. respecting the sizes specified by the user (you can simultaneously set *hgeo_flag* and *hphy_flag*). The default value is 1.
- ***hphydef***: real indicating the size of the edges for a physical mesh (*hphy_flag* ≠ 0). The default is *diag* * 0.01, where *diag* is the length of the diagonal of the *cube*.
- ***hphymin***: real indicating the minimum size of the edges for a physical mesh (*hphy_flag* ≠ 0). The default value is that of *hgeomin.*
- ***hphymax***: real indicating the maximum size of edges for a physical mesh (*hphy_flag* ≠ 0). The default value is that of *hgeomax*
- ***intermedfile***: boolean (0, 1) activating the generation of intermediate files in order to check the correct running of the software. If *intermedfile*=1 and *refs*=0, only general files are created. If *refs* ≠ 0, *intermedfile* is forced to 1 and files concerning more particularly the fracture *refs* are also created. The default value is 0.
- ***memory***: integer indicating the maximum number of vertices generated on each fracture. The default value is 100,000.

- ***number_of_threads***: integer *n* indicating the desired number of *threads* on which the parallel parts of the software are executed (if it is 0 then the process remains sequential). For example, on a desktop computer equipped with a single quad-core processor, good performance will be obtained with $n = 4$. This performance is sometimes slightly improved with $n > 4$, due to a hyper-threading phenomenon. The default value is 0.
- ***refs*** : integer giving a fracture number (see also *intermedfile* parameter). The default value is 0. The correspondence between *refs* and the original fracture number is in the *modfrac_frac_meshed.vector* file (see below).
- ***size_computation***: boolean (0, 1) forcing size gradation around small edges. The default value is 0.
- ***verb***: integer (0 to 100) indicating the percentage of verbosity, or level of printing of software messages. The default value is 0.

Specific parameters

The parameters below have been added especially for the MODFRAC software.

- ***FRACconforming***: boolean (0, 1) indicating that the discretizations of the sub-segments generated in step 8 must be conforming. Otherwise, if a sub-segment is at the intersection of two fractures f_1 and f_2, the number of edges of the discretization of this sub-segment on the fracture f_1 can be different from the number of edges on the fracture f_2. The default value is 1.
- ***FRACcube_faces***: integer *n* indicating the faces of the *cube* to which the fractures must be connected (step 7). This integer contains several digits, each digit representing a face of the *cube*. By convention, the faces are numbered 1 ($x = xmin$), 2 ($x = xmax$), 3 ($y = ymin$), 4 ($y = ymax$), 5 ($z = zmin$) and 6 ($z = zmax$). For example, if $n = 12$, all fractures connected (directly or indirectly) to faces 1 and 2 will be retained. The default value is 0 (all fractures, connected or not, are retained).
- ***FRACcubemin***: 3 real numbers (*xmin*, *ymin*, *zmin*) representing the minimum coordinates of the corners of the *cube*. This parameter is required.
- ***FRACcubemax***: 3 real numbers (*xmax*, *ymax*, *zmax*) representing the maximum coordinates of the corners of the *cube*. This parameter is required.
- ***FRACell_to_pol***: integer *n* indicating that each ellipse must be replaced by a polygon with *n* sides (step 4). The default value is 0 (in this case, ellipses are not converted to polygons before meshing).
- ***FRACeps_abs***: real indicating an absolute threshold of distance ε_a below which two points are considered as coincident. The default value is $\varepsilon_a = \varepsilon_r$ *diag*, where ε_r is given by the *FRACeps_rel* parameter and *diag* is the length of the diagonal of the *cube*.
- ***FRACeps_rel***: real indicating a relative threshold of distance ε_r to calculate the absolute threshold ε_a defined by the parameter *FRACeps_abs*. The default is 10^{-6}.
- ***FRACfilename***: character string indicating the name of the fracture network file. The default is the string "frac-no.vector".
- ***FRACfixed_constraints***: boolean (0, 1) indicating to fix (freeze) certain vertices during the final optimization of the mesh (step 10), in order to compare certain numerical tests very accurately. If this boolean is true, the optimization does not move the points located on the straight geometric constraints (starting with a 1 in the *constraints.dat* file). The optimization also does not move the vertices located on the 4 sides of the *cube* faces that contain these constraints, to fully preserve the constrained areas. The default value is 0 (free vertices).
- ***FRACL***: real *d* defining a real cube (and not any parallelepiped) centered at the origin and with side *d*. *FRACL d* is equivalent to *FRACcubemin* $-d/2$ $-d/2$ $-d/2$ and *FRACcubemax* $+d/2$ $+d/2$ $+d/2$.
- ***FRACpad_zeros***: boolean (0, 1) indicating that the *inter-edges.dat* and *inter-cube-edges.dat* files must be padded with 0s in order to make the number of elements per line constant. The default is 0 (smaller file sizes).

- ***FRACporous***: boolean (0, 1) indicating that the numerical simulations relate to a porous medium. In this case, all fractures are meshed (***FRACcube_faces*** is forced to 0), the faces of the *cube* are also meshed and the output mesh of MODFRAC can be used as input of a volume mesher. The default value is 0.
- ***FRACqmin***: real *qmin* indicating the minimum quality required for the mesh (step 10). This value, generally small (of the order of 10^{-4}), is often essential for later numerical simulations. For any triangle of quality $q < qmin$, the MODFRAC software improves its quality up to a value $q' \geq qmin$ by moving its vertices if possible. The default value is 0.0 (no improvement requested).
- ***FRACsimulation***: boolean (0, 1). If it is 0, only the mesh file *amadeus.mesh* is generated. If it is 1, other useful files for numerical simulations are generated. The default value is 1.
- ***FRACwrite_frac***: Boolean (0, 1) activating the conversion of a fracture network in *.disk* format into a network in *.vector* format (see the description of this format below). The name of the output file is *frac-no.vector*. The default value is 0 (no conversion).
- ***FRACwrite_inter***: integer (0, 1, 2) indicating the option for generating inter-* files (see below). The default is 2.
- ***FRACwrite_subdomain***: integer (0 to 7) indicating the option for generating subdomain files (see below). The default value is 0.

After reading the parameters specified by the user (step 1), the MODFRAC software reads a file defining a network of fractures (step 2). The name of this file is given by the *FRACfilename* parameter. The suffix of this name (*.vector*, *.disk* or *.geo*) indicates the format of the file.

Fracture file with *.vector* suffix, in particular *frac-no.vector* (input)

In this format (cf. Table 9 of reference [1]), each fracture is delimited by an ellipse defined by its center, its normal, its major axis and its minor axis. The first line of the file specifies the list of variables used:
no Laxis_maj Laxis_min c_x c_y c_z n_x n_y n_z no_fracture u_x u_y u_z v_x v_y v_z

Fracture file with *.disk* suffix (input)

In this other format [2], each fracture is delimited by a circle. The first line of the file again gives the variables used: *label id xc yc zc dip dipdir half_length aperture ...*

- *label* and *id* are variables that can be ignored
- *xc*, *yc* and *zc* are the coordinates of the center of the circle
- *dip* is the dip and *dipdir* the azimuth (geological data giving the normal to the fracture)
- *half_length* is the radius of the circle
- *aperture* is the aperture, which determines the transmissivity of the fracture
- Additional variables, following the *aperture* variable, depend on the context of use.

The second line contains 2 integers: the total number of fractures and the number of additional variables following *aperture*. The following lines contain, for each fracture, the values of the variables that describe it.

Fracture file with *.geo* suffix (input)

In this format of the GMSH software [3], each fracture is delimited by a convex planar polygon. The syntax is recalled below in a simplified way:

Point(p) = {x, y, z, h}
Line(l) = {p, q}
Line Loop(ll) = {l1, l2, l3, ...}
Plane Surface(s) = {ll}

In this way, we successively define points, straight segments joining two points, loops of straight segments and finally planar surfaces delimited by loops. The MODFRAC software deduces the origin and the normal of each polygon, then the intersections of the polygons with each other (even if they are already given in the *.geo* file) in order to guarantee the topological coherence of the whole.

File with the well description *wells.geo* (input)

The exploitation of underground resources often requires the drilling of one or more wells. The optional file *wells.geo* (in *.geo* format as before) gives the description of such wells (step 3) in the form of planar polygonal faces. An example is given in Section 3.

Note that the *wells.geo* file can be used to describe not only wells, but also any surface made up of one or more planar polygons.

File with the geometric constraints *constraints.dat* (input)

As indicated in the introduction, optional step 3 gives the possibility of specifying other geometric constraints than the previous ones. Each constraint is defined by a text line having one of the following forms:

a) 1 Px Py Pz Qx Qy Qz
b) 2 Ox Oy Oz Nx Ny Nz
c) 2 Ox Oy Oy Px Py Pz Qx Qy Qz

If the line begins with a 1, it is a linear constraint (1D), which is a straight segment joining points P and Q located on the same face of the *cube*.
If it starts with a 2, it is a surface constraint (2D), which is a plane intersecting the *cube*. This plane can be defined by its origin O and its normal N (6 reals), or by its origin O and two points P and Q belonging to it (9 reals).

Files for fracture selection *frac_meshed.vector* and *frac_unmeshed.vector* (input)

The *frac_meshed.vector* (resp. *frac_unmeshed.vector*) file, if it exists, gives the list of the numbers of the fractures to be meshed (resp. not to be meshed).

Execution

After having read the previous data, the MODFRAC software runs as indicated previously by calculating the intersections of the fractures with each other and with the faces of the *cube*, by selecting the only useful fractures, by completing this model with geometric and topological information, and by generating a mesh. Various information is displayed (depending on the *verb* parameter), in particular the calculation times in the sequential and parallel parts. The files generated by the software are described below.

Mesh file *amadeus.mesh* (output)

The mesh file written by MODFRAC is in *.mesh* format (cf. Section 7.2.1 of report [4]). This is a text file that can contain several fields, including:

- Vertices : coordinates of the vertices.
- Edges : edges of the discretization of the boundaries of the fractures and their intersections.
- Triangles : triangles of the mesh.

As a reminder, the *export_amadeus* parameter allows you to add other fields or to generate an additional *amadeus-visu.mesh* file. The *.mesh* format is compatible with many 3D visualization and volume mesh software, in particular GHS3D [5].

Files generated for numerical simulations (output)

In the general case (default values of the *FRACsimulation* and *FRACwrite_** parameters), the files below are generated for subsequent numerical simulations, typically in a MATLAB environment [6].

- ***info.dat****:* this file contains various information, namely *maxvglo* = largest global vertex number of the extremities of sub-segments, *nclassic* = number of "classic" fractures (selected in the input file, and not created artificially by geometric constraints), *nconstraints* = number of geometric constraints, *nsegff*, *nsegfc*, *nsegcc* = number of fracture/fracture, fracture/cube and cube/cube intersection segments (the latter being 12 in number), *nsubsegff*, *nsubsegfc*, *nsubsegcc* = number of fracture/fracture, fracture/cube and cube/cube intersection sub-segments, *ndom* = number of domains, *nsubdom* = number of subdomains.
  Face numbering: by convention, "classic" fractures are numbered from 1 to *nclassic*, fractures created by geometric constraints from *nclassic*+1 to *N* = *nclassic*+*nconstraints*, and faces of the *cube* from *N*+1 to *N*+6.
- *edges.vector* : mesh edges with their characteristics [1, Table 11].
- *inter-edges.dat:* for each discretized intersection, its number, number of a fracture containing it (for non-conforming cases), global numbers of its edges of the discretization.
- *inter-no.vector*: for each discretized intersection, line number, number of edges, numbers of the two fractures creating the intersection, number of a fracture (for non-conforming cases), number of the intersection [1 , Table 16].
- *inter-vertices.dat*: for each discretized intersection, its number, number of a fracture containing this intersection (for non-conforming cases), global numbers of the vertices of the discretization.
- *modfrac_frac_meshed.vector*: for each fracture selected and meshed by the MODFRAC software, its original number in the *frac-no.vector* file.
- *null-patches.vector* : numbers of non-meshed fractures, if any.
- *numbers.vector* : for each fracture, number of vertices, number of edges and number of triangles.
- *quality-data.dat, quality-hist.dat*: information on the quality of the mesh, in free form or in a form adapted to an Excel-type spreadsheet.
- *triangles.vector*: for each triangle, local numbers of its vertices and edges [1, Table 10].
- *vertices-glo.vector*: 3D coordinates of the vertices of the mesh.
- *vertices.vector*: 2D coordinates of mesh vertices [1, Table 12].

Intermediate files conditionally generated (output)

If *intermedfile* ≠ 0, intermediate files are generated in order to check the correct execution of the software. The *intermed-tln3d.mesh* file allows you to visualize all the intersection sub-segments in 3D. The *intermed.geo* file gives the geometric model in the *.geo* format of the GMSH software [3].
If moreover *refs* ≠ 0, a dozen files concerning face *refs* are generated with names *intermed-refs-**, especially *intermed-refs-dc-cont-2d.mesh* which contains the discretization used as input to the planar mesher and *intermed-refs-frac_meshed.vector* which contains the numbers of the fractures connected to the fracture *refs*.

File *frac-no.vector* conditionally generated (output)

If *FRACwrite_frac* ≠ 0, a fracture network in *.disk* format (circles) is converted into *.vector* format (ellipses). These two formats are described above.

Files *inter-∗* conditionally generated (output)

If *FRACwrite_inter* ≠ 0, the software describes the fracture intersections in the *inter-no.vector*, *inter-vertices.dat* and *inter-edges.dat* files (see above). If *FRACwrite_inter* = 1, these are the intersections concerning all the faces (fractures and faces of the *cube*). If *FRACwrite_inter* = 2, intersections not

involving any face of the *cube*. Finally, if *FRACwrite_inter* = 3, intersections concerning at least one face of the *cube*.

Files of subdomains, conditionally generated (output)

If *FRACwrite_subdomain* ≠ 0, the local numbers of the subdomains are given, with several options: in a separate *subdomain-*.vector* file or not, in the *amadeus.mesh* file or not, for all the fractures and the 6 faces of the *cube*, or for the 6 faces of the *cube* only. These options are determined by the value of the parameter:

- 0 or 1 : no numbers given
- 2 : file *subdomain-cube*, without fractures
- 3 : file *subdomain-frac-cube*, with fractures
- 4 : file *amadeus,* without fractures
- 5 : file *amadeus*, with fractures
- 6 : file *subdomain-cube* and *amadeus*, without fractures
- 7 : file *subdomain-frac-cube* and *amadeus*, with fractures

3 Examples

In this section, several examples of meshes generated with MODFRAC are presented.

The geometry of the fracture networks presented in this section was provided by the DFN.lab software developed by Fractory, a joint laboratory between ITASCA Consultants SAS, the CNRS and the University of Rennes (https://fractorylab.org).

The simulations were run on an Intel(R) Core(TM) i7-7820HQ CPU @ 2.90GHz laptop with 4 cores (8 logical processors) and 32 GB of RAM.

❖ **Special fracture networks (sugar box type and Warren&Root)**

The first examples relate to particular networks of the sugar box and Warren&Root type. The *cube* is cut completely by an organized network of fractures whose length and width are significantly greater than the size of the *cube*.
The following example contains 15 fractures organized as a sugar box network. The *frac-no.vector* input file is given by Table 1.

No	Laxis_maj	Laxis_min	c_x	c_y	c_z	n_x	n_y	n_z	no_fracture	u_x	u_y	u_z	v_x	v_y	v_z
1	10	10	0	0	0	0	1	0	0	0	0	1	1	0	0
2	10	10	0	1	0	0	1	0	1	0	0	1	1	0	0
3	10	10	0	2	0	0	1	0	2	0	0	1	1	0	0
4	10	10	0	-1	0	0	1	0	3	0	0	1	1	0	0
5	10	10	0	-2	0	0	1	0	4	0	0	1	1	0	0
6	10	10	0	0	0	1	0	0	5	0	0	1	0	1	0
7	10	10	1	0	0	1	0	0	6	0	0	1	0	1	0
8	10	10	2	0	0	1	0	0	7	0	0	1	0	1	0
9	10	10	-1	0	0	1	0	0	8	0	0	1	0	1	0
10	10	10	-2	0	0	1	0	0	9	0	0	1	0	1	0
11	10	10	0	0	0	0	0	1	10	1	0	0	0	1	0
12	10	10	0	0	1	0	0	1	11	1	0	0	0	1	0
13	10	10	0	0	2	0	0	1	12	1	0	0	0	1	0
14	10	10	0	0	-1	0	0	1	13	1	0	0	0	1	0
15	10	10	0	0	-2	0	0	1	14	1	0	0	0	1	0

Table 1. Sugar box example: file frac-no.vector.

The geometry corresponding to this file is given by Figure 1.

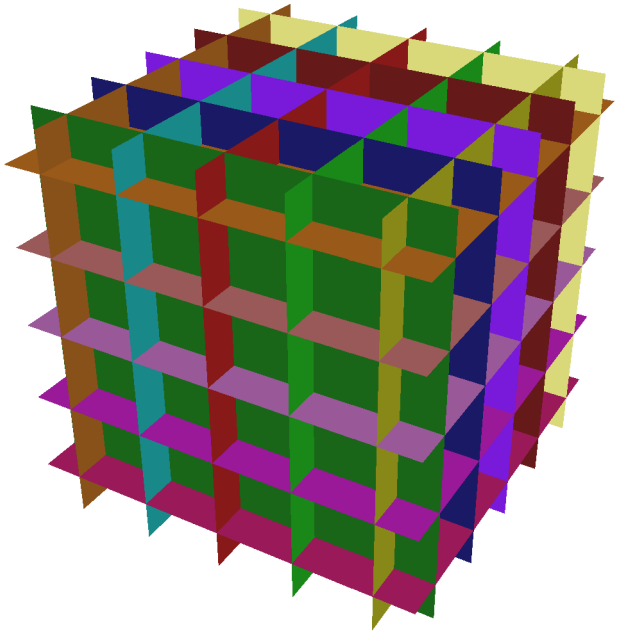

Figure 1. Sugar box geometry – 15 fractures.

The following *modfrac.env* file (Table 2) is used to generate the mesh shown in Figure 2. By default the mesh is conformal (*FRACconforming* at 1). This mesh was made by MODFRAC in 1s.

```
verb 10

FRACL 5
FRACcube_faces 12
FRACporous 0
FRACqmin 1.e-6
FRACwrite_frac 0

hgeo_flag 1
hgeomax 1000.
angle_mesh 30
```

Table 2. File modfrac.env.

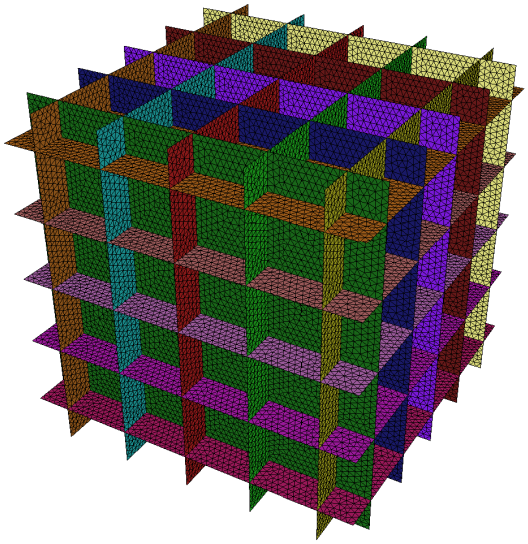

Figure 2. Sugar box mesh – 15 fractures.

The quality of the mesh is given by the file *quality-data.dat* (Table 3) and by the histogram of Figure 3.

111240 (#triangles)
0.789048 (qmin)
42055 (qmin:triangle)
25408 (qmin:v1)
25369 (qmin:v2)
25758 (qmin:v3)
6 (qmin:fracture_ref)
0.980217 (qmoy)
0.999999 (qmax)
80054 (qmax:triangle)
40272 (qmax:v1)
40324 (qmax:v2)
39768 (qmax:v3)
11 (qmax:fracture_ref)

Table 3. File quality-data.dat.

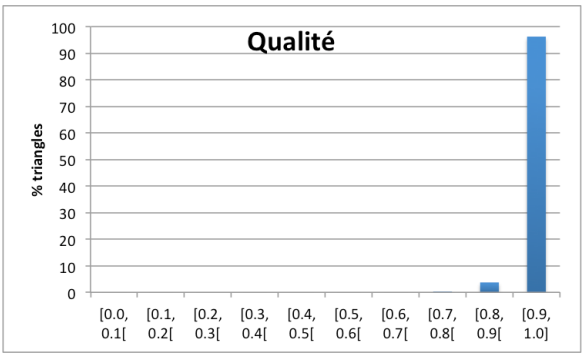

Figure 3. Mesh quality histogram.

From the *quality-data.dat* file (Table 3) are deduced properties of the generated mesh:

- It contains 111,240 triangles (#triangles),
- It has a minimum quality 0.789048 (qmin) at triangle 42055 (qmin:triangle) which has vertices 25408 (qmin:v1), 25369 (qmin:v2) and 25758 (qmin:v3) belonging to fracture 6 (qmin: fracture_ref). In this example, all fractures have been selected and meshed. Otherwise, the correspondence between the fracture_ref number and the original fracture number is in the *modfrac_frac_meshed.vector* file.
- It has an average quality of 0.980217 (qmoy),
- It has a maximum quality of 0.999999 (qmax) at triangle 80054 (qmax:triangle) which has as vertices 40272 (qmax:v1), 40324 (qmax:v2) and 39768 (qmax:v3) belonging to fracture 11 (qmax :fracture_ref),
- The mesh quality is good, no triangle quality being below 0.7.

Other specific geometries can be meshed, such as Warren&Root type networks for which the fractures are not necessarily regularly spaced, as shown in Figure 4.

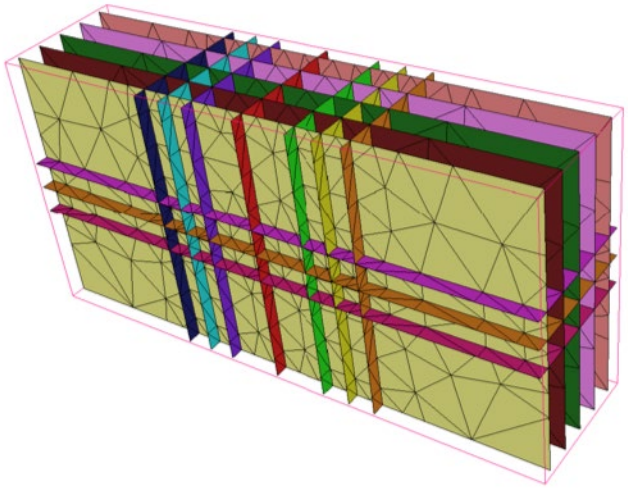

Figure 4. Mesh of a Warren&Root type network – 15 fractures.

❖ **Networks of fractures represented by ellipses**

Now, we consider Discrete Fracture Network (DFN) type fracture networks, for which the fractures are discs or ellipses which can cut the *cube* and/or end up completely immersed in the *cube*.

Let's start with small DFNs to illustrate the different options available as input to the *modfrac.env* file.

The network of fractures presented in Figure 5 is contained in a cube of side L20. It has 1397 fractures described in a *frac-no.vector* file.

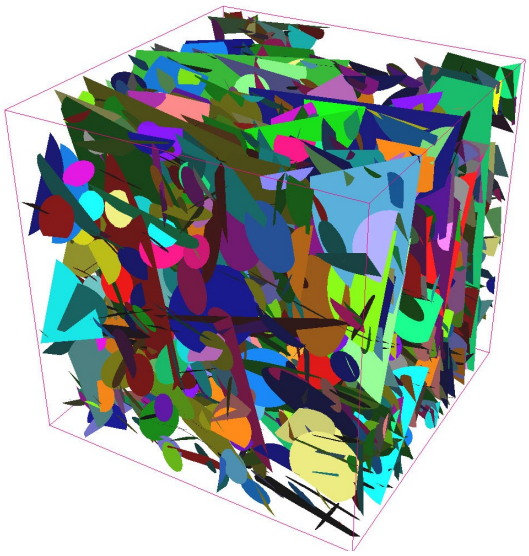

Figure 5. DFN L20 – 1397 fractures.

```
verb 10
FRACL 20
FRACcube_faces 12
FRACporous 0
FRACqmin 1.e-4
FRACwrite_frac 0
hgeo_flag 1
hgeomax 1000.
angle_mesh 30
number_of_threads 4
export_amadeus 1
```

Table 4. File modfrac.env .

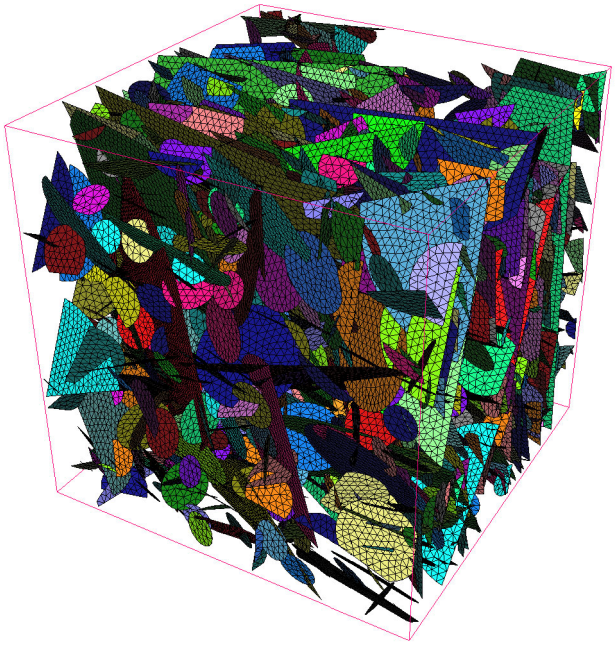

Figure 6. DFN L20 – Mesh – 1397 fractures.

The mesh generated from the *modfrac.env* file given in Table 4 is shown in Figure 6. The mesh contains 178,407 triangles and was completed in 17s.

The quality of the mesh is given by file *quality-data.dat* (Table 5) and the histogram of Figure 7. The quality is slightly lower than the previous one, due to the many geometric constraints forming angles close to 0° or 180°. Note that the minimum quality criterion ***FRACqmin*** imposed in the *modfrac.env* file is satisfied: 0.000565164 (qmin) > 0.0001 (***FRACqmin***).

```
178407 (#triangles)
0.000565164 (qmin)
67234 (qmin:triangle)
15628 (qmin:v1)
3337 (qmin:v2)
620 (qmin:v3)
362 (qmin:fracture_ref)
0.945222 (qmoy)
1 (qmax)
74423 (qmax:triangle)
15251 (qmax:v1)
15252 (qmax:v2)
64257 (qmax:v3)
421 (qmax:fracture_ref)
```

Table 5. File quality-data.dat.

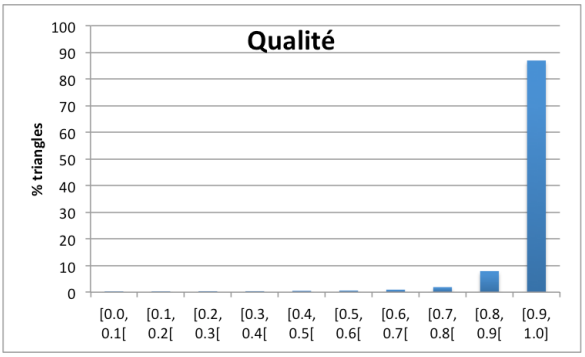

Figure 7. Mesh quality histogram.

Thanks to the *gradation* and *angle_mesh* parameters, different levels of refinement can be specified to MODFRAC, as shown in Figures 8 and 9.

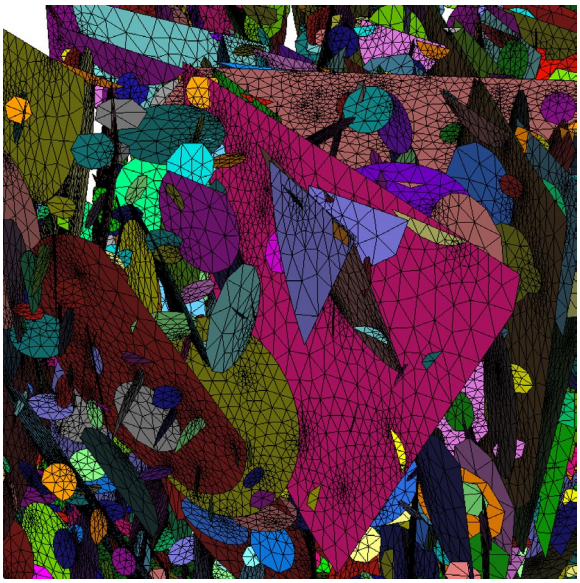

Figure 8. Angle 30° and gradation 1.4.

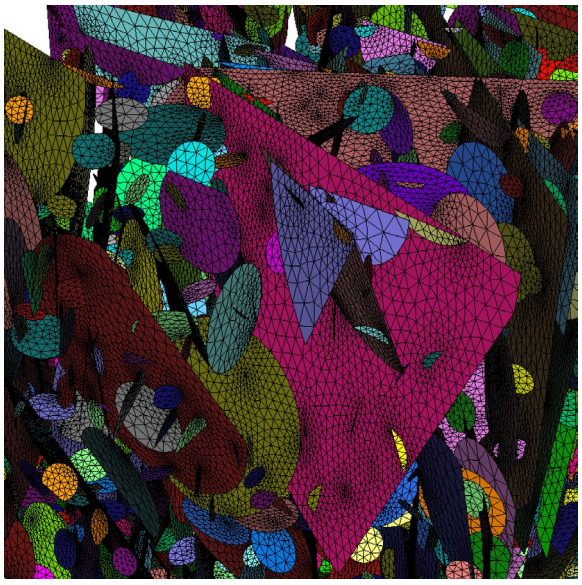

Figure 9. Angle 10° and gradation 1.2.

MODFRAC is able to mesh much larger fracture networks. For example the network shown in Figure 10 contains 1,176,566 fractures. The mesh made by MODFRAC and generated from the *modfrac.env* file given in Table 6 contains 20,723,302 triangles and was generated in 14 minutes with 4 threads.

```
verb 10
FRACL 200
FRACcube_faces 12
FRACporous 0
FRACqmin 1.e-4
FRACwrite_frac 0
hgeo_flag 1
hgeomax 1000.
angle_mesh 30
number_of_threads 4
export_amadeus 1
memory 800000
```

Table 6. File modfrac.env.

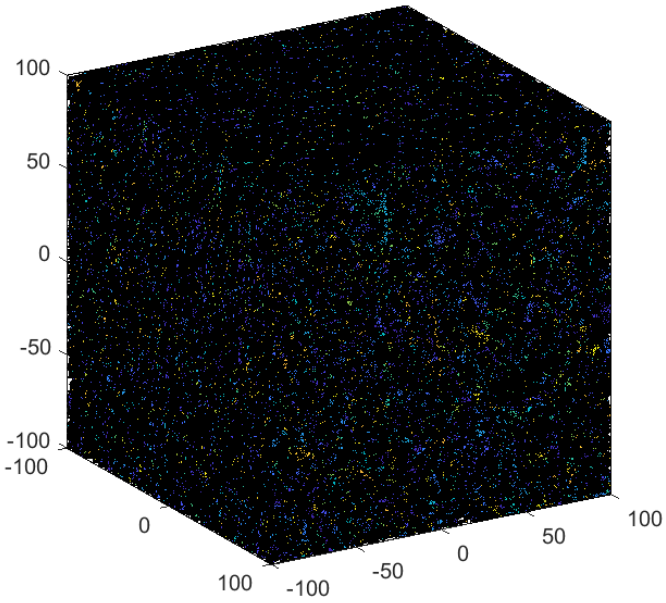

Figure 10. DFN L200 – Mesh – 1,176,566 fractures.

The quality of the mesh is given by the *quality-data.dat* files of Table 7 and by the histogram of Figure 11.

```
20723302 (#triangles)
4.81633e-05 (qmin)
8229123 (qmin:triangle)
3143093 (qmin:v1)
3143092 (qmin:v2)
10401401 (qmin:v3)
327098 (qmin:fracture_ref)
0.616738 (qmoy)
1 (qmax)
19269394 (qmax:triangle)
9065201 (qmax:v1)
10872669 (qmax:v2)
9065200 (qmax:v3)
1067726 (qmax:fracture_ref)
```

Table 7. File quality-data.dat.

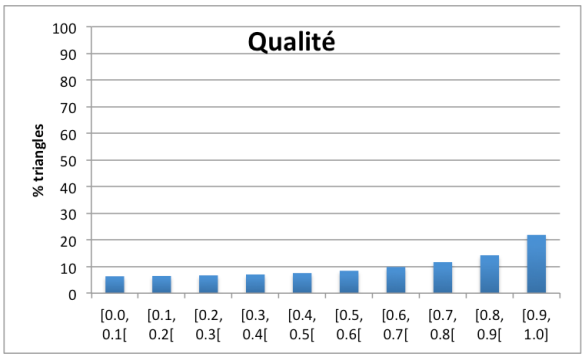

Figure 11. Mesh quality histogram.

For such a complex network, the mesher is able to improve the minimum quality of the triangles, even if it remains a little below the required quality (4.8e-05 < ***FRACqmin***). The screen output indicates that this happens for 3 triangles.

❖ **Networks of fractures represented by polygons**

By using the ***FRACell_to_pol*** parameter, it is possible to transform ellipses into polygons before meshing them. Warning! This can change the fracture network considered depending on the ***FRACell_to_pol*** number of sides chosen. For example, on the L20 Figure 5 test case, by adding the ***FRACell_to_pol*** parameter to 10 to the *modfrac.env* file of Table 4, we obtain the Figure 12 mesh which contains 1311 polygons (while the initial network contained 1397 fractures). Indeed, an ellipse can intersect the *cube* while the polygon which discretizes it is entirely exterior to the *cube* and two ellipses can intersect between them while the corresponding polygons do not intersect.

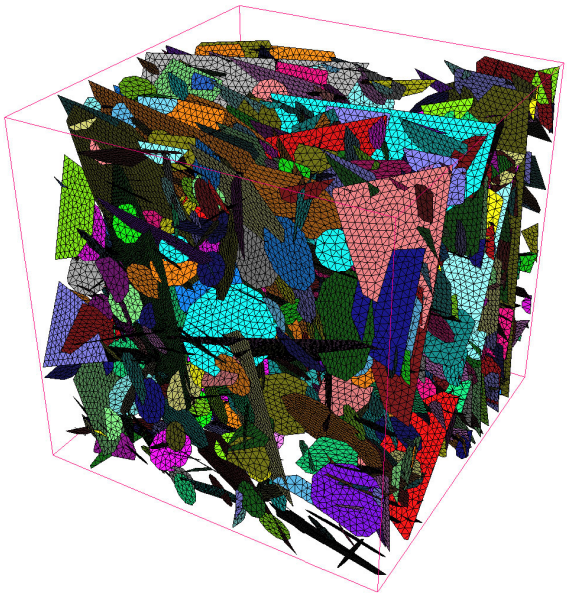

Figure 12. DFN L20 – ellipses are replaced by 10-sided polygons – 1311 polygons.

The quality is given by Table 8 and by the histogram of Figure 13.

132824 (#triangles)
0.000498204 (qmin)
4090 (qmin:triangle)
427 (qmin:v1)
17855 (qmin:v2)
430 (qmin:v3)
29 (qmin:fracture_ref)
0.941964 (qmoy)
1 (qmax)
68418 (qmax:triangle)
18139 (qmax:v1)
18140 (qmax:v2)
56964 (qmax:v3)
534 (qmax:fracture_ref)

Table 8. Fichier quality-data.dat.

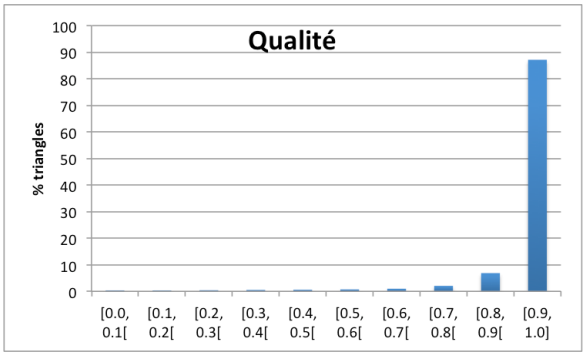

Figure 13. Histogramme de qualité du maillage.

MODRAC is also able to take a ***.geo*** file as input. In this format of the GMSH software [3], each fracture is delimited by a convex planar polygon.

❖ **Addition of wells**

A simple example of *wells.geo* file is given below (Table 9).

```
Point(1) = {0.5, 0, -3, h};
Point(2) = {0.353553, 0.353553, -3, h};
Point(3) = {0, 0.5, -3, h};
Point(4) = {-0.353553, 0.353553, -3, h};
Point(5) = {-0.5, 0, -3, h};
Point(6) = {-0.353553, -0.353553, -3, h};
Point(7) = {0, -0.5, -3, h};
Point(8) = {0.353553, -0.353553, -3, h};
Point(9) = {0.5, 0, 5, h};
Point(10) = {0.353553, 0.353553, 5, h};
Point(11) = {0, 0.5, 5, h};
Point(12) = {-0.353553, 0.353553, 5, h};
Point(13) = {-0.5, 0, 5, h};
Point(14) = {-0.353553, -0.353553, 5, h};
Point(15) = {0, -0.5, 5, h};
Point(16) = {0.353553, -0.353553, 5, h};

Line(1) = {1, 2};
Line(2) = {2, 3};
Line(3) = {3, 4};
Line(4) = {4, 5};
Line(5) = {5, 6};
Line(6) = {6, 7};
Line(7) = {7, 8};
Line(8) = {8, 1};
Line(9) = {9, 10};
Line(10) = {10, 11};
Line(11) = {11, 12};
Line(12) = {12, 13};
Line(13) = {13, 14};
Line(14) = {14, 15};
Line(15) = {15, 16};
Line(16) = {16, 9};
Line(17) = {1, 9};
Line(18) = {2, 10};
Line(19) = {3, 11};
Line(20) = {4, 12};
Line(21) = {5, 13};
Line(22) = {6, 14};
Line(23) = {7, 15};
Line(24) = {8, 16};

Line Loop(1) = {-8, -7, -6, -5, -4, -3, -2, -1};
Line Loop(2) = {9, 10, 11, 12, 13, 14, 15, 16};
Line Loop(3) = {1, 18, -9, -17};
Line Loop(4) = {2, 19, -10, -18};
Line Loop(5) = {3, 20, -11, -19};
Line Loop(6) = {4, 21, -12, -20};
Line Loop(7) = {5, 22, -13, -21};
Line Loop(8) = {6, 23, -14, -22};
Line Loop(9) = {7, 24, -15, -23};
Line Loop(10) = {8, 17, -16, -24};

Plane Surface(1) = {1};
Plane Surface(2) = {2};
Plane Surface(3) = {3};
Plane Surface(4) = {4};
Plane Surface(5) = {5};
Plane Surface(6) = {6};
Plane Surface(7) = {7};
Plane Surface(8) = {8};
Plane Surface(9) = {9};
Plane Surface(10) = {10};
```

Table 9. File wells.geo.

The previous file describes a vertical well going from surface $z = 5$ to depth $z = -3$, the global domain being defined in the environment file *modfrac.env* as a cube centered at the origin and of side 10. The cross section of this well is approximated by an octagon, also centered at the origin. The surface mesh created by MODFRAC, as well as the volume mesh coming from GH3D, are illustrated in Figure 14.

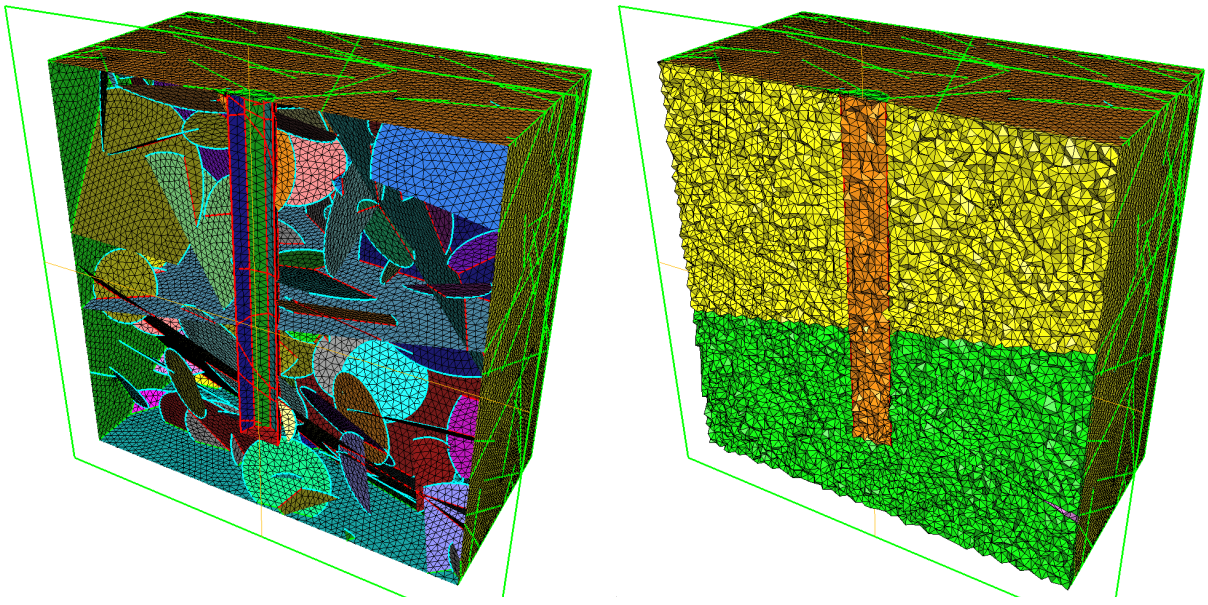

Figure 14. On the left, section of a surface mesh containing a well. On the right, section of the corresponding solid mesh.

❖ **Fractured porous mesh**

The ***FRACporous*** option at 1 also meshes the faces of the *cube*. The output of MODFRAC can then be proposed as input to a volume mesher. Figure 10 gives an example of a volume mesh, produced with the GHS3D software (Inria), from the surface mesh generated with MODFRAC.

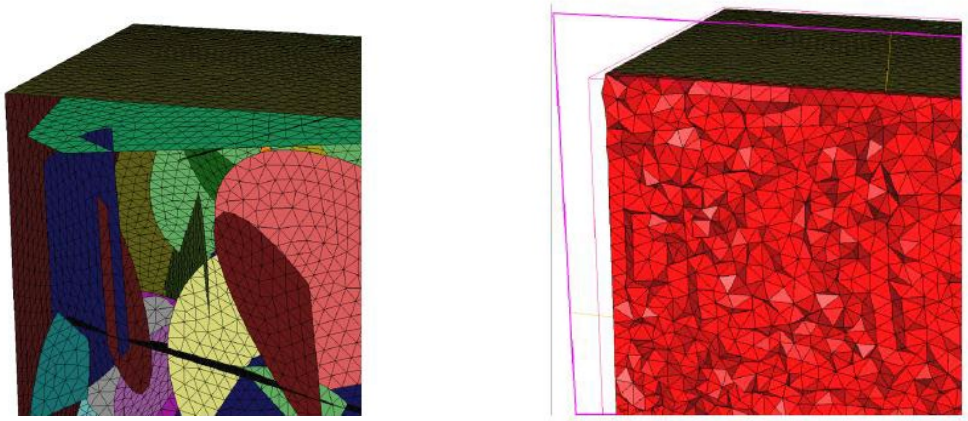

Figure 15. Example of a volume mesh (right image) produced with GHS3D (Inria) from the surface mesh generated with MODFRAC (left image).

As an example, in the article [7], a network of 52 polygons is proposed. Figure 16 illustrates the test case.

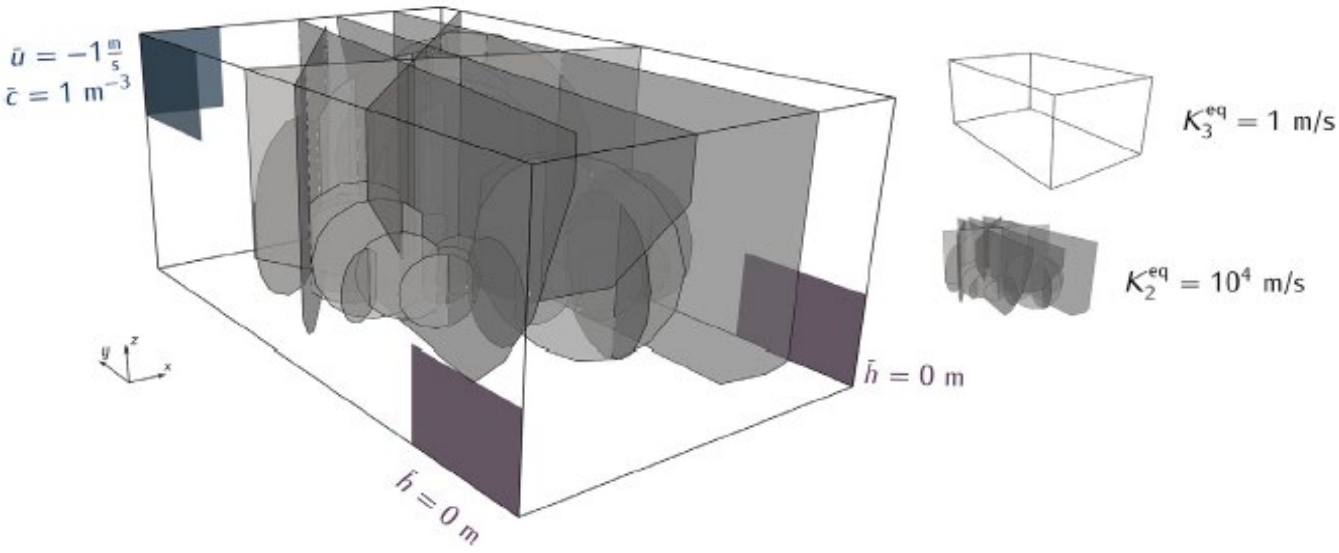

Figure 16. An example of a fractured porous test case [7] – 52 fractures represented by polygons.

The data is available at the following link: https://git.iws.uni-stuttgart.de/benchmarks/fracture-flow-3d.git.

By using the modfrac.env file with ***FRACporous*** at 1 and ***FRACfixed_constraints*** at 1 as proposed in Table 11 and the *constraints.dat* file proposed in Table 12 to define the shaded areas in Figure 16, MODFRAC can generate the mesh shown in Figure 17.

```
verb 10

FRACcubemin -500 100 -100
FRACcubemax 350 1500 500
FRACcube_faces 12
FRACporous 1
FRACqmin 1.e-4
FRACwrite_frac 0
FRACfixed_constraints 1
FRACwrite_subdomain 2

hgeo_flag 1
hgeomax 1000.
angle_mesh 30

number_of_threads 4
export_amadeus 1
size_computation 1
memory 800000
```

Table 11. File modfrac.env.

```
# d Omega in 0
1       -200.0 1500.0 500.0     -200.0 1500.0 300.0
1       -200.0 1500.0 300.0     -500.0 1500.0 300.0
# d Omega in 1
1       -500.0 1500.0 300.0     -500.0 1200.0 300.0
1       -500.0 1200.0 300.0     -500.0 1200.0 500.0
# d Omega out 0
1       -500.0 100.0 100.0      -500.0 400.0 100.0
1       -500.0 400.0 100.0      -500.0 400.0 -100.0
# d Omega out 1
1       350.0 100.0 100.0       350.0 400.0 100.0
1       350.0 400.0 100.0       350.0 400.0 -100.0
```

Table 12. File constraints.dat.

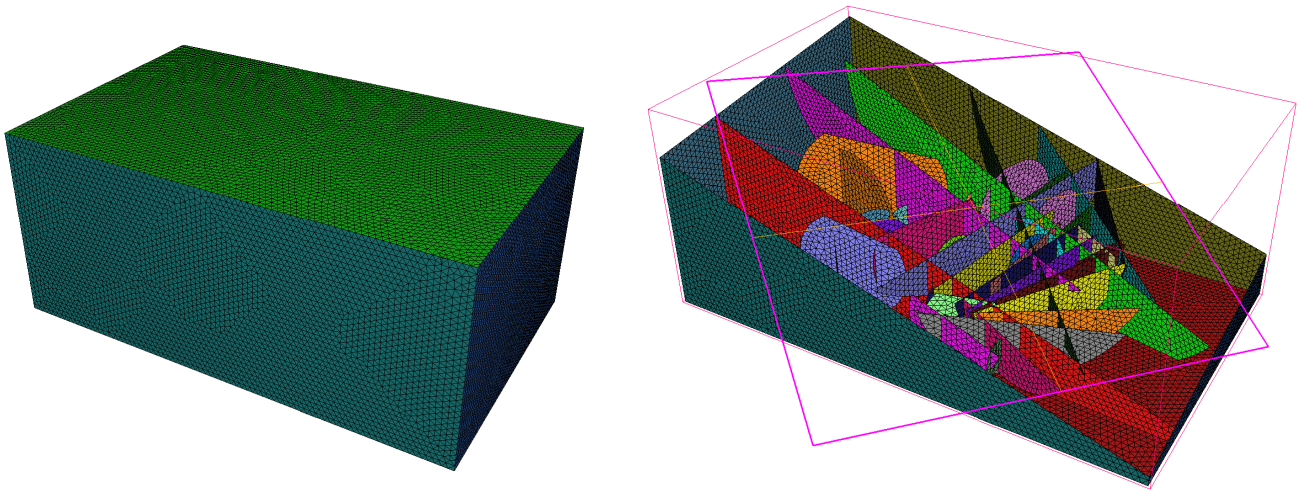

Figure 17. Fractured porous test cases [7] – 52 fractures. Surface mesh: overview on the left and section on the right.

This mesh can then be proposed as input to a volume mesher, such as GHS3D (Inria) (see Figure 18).

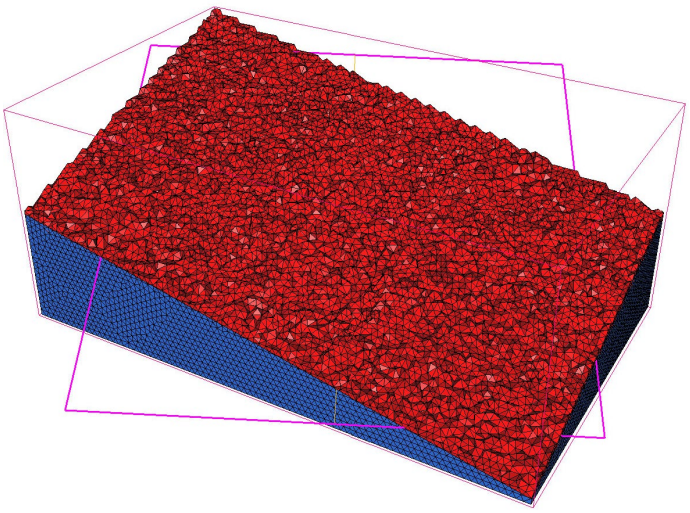

Figure 18. Fractured porous test case [7] – 52 fractures. Volume mesh generated by the software GHS3D (Inria).

These meshes can be used to perform numerical simulations. By way of example, Figure 19 shows the hydraulic head solution obtained by solving a stationary single-phase flow within fractured porous media [7] using a mixed hybrid finite element method.

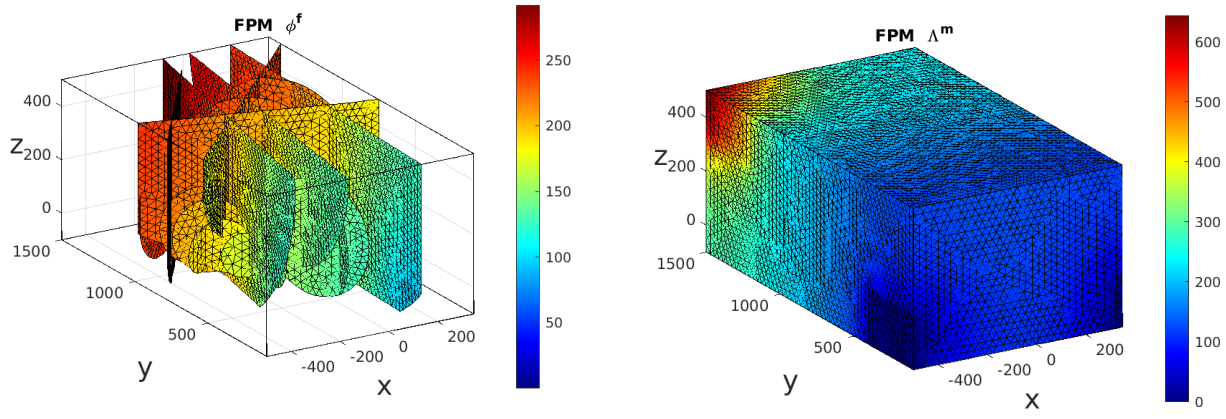

Figure 19. Fractured porous test cases [7] – 52 fractures. Hydraulic head obtained by solving the problem of flow in fractured porous media [7] with a hybrid mixed finite element method (RT0) implemented in the software nef-flow-fpm (Inria).

The following example shows a network containing 87,335 fractures. Figure 20 shows the mesh generated with MODFRAC. It contains 3,929,522 triangles and was made in 1 minute 37 seconds with 4 threads.

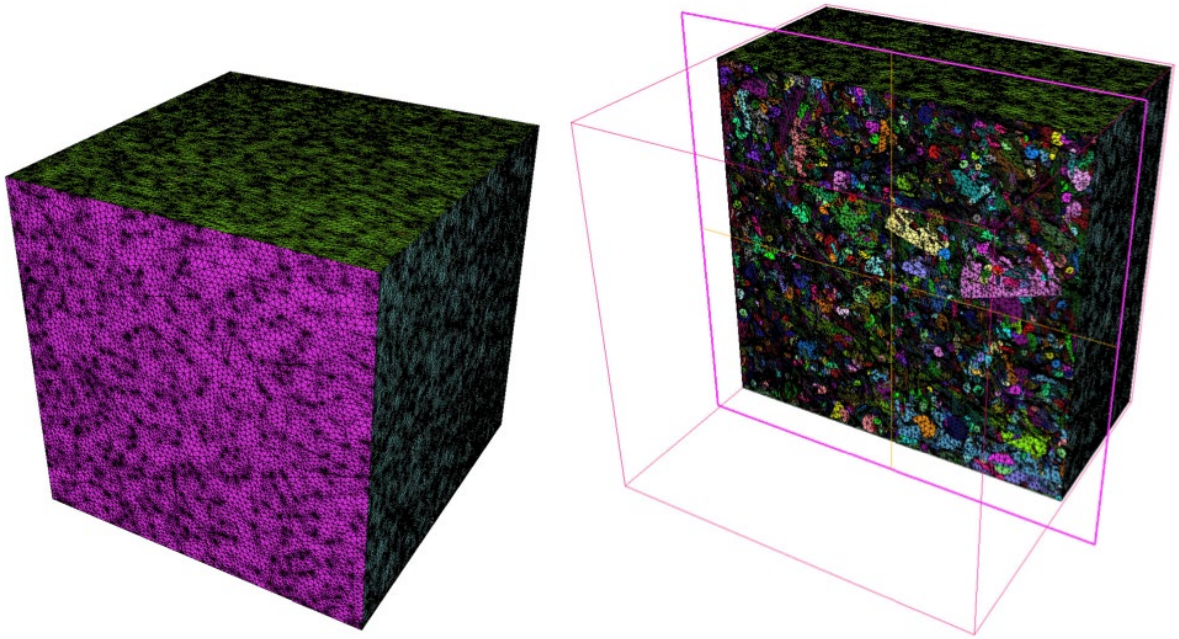

Figure 20. Fractured porous test cases – 87,335 fractures. Surface mesh: overview on the left and section on the right.

This mesh can then be proposed as input to a volume mesher, such as GHS3D (Inria) (see Figure 21).

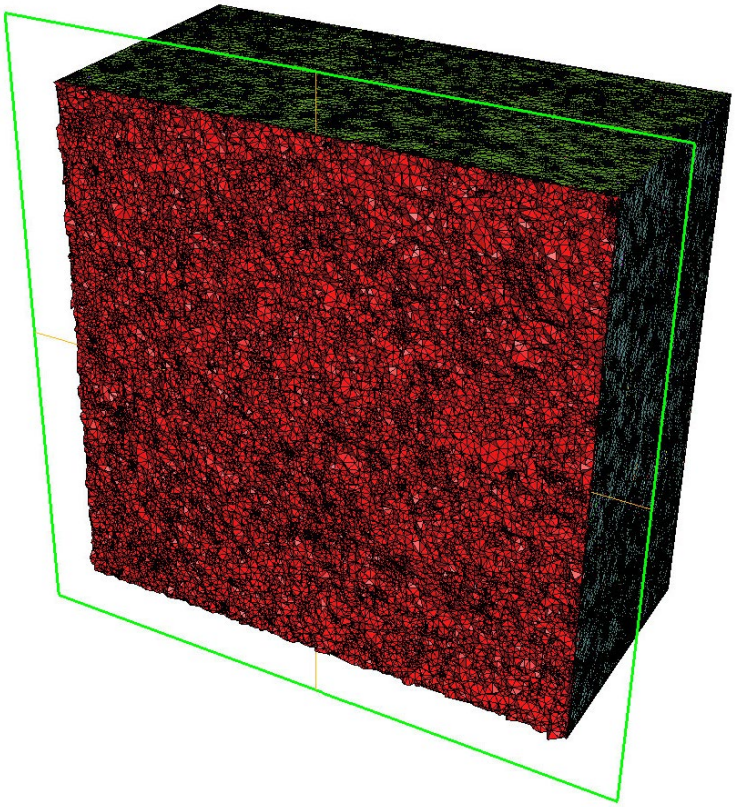

Figure 21. Fractured porous test cases – 87,335 fractures. Volume mesh generated by the software GHS3D (Inria).

3 Future work

The MODFRAC software now meets the essential demands concerning the modeling and meshing of fractures. However, these demands are constantly evolving, which implies new work to be carried out. This section describes the main ongoing studies.

Triangulated surfaces. Currently, the objects processed by MODFRAC (fractures, wells, *cube*) are defined by ellipses or planar polygons. It is planned to add triangulated faces to it in order to simply represent any warped surface. A first advantage concerns the *cube*, which could thus be generalized to a domain Ω of any shape. In addition, it would be possible to represent with better precision wells of cylindrical section and also following any path, as permitted by directional drilling techniques. Finally, the fractures themselves could be of arbitrary shapes. To integrate this new representation of surfaces, it is necessary to calculate the intersection lines drawn on each surface (triangle/triangle intersection in 3D), then transfer them to an isometric unfolding of each surface (which will be an associated parametric domain). Then, these lines must be made conforming in each parametric domain (edge/edge intersection in 2D). Once this representation is established, we will have a valid parametric domain and a discrete parameterization of each surface, and thus generate any desired mesh.

Domain decomposition. Another study concerns the decomposition of a parallelepipedic domain Ω into several blocks. This will allow additional parallelization of the calculations, resulting in new time savings and above all the processing of an even larger number of fractures, of the order of several tens of millions.

References

[1] J.R. de Dreuzy, Géraldine Pichot, B. Poirriez d, J. Erhel. Synthetic benchmark for modeling flow in 3D fractured media. Computers & Geosciences. 50 (2013) 59-71.

[2] R. Le Goc, File format *.disk*, Fractory internal report, https://fractorylab.org.

[3] Ch. Geuzaine, J.F. Remacle, Gmsh Reference Manual, https://gmsh.info/doc/texinfo/gmsh.pdf.

[4] P. Frey, MEDIT: An interactive mesh visualization software, RT-0253, Inria, 2001.

[5] Inria-Simulog, GHS3D, tetrahedral mesh generator, internal report, 2005.

[6] MATLAB, data analysis, algorithm development and creation of mathematical models, https://www.mathworks.com/.

[7] I. Berre, W. M. Boon, B. Flemisch, A. Fumagalli, D. Gläser, E. Keilegavlen, A. Scotti, I. Stefansson, A. Tatomir, K. Brenner, S. Burbullah, P. Devloo, O. Duran, M. Favino, J. Hennicker, I. Lee, K. Lipnikov, R. Masson, K. Mosthaf, M. G. C. Nestola, C. Ni, K. Nikitin, P. Schälde, D. Svyatskiy, R. Yanbarisov, P. Zulian. Verification benchmarks for single-phase flow in three-dimensional fractured porous media. Advances in Water Resources, Volume 147. 2021.