VORO2MESH Manual

Version 1.7

A MESH generator for the IFD Method

S. Bonduà, V. Bortolotti, C. Cormio, E.M. Vasini

DICAM – University of Bologna

Summary

1 Introduction

In the numerical modelling approach, the space domain and time are discretized. Depending on the scheme adopted, the discretization have some requirements/constraints.

For the Integral Finite Difference (IFD) method, it is requested that the conjunction line between two connected centroids is orthogonal to the interface separating the volume between two adjacent blocks. In the case of structured grid (Berry et al., 2014; Cormio et al., 2012), the orthogonal condition constraint is always implicitly satisfied.

The orthogonal condition play a fundamental role during the generation of the grid. Unstructured grid are natural candidate to take full advantages of the IFD method scheme, but are not easy to handle. The Voronoi approach is a robust method of discretization, ensuring the orthonormal constraint.

With the Voronoi method, given any set of points (even in a regular path), a discretization of the space can be always obtained, resulting in a grid that satisfy the orthogonal constraint. Several algorithms was developed for the 2D case and implemented in open source codes. A few algorithms are implemented for the 3D case and it is worth to mention Qhull (Barber et al. 1996), TetGen (Hang Si, 2015), and voro++ (Rycroft, 2009).

2 TOUGH2 & VORO++

TOUGH2 (Pruess et al., 1999) is a widely used numerical simulator for geothermal reservoir engineering, nuclear waste disposal, environmental assessment and remediation, underground gas storage and hydrology. It is a free and open source software (distributed by the Lawrence Berkeley National Laboratory) structured in in FORTRAN modules. Using dedicated modules for the management of thermodynamics and thermophysics properties (called Equation of State, EOS) TOUGH2 is very flexible and can be easily adapted to specific simulation requirements. It uses IFDM and therefore can manage one, two and three dimensional structured and unstructured spatial discretization.

In particular, TOUGH2 require block information about volume and connection with adjacent cells interface area. So, given set of points in the 3D space, that represent the blocks nodes, we can generate the 3D Voronoi tessellation and then calculate all the values to fill the ELEME and CONNE data (ref. TOUGH2 manual).

Regarding the full 3D Voronoi tessellation computation, a few algorithms have been developed by the scientific community, lioke Qhull (Barber et al. 1996), TetGen (Hang Si, 2015) and voro++ (Rycroft, 2009).

Our approach adopt the voro++ libraries that distributed under a modified BSD license, that makes it free for any purpose.

3 VORO2MESH

The VORO2MESH is a software developed at DICAM (University of Bologna) able to calculate a 3D Voronoi tessellation of a set of points, generating a ready to use MESH file for TOUGH2. The mesh file of TOUGH2 contains the ELEME and CONNE information as specify in the TOUGH2user's manual. Several options for the grid generation are available. The VORO2MESH computation parameters and options must be stored in a file called **voro.par**. In particular, **voro.par** must contain all the parameters for the discretization. In Table 1 is shown the **voro.par** structure and keywords description. Everything that follows the comment symbol is "!"is ignored by the parser. For a complete list of the available keywords, see Appendix F – Complete list of VORO2MESH input keywords.

!VORO2MESH parameter file

x_max=+1726500.0000 !x max
x_min=+1701500.0000 !x min
y_max=+4759500.000 !y min
y_min=+4732500.000 !y max
z_max=2000.0 ! z max
z_min=-6000.0 ! z min
toler=1.001 ! all connection with area<toler will be skipped from CONNE
toler_dist2=1.0E-06 !minimum square distance between two points. If d(p1,p2)<toler_dist2 the program will terminate.
read_rocktype=1 !0=no,1=yes;
auto_nxnynz=1 !0=manual;1=auto. Parameter for compute the voronoi tessellation
print_vtkXML_file=1 !0=no;1=yes; The voronoi geometry will be exported to a VTK file in a vtu format (see Paraview documentation);
n_x=10 ! number of domain subdivision for voronoi computation
n_y=10 ! number of domain subdivision for voronoi computation
n_z=1 ! number of domain subdivision for voronoi computation
r_max=2.0 !
wall_type=1 !1: regular box;2=cylinder
add_walls=0 !0:no; 1:yes. If yes, a file called "wallslist.dat" must be present. Inside, a list of ABCD have to be present, that are parameter of the equation Ax+By+Cz+D>0 (the cutting wall)
tolerance_walls=0.1 !parameter for evaluate inside points if(ax+by+cz-d>tolerance_walls) then ok.
min_distance_from_walls=10.0 !if dist(point,walls(i))<min_distance_from_walls the node is skipped
fit_surface=3 !0:no fit; 1=execute fit ;2:fitsuface2;3:fitsurface3. if fit_surface=3, a file list called surface list must be present.
n_layers=5 !
refine_mesh=1 !if(refine_mesh=1)norefinement;if refinemesh>1, each square is then divided in refine_mesh^2 elements. Not implemented.
coarse_mesh=1 !if(coarse_mesh=1)no coarsening;if coarse_mesh>1, a square is then taken skipping coarse_mesh elements. Note that refinemesh=2 and coarse_mesh=2 give the same number of elements, but generate different meshes. Not implemented.
vertical=2 !0: the point belongs to the segment;1:the line is vertical centered;2: the line is vertical, the node have z as multiple of 2*offset(semi regular grid).use with
variable_n_layers=1 !0=no; 1=for each xy, calculate n_layer=int(distance/offset);
offset=50.0 !if offset<0.0 then offset=min(dx,dy)
blocks_thick=70.0 !general value for blocks height;if blocks_thick <0 then blocks_thick=offset.
cut_model_top_bottom=1 !0=no; 1=yes; Node outbound from the top/bottom surface are used to compute voronoi tessellation but are skipped from the MESH file.
assign_infinite_volume_to_boundary=0 !0=no;1=yes.
create_incon=0 !:0=no; 1=yes
read_por_perm_tables=0 ! 0=no; 1=yes
format_por_perm_tables=1 ! 1=x,y,z,por,k,[ky],[kz]; 2=gridded files(one file for each values). 2=NOT IMPLEMENTED.
number_of_points_por_perm=1 !number of points used to calculate por and perm. Actually use 1 or 4 ONLY.
dist_mode_por_perm=0 ! 0=2D; 1=3D;
max_dist_por_perm=850.0 !
create_gener=0 !:0:no; 1=yes; 2=use mask.dat
min_2d_dist=850.0 ! when creating GENER, if we have two block with d(P1,P2)<min_2d_dist, only the lower block is taken...(feature for ENI project)
assign_roktype_to_gener=0 ! 0 no; >0 rocktype=assign_rocktype_to_gener
divide_control_inside_points=1 !0=no;1=yes;

por_perm_upscaling=1 !1=mean;2=series and parallel calculation (use 2 only with number_of_points_por_perm=4) exclude_out_pts=1 !0=no;1=yes; points not inside the domain will not print in "in.dat.ready". During reading, outside points are skipped in any case. two_digit=1 !0=use 3 digit exponent(example 1.403E+003);1=use two digit exponent (example: 1.4031E+03) blocks_thick=50.0 !general value for blocks height debug_mode=0 !0=no;1=yes write_tough2viewer_dat=1 !0:no; 1:yes block_names_format=0 !default=0;=0,standard A3I[len_char_eleme_name-3];=1 use I[len_char_eleme_name] len_char_eleme_name=5 !default 5; must be 5<=len_char_eleme_name<=9 len_char_volume=10 ! len_char_eleme_surface=10 ! len_char_coordinates=10 ! len_char_d1d2=10 ! len_char_surface=10 ! len_char_cosine=10 ! CVT=0 !If CVT=1, the internal points are moved to have CVT CVT_max_iter=50 !maximum number of iterations k_s=0.0 ! k_cvt=0.01 ! HLBFGS_FLAG=0 ! end=end
Table 1 – the **voro.par** file structure and keywords description.

4 Input data files

The VORO2MESH input is at least composed of two files: the parameters and the data files. The former contains (**voro.par**, see table 1 for an example), as requested by the voro++ library, geometric parameters about the 3D domain size, domain cutting planes, and directives about the seed points to be used. The latter file can be of two types. The first one, named **voro.par**, contains just the list of the nodes (see table 2): id (a unique key); (x, y, z, coordinates); rock type (optional, for rock type assignment. This option is activated setting the keyword read_rocktype=1). Note that if the rock type is set to a negative value, then the node will be used during the tessellation computation, but the corresponding block will not be included in the MESH file and also will not visualized by the viewer. Each outbound point (external to the domain defined by the cutting planes) is skipped from further calculation

The second type (**SurfaceFiles.dat**) contains the list of the names of the surface files used to discretize the domain as defined in the parameters file. In addition, a supplementary file containing the ordered list (from top to bottom) of the necessary geological boundary surfaces could be present. Surface files must be stored, in the format specified in **Errore. L'origine riferimento non è stata trovata.**, in the same folder from which the input files are loaded. Optionally the domain can be cut with planes expressed in the implicit form Ax+By+Cz+D=0, and the list of planes parameters A, B, C, D must be present in a specific file named **WallList.dat** file.

Header – in.dat file format 0 72.594696 927.405334 -279.704498 1 1 27.405306 972.594666 -202.790009 2 2 164.947006 925.829895 -234.842972 1 3 135.052994 974.170105 -152.565567 2 4 250.021210 924.671692 -218.623108 1

5 249.978790 975.328308 -132.403046 2
6 335.082825 925.825134 -234.762024 1
7 364.917175 974.174866 -152.468536 2
…
(comment line)

5 Table 2 – The in.dat file. See Appendix A – Parameter File voro.par

The **voro.par** contains directive of working mode (point/surfaces), bounding domain, output options of VORO2MESH. Keywords are briefly resumed in the given example file.

```
##################################
# voro.par example parameter file#
##################################
#
# Domain definition
x_max=+192200.00
x_min=+16900.0000
y_max=+802800.00
y_min=+652800.000
z_max=-1000.00
z_min=-8000.0# Operating ways
fit_surface=1 #0:no fit, use in.dat as input file; 1=execute fit: the
seed point will be stored in the in.dat.ready file.
# MESH generation parameters
toler=1.001 #in the CONNE section, if area interface between adjacent
# blocks < toler, then connection will be discarded.
read_rocktype=1 #0:no,1=yes
block_names_format=0 #default=0, standard A3I[len_char_eleme_name-3];=1
use I[len_char_eleme_name]
len_char_eleme_name=5 #default 5; must be: 5<=len_char_eleme_name<=9
len_char_volume=10 #
len_char_eleme_surface=10 #
len_char_coordinates=10 #
len_char_d1d2=10 #
len_char_surface=10 #
len_char_cosine=10 #
#voro++ parameters
auto_nxnynz=1 #0=manual;1=auto, then n_x,n_y, n_z are not used.
n_x=7 # number of subdivision of the domain in x
n_y=7 # number of subdivision of the domain in y
n_z=7 # number of subdivision of the domain in z
# printing options
write_tough2viewer_dat=1 #0:no; 1:yes
print_vtkXML_file=1 #0=no;1=yes.
export_surface_as_ply=0 #0=no;1=yes: the input surface are exported as
ply files.
end=end #end of parameter file
```

Appendix B – input points file format for details.

5.1 Walls

By default, the VORO2MESH programs discretize the space in a box, defined by xmin, ymin, zmin, xmax, ymax, zmax. The voro++ library allows to consider a set of planes that define the convex volume of interest.

VORO2MESH use walls plane by reading from a file a list of planes. Each plane must be defined by the four parameters values (in meters) of the implicit equation of a plane in the 3D space, Ax+By+Cz+D=0. To activate this option, the readWalls parameter must be set equal to 1 in the **voro.par** parameters file. If readWalls is set to 1, a file named wallsList.dat (see table 3) must be present in the same folder of the executable of VORO2MESH. Optionally, in the MESH file, a value of rocktype=BOUND can be set for the cells cut by the walls planes and a volume of 1.E+55 (boundary conditions) can be set for each cell that is defined to be BOUND. This option is activated by setting the variable assign_infinite_volume_to_boundary of the **voro.par** parameter file to 1.

-1700 -8500 0 -5634310000 0 -141100 0 -92138300000 1700 -1700 0 -786420000 3400 0 0 653140000 83300 3400 0 18239470000 6800 1700 0 2543540000 3400 1700 0 1907740000 ….
Table 3 – structure of the wallsList.dat file. Values of A, B, C and D are expressed in meters. No comment lines are admitted.

6 Output files

The output of a VORO2MESH run consists of several files:

-Tough2viewer.dat, which contains all the geometrical information to visualize the model with TOUGH2Viewer (Bonduà et al., 2012; Bonduà et al 2017). See Appendix E – Tough2Viewer.dat file format for details;

-MESH, a ready to use TOUGH2 MESH file. If the input data file contains the rock type, then the MESH is filled also with this information. If the data file is of surface type, a progressive rock type index is assigned to each blocks encompassed between two surfaces;

-statistic.dat, which contains statistical grid information (e.g.: minimum and maximum volume for each rock type, elements number, number of connections, minimum and maximum connection area, etc.). See Appendix F – Statistic.log file format;

-Optional grid file in vtu format (ParaView VTK Unstructured Data) for visualization with ParaView;

-Optionally, a ready to use TOUGH2 INCON file. The value of each primary variable used by the simulator can optionally be computed as a linear combination of the coordinates of the node and specific primary variable (as for example a temperature gradient) defined and inputted by the user.

-Optionally, a set of surface ply file (Bourke, 2011), converted from the GRASS format. Useful for visualization of the input surface and the resulting grid at the same time.

7 Examples

The following examples are obtained using as input data a file containing the nodes in the [ID, x, y, z, rocktype] format. The points are generate with a tool not included with VORO2MESH. The visualization is obtained by means of Tough2Viewer (Bonduá et al., 2012).

7.1 Example 1: radial mesh in a rectangular domain

The radial tessellation is obtained inside a regular box.

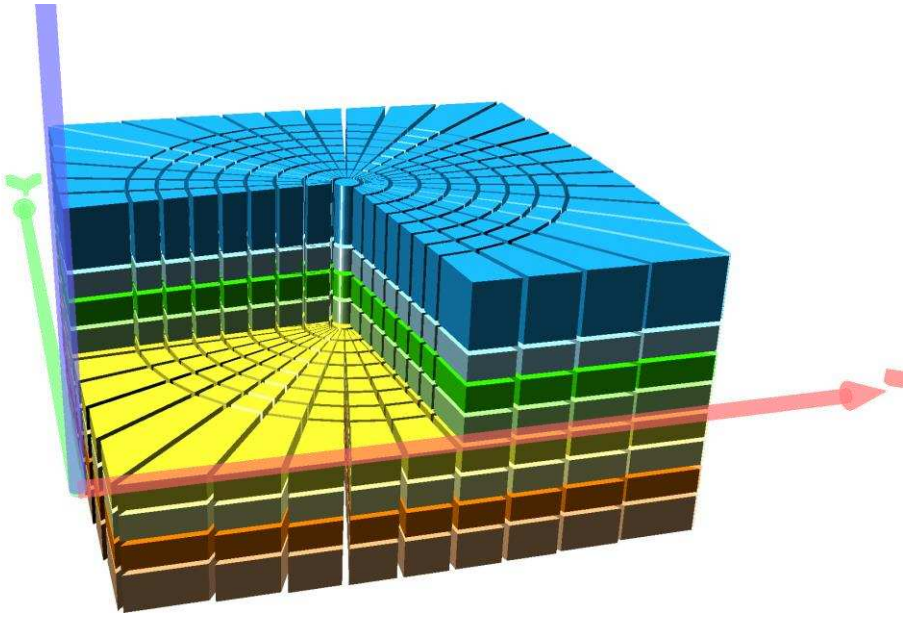


Figure 1 – A 3D view of a radial mesh in a rectangular domain. Each stratum (layer) has a different rocktype.

7.2 Example 2: radial mesh with different rocktype for boundary blocks.

In this example, the boundary blocks rocktype is set to a specific value, so that it is possible to remove they from visualization.

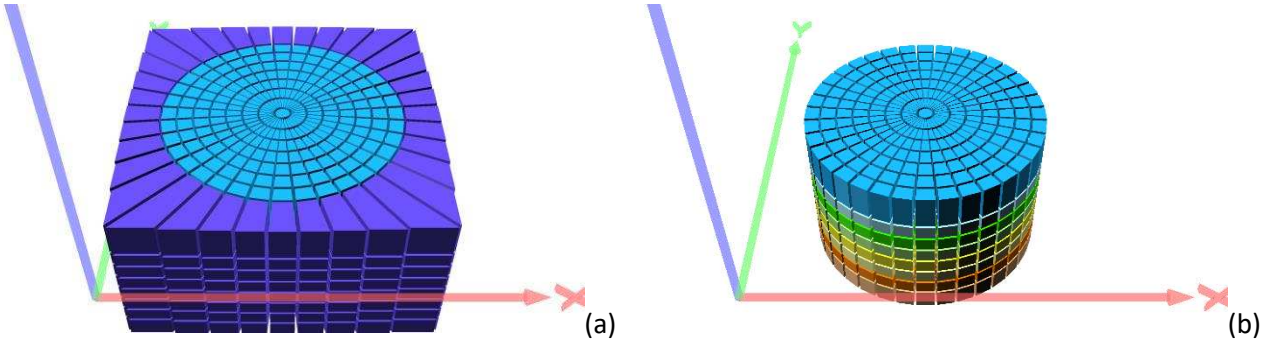


Figure 2 – (a) a 3D view of the model. Each stratum has a different rocktype. Boundary blocks are marked with a a specific rocktype.(b) boundary blocks are removed from visualization.

7.3 Example 3: Radial grid with cutting planes

Similar to the previous one, the tessellation is obtained adding a set of cutting planes (walls, 180 cutting planes to approximate the cylinder lateral shape). The boundary blocks have the lateral vertical face composed of a set of rectangles, see figure 3b.

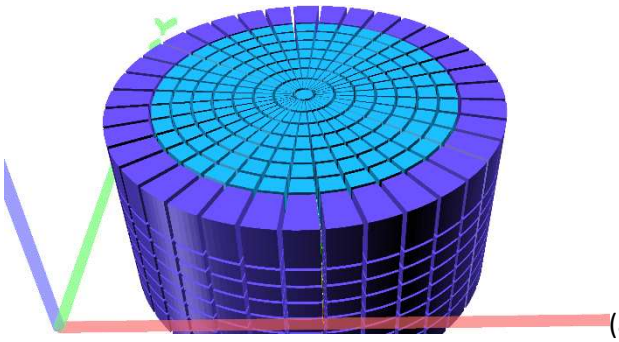
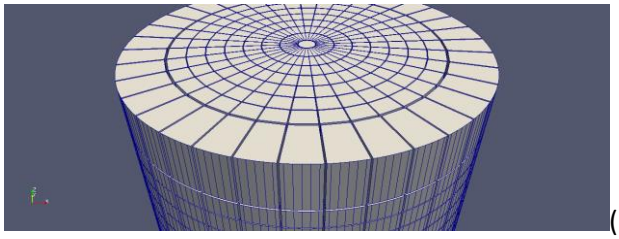

(a)

(b)

Figure 3 – a 3D view of a radial grid with cutting planes. The rectangular domain is cut by 180 planes. Each plane is tangent to a cylinder of radius 10 m. Note that boundary blocks have segmented boundary vertical faces. Instead block interface is a just one vertical face.

7.4 Example 4: A semi-telescopic grids

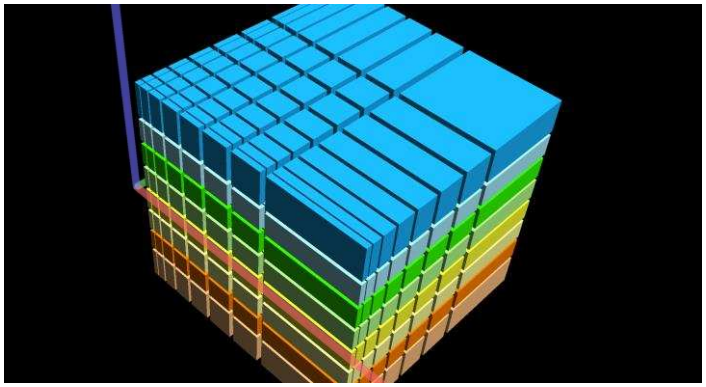


Figure 4 – Semi - Regular grid.

7.5 Example 5: A semi-telescopic grid with cutting planes

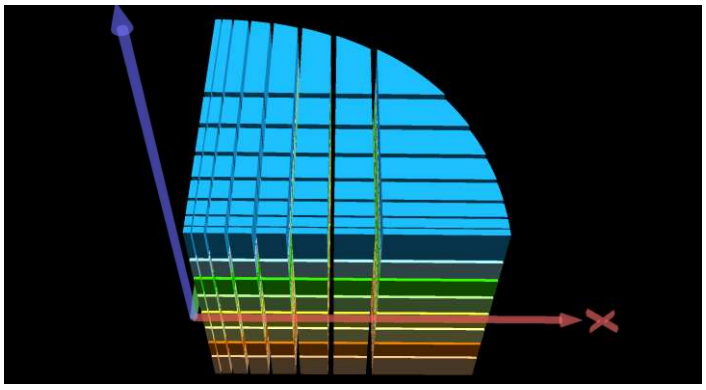


Figure 5 – A 3D view of the semi-regular grid, after applying cutting planes.

8 VORO2MESH – tools

How previously mentioned, the VORO2MESH discretization allows generation of totally unstructured grid for TOUGH2 simulations. Often discretization must take care about shape of geological structures, as surfaces, discontinuity, faults, pinch out. The arrangement of points nodes that permit to the 3D discretization to represent geological shapes is subject to several consideration.

1. A surface is well represented in the mesh when a set of nodes (called drive nodes) is positioned to each side of the surface.

2. The surrounding nodes are sufficiently far from the drive nodes.

Following these criteria, we develop a tool, named FitSurfaces(), to generate the set of the drive nodes conforming to a specified geological shape.

Note that the approach used by the tool is not fully 3D, in the sense that points are generated from a 2.5 gridded data representing a separation surface between geological layer. Therefore, for example, a sphere surface cannot be represented with this file format, but need of two file, one for each hemisphere. Also pinch out can be handled with some limitations.

Moreover, this technique consider sedimentary surfaces with low variation of deeps. In case of more complicated shapes, the tool can still perform the discretization of the domain and the rocktype assignation, but it cannot generate the drive nodes. Next release will take in account for faults and vertical planes.

9 To generate grids conforming to surfaces, user have to prepare the input surface gridded map in the file format as specified in Appendix A – Parameter File voro.par

The **voro.par** contains directive of working mode (point/surfaces), bounding domain, output options of VORO2MESH. Keywords are briefly resumed in the given example file.

```
##################################
# voro.par example parameter file#
##################################
#
# Domain definition
x_max=+192200.00
x_min=+16900.0000
y_max=+802800.00
y_min=+652800.000
z_max=-1000.00
z_min=-8000.0# Operating ways
fit_surface=1 #0:no fit, use in.dat as input file; 1=execute fit: the seed
point will be stored in the in.dat.ready file.
# MESH generation parameters
toler=1.001 #in the CONNE section, if area interface between adjacent
# blocks < toler, then connection will be discarded.
read_rocktype=1 #0:no,1=yes
block_names_format=0  #default=0,  standard  A3I[len_char_eleme_name-3];=1
use I[len_char_eleme_name]
len_char_eleme_name=5 #default 5; must be: 5<=len_char_eleme_name<=9
len_char_volume=10 #
len_char_eleme_surface=10 #
len_char_coordinates=10 #
len_char_d1d2=10 #
len_char_surface=10 #
len_char_cosine=10 #
#voro++ parameters
auto_nxnynz=1 #0=manual;1=auto, then n_x,n_y, n_z are not used.
n_x=7 # number of subdivision of the domain in x
n_y=7 # number of subdivision of the domain in y
n_z=7 # number of subdivision of the domain in z
# printing options
write_tough2viewer_dat=1 #0:no; 1:yes
print_vtkXML_file=1 #0=no;1=yes.
```

```
export_surface_as_ply=0 #0=no;1=yes: the input surface are exported as ply
files.
end=end #end of parameter file
```

Appendix B – input points file format.

9.1 FitSurfaces()

To activate this feature, in the **voro.par**, the variable fitsurface in **voro.par** must be set equal to 3 or 5 (the newest version with automatic local refinement in region with low thickness) or 7 (a new version with bug fixes and additional options).

If fitsurface is set to 3 or 5, then a file named **surfaceList.dat** must be present. In the file **surfaceList.dat,** for each surface to be used in the node generation, must be present a record with the following format:

[File Name] [(Float) offset] [(Float) block thickness].

Where:

[File Name]: is the file name containing the gridded surface data;

[(Float) offset]: is the distance of the generated drive nodes from the surface;

[(Float) block thickness]: is the thickness of the blocks (this value is used only when option vertical=2).

In the **surfaceList.dat**, the order of the surfaces must be vertically from top to bottom. Before of starting with mesh generation, a check about the input surface order is done, verifying if for each x and y, $z(i+1)(x,y)<z(i)(x,y)$. If this condition is not satisfied, the program will exit.

In case of pinch out, the z value representing absence of the surface must be set greater than z_max or lower of z_min (in other words, outside the domain to be discretized).

Gridded data are ASCII text format (GRASS data style), as specified in

.

In case of multiple surfaces, they must have the same number of rows and columns.

The number of generated point for each surface is 2x(rows-1)x(cols-1) without mesh refinement.

9.1.1 Drive nodes

In a 3D space, a surface can be represented as a regular grid of elevation points. This implies that it may be decomposed with a set of squares which corner are represented by four elevation coordinates. Therefore, a Voronoi tessellation including the surface can be obtained forcing a pairs of points (the drive nodes) on both sides of each 3D square, which represent the nodes of the 3D Voronoi blocks neighbouring to the surface. The distance of the points from the plane is set by the user (offset from the surface). For each surface can be defined a custom value of offset.

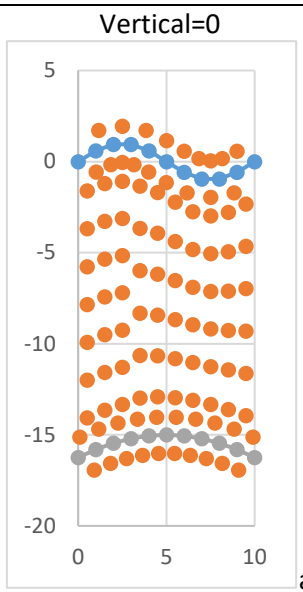
9.1.2 Inside points

We define “inside points” node are not in contact with the two surfaces of top and bottom. The vertical space between the drive nodes, the inside points, can be generated in several ways:

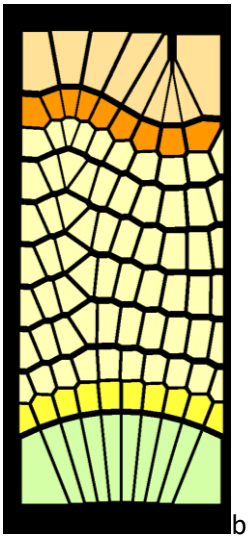
1. using the conjunction line between top and bottom drive nodes and vertically splitting the space with the same number of points (Figure 6a, 6b);
2. using a vertical line passing from the 3D squares centre and vertically splitting the space with the same number of points (Figure 6c, 6d);
3. using the vertical line passing from the 3D square centre and vertically splitting the space imposing to the points to have the same elevation value (Figure 6e, 6f).

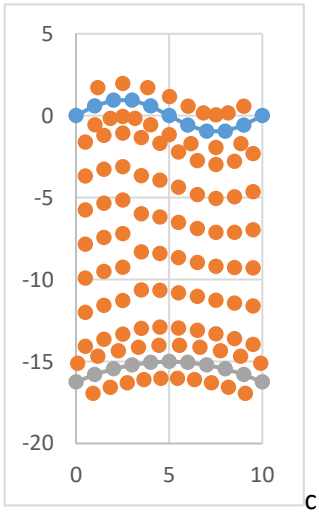
Comments on the 3 options:

1. vertical=0. This produces blocks with a better approximation of the surface, but the number of connections will increase, and as consequence increase the number of equations to be solved by the numerical simulation.
2. vertical=1. This allows to have vertical piles of blocks , but still produce high number of connections.
3. vertical=2. In this case, like the previous one, the nodes are along the vertical line passing from the centre of the 3D square of the gridded surface and the block-z elevation is a multiple of the block thickness. In this way, a variable number of blocks is obtained between two surfaces, but a regular discretization (far from drive nodes) reduce the number of connection to a minimum (6, for an internal parallel prism) for each node, resulting therefore in a reduction of equation to be solved by the numerical simulator.

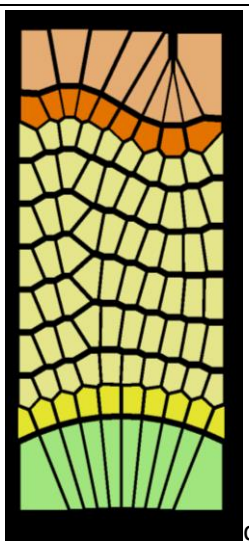


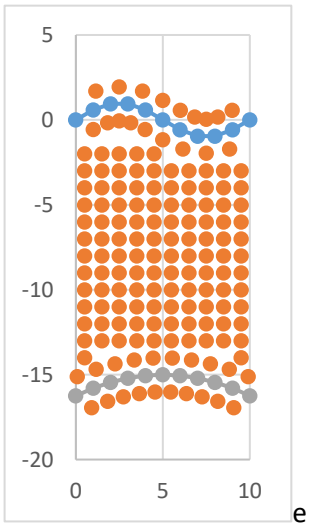
a


b



c


d



e

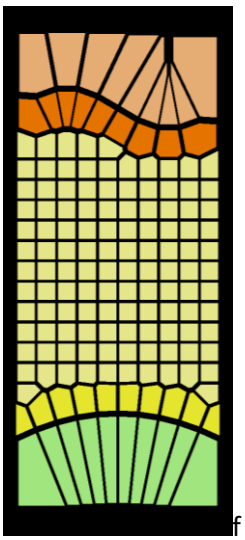

f

Figure 6 – Vertical discretization of a volume included between two surfaces. (a), (c) and (e) show the distribution of seed points. (b), (d) and (f) show sections of the corresponding discretized domain.

Special cases

1. The distance between two surfaces is lower than the offset used for the discretization. In this case only one point is put inside the two surface, and the drive nodes are modified to have a block with a correct thickness;
2. in case of pinch out, the corresponding drive nodes and inside points are not generated. The algorithm will refer to the next valid surface. In this case some manual rocktype assignation must be done.

9.1.3 Examples

In the following, some example of discretization is given.

9.1.3.1 Example 6: 2D, vertical slice, two surfaces

This example use two surface file as input. The inside points belongs to the conjunction line between the drive nodes (parameter keyword vertical=0 in the file **voro.par**). Each conjunction line is discretized with the same number of node points. See figure 6b.

9.1.3.2 Example 7: 2D, vertical slice, two surfaces

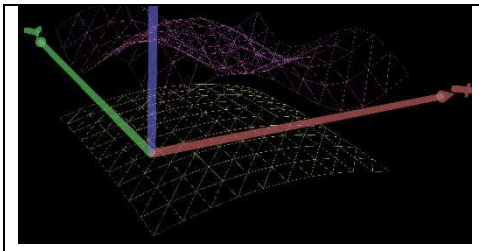
This example use two surface file as input. The inside points belongs to the vertical line between the drive nodes (parameter keyword vertical=0 in the file **voro.par**). Each vertical line is discretized with the same number of node points. See figure 6d.

9.1.3.3 Example 8: 2D, vertical slice, two surfaces

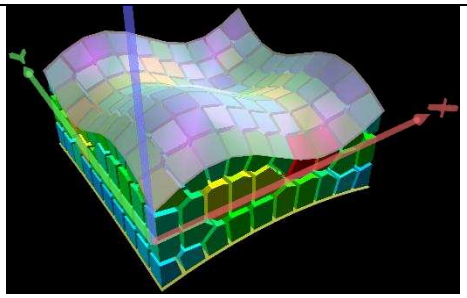
This example use two surface file as input. The inside points belongs to the vertical line between the drive nodes (parameter keyword vertical=0 in the file **voro.par**). Each vertical line is discretized at regular interval, to obtain regulars blocks. See figure 6f.

9.1.3.4 Example 9: 3D, two surfaces

This sample use two surfaces as input. In figure 7a are shown the input surfaces and in figure 7b the resulting model, removing blocks over the top and below the bottom surfaces. The discretization use the option vertical=2 to reduce the number of connections.



(a)



(b)

Figure 7 - -(a) input surfaces; (b) Grid blocks and surfaces

10 References

Barber, C.B., Dobkin, D.P., and Huhdanpaa, H.T., 1996. The Quickhull algorithm for convex hulls, ACM Trans. on Mathematical Software, 22(4),469-483, http://www.qhull.org.

Battistelli A., Marcolini M. (2009). TMGAS: a new TOUGH2 EOS module for the numerical simulation of gas mixtures injection in geological structures. Intl. J. Greenhouse Gas Control, 3, p. 481-493.

Berry, P., Bonduá, S., Bortolotti, V., Cormio, C., Vasini, E.M., 2014. A GIS-based open source pre-processor for georesources numerical modeling, Environmental Modelling & Software, 62, 52-64, ISSN 1364-8152.

Bonduá, S., Berry, P., Bortolotti, V., Cormio, C. (2012). TOUGH2Viewer: A post-processing tool for interactive 3D visualization of locally refined unstructured grids for TOUGH2. Computers & Geosciences, 46, p. 107-118.

Bonduà, S.; Battistelli, A.; Berry, P.; Bortolotti, V.; Consonni, A.; Cormio, C.; Geloni, C.; Vasini, E. M., (2017). 3D Voronoi grid dedicated software for modeling gas migration in deep layered sedimentary formations with TOUGH2-TMGAS, COMPUTERS & GEOSCIENCES, *in press*.

Bourke P. PLY - Polygon File Format. http://paulbourke.net/dataformats/ply/. Published 2011. Accessed May 3, 2016.

Cormio C., Berry P., Bonduà S., Bortolotti V. (2012). Innovative tools for continuum discretization, better management of TOUGH2 input data and analysis of the numerical simulation results. Proceedings, TOUGH Symposium 2012, LBNL, Berkeley, California, September 17-19, 2012.

Hang Si., 2015. TetGen, a Delaunay-Based Quality Tetrahedral Mesh Generator, ACM Transaction on Mathematical Software, 41 (2), Article 11, 36 pages, 2015. DOI=10.1145/ 2629697 http://doi.acm.org/10.1145/2629697.

Pruess K., Oldenburg C.M., Moridis G.J. (1999). TOUGH2 User's Guide, Version 2.0. Lawrence Berkeley National Lab., report LBNL-43134.

Pruess K. (2004). A composite medium approximation for unsaturated flow in layered sediments. Journal of Contaminant Hydrology. 70, p. 225– 247.

Rycroft, C.H., 2009. Voro++: A three-dimensional Voronoi cell library in C++, Chaos 19, 041111

11 Appendix A – Parameter File voro.par

The **voro.par** contains directive of working mode (point/surfaces), bounding domain, output options of VORO2MESH. Keywords are briefly resumed in the given example file.

```
##################################
# voro.par example parameter file#
##################################
#
# Domain definition
x_max=+192200.00
x_min=+16900.0000
y_max=+802800.00
y_min=+652800.000
z_max=-1000.00
z_min=-8000.0# Operating ways
fit_surface=1 #0:no fit, use in.dat as input file; 1=execute fit: the seed
point will be stored in the in.dat.ready file.
# MESH generation parameters
toler=1.001 #in the CONNE section, if area interface between adjacent
# blocks < toler, then connection will be discarded.
read_rocktype=1 #0:no,1=yes
block_names_format=0  #default=0,  standard  A3I[len_char_eleme_name-3];=1
use I[len_char_eleme_name]
len_char_eleme_name=5 #default 5; must be: 5<=len_char_eleme_name<=9
len_char_volume=10 #
len_char_eleme_surface=10 #
len_char_coordinates=10 #
len_char_d1d2=10 #
len_char_surface=10 #
len_char_cosine=10 #
#voro++ parameters
auto_nxnynz=1 #0=manual;1=auto, then n_x,n_y, n_z are not used.
n_x=7 # number of subdivision of the domain in x
n_y=7 # number of subdivision of the domain in y
n_z=7 # number of subdivision of the domain in z
# printing options
write_tough2viewer_dat=1 #0:no; 1:yes
print_vtkXML_file=1 #0=no;1=yes.
export_surface_as_ply=0 #0=no;1=yes: the input surface are exported as ply
files.
end=end #end of parameter file
```

12 Appendix B – input points file format

The file has one line header, flowed of the coordinate points in the following format:

[id] [x] [y] [z] [rocktype]

Where:

[id]: unique identificative number

[x] [y] [z] Cartesian coordinate

[rocktype] optional: the rocktype value for the node

Lines starting with character “#” are skipped.

Example:

Header – in.dat file format

0 72.594696 927.405334 -279.704498 1

1 27.405306 972.594666 -202.790009 2

2 164.947006 925.829895 -234.842972 1

3 135.052994 974.170105 -152.565567 2

4 250.021210 924.671692 -218.623108 1

5 249.978790 975.328308 -132.403046 2

6 335.082825 925.825134 -234.762024 1

7 364.917175 974.174866 -152.468536 2

…

(comment line)

13 Appendix C – List of Surfaces

The file must contain the ordered list of surfaces (from top to bottom). For each surface the user must specify the offset between the surface and the drive nodes (the nodes near the surface) and the block thickness between the drive nodes for the block below the surface (block thickness may change between layers).

Example:

```
surface_test_top.dat 70 140
surface_test_bottom.dat 70 140
```

14 Appendix D – Gridded surface file format

The boundary geological surface file (in GRASS ASCII format) has a header section that describes the location and size of the data, followed by the elevation of the surface points.

The header has 6 lines:

north: xxxxxx.xx
south: xxxxxx.xx
east: xxxxxx.xx
west: xxxxxx.xx
rows: r
cols: c

The north, south, east, and west field values entered are the coordinates of the edges of the geographic region. The rows and cols field values entered is the dimension of the data matrix. The following are 44 rows of 52, float or double data. Null values are identified by a value greater of z_max (or lower of z_min).

Example from the case study:

```
north: 801750
south: 655550
east: 191250
west: 17850
rows: 44
cols: 52
1.0E5 1.0E5 1.0E5 1.0E5 1.0E5 1.0E5 1.0E5 1.0E5 -3714.75 -3738.88 -3831.66 ……
```

15 Appendix E – Tough2Viewer.dat file format

The geometrical TOUGH2Viewer file has no header. For each block, the data structure is as follows (blank space separated values):

id x y z n_vertex [n_vertex(x,y,z)] n_faces [n_faces(i_vertex, i_vertex+1, …)] [n_faces(vx,vy,vz)]

Where:

Id: integer, block label. Not used.

x y z: block node coordinate

n_vertex: integer: number of block vertexes.

[n_vertex(x,y,z)]: sequence of x, y, z coordinates for each vertex enclosed between bracket. Coordinates are relative to the blocks center node.

n_faces: number of faces of the cell

[n_faces(i_vertex, i_vertex+1, …)]: bracket sequences (one for each face) of integers representing the vertex of each face.

[n_faces (vx, vy, vz)]: for each face, a bracket sequence of double representing normalized vector of the normal to the face.

To generate this file format from voro++, it is necessary to use the following command line options: "%i %q %w %p %s %t %l".

Example, one line for one block (in the following is reported only one line):

0 27.4053 27.4053 -202.79 10 (-27.4053,-4.62727,-83.8273) (80.8347,-27.4053,-33.616) (-27.4053,-27.4053,198.378) (61.7832,61.7832,10.0153) (-27.4053,80.8347,-33.616) (13.0548,72.5947,-14.6858) (72.5947,13.0548,-14.6858) (61.2534,61.2534,6.96889) (-5.43639,-27.4053,-84.3027) (-27.4053,-27.4053,-92.4444) 8 (1,6,7,5,4,0,8) (1,2,3,6) (1,8,9,2) (2,4,5,3) (2,9,0,4) (3,7,6) (3,5,7) (8,0,9) (0.451894,0.451894,-0.769145) (0.906139,-0.0132613,0.422771) (3.18904e-016,-1,1.22161e-017) (-0.0132613,0.906139,0.422771) (-1,9.51153e-016,-1.68966e-015) (0.924535,0.314346,-0.215455) (0.314346,0.924535,-0.215455) (0.327509,0.334319,-0.883724).

16 Appendix F – Statistic.log file format

For each rocktype, a short resume of the blocks volume (min, max, average, total) is reported. Moreover, a CONNE section resumes the elements with min and max number of connections, minimum, maximum and average interface area.

Example of a statistic.log file:

```
Number of rocktype= : 3
Statistics for rocktype: -1
Number of ELEME: 7983
Min Volume cell:   4.1036e+09
Max Volume cell:   2.3436e+10
Total Volume   :   7.5717e+13
mean Volume    :   9.4848e+09
Statistics for rocktype: 1
Number of ELEME: 36258
Min Volume cell:   3.1280e+08
Max Volume cell:   1.9197e+09
Total Volume   :   1.9528e+13
mean Volume    :   5.3806e+08
Statistics for rocktype: -2
Number of ELEME: 7983
Min Volume cell:   9.1746e+08
Max Volume cell:   2.7106e+10
Total Volume   :   6.8759e+13
mean Volume    :   8.6132e+09
Total Vol(con.vol) :   1.6401e+14
Total Vol sum rock :   1.6401e+14
Difference         :     -0.21875 [-1.33372e-13 (percentage)]
**************************
Number of CONNE: 192186
min num. conne : 6 element [A21 1]
max num. conne : 25 element [H72 1]
mean num. conne:       3.6895
min conne area :       1.0021
```

```
max conne area :     7.225e+06
```

17 Appendix F – Complete list of VORO2MESH input keywords.

The **voro.par** contains all directives and options for generating the MESH file for TOUGH2 with VORO2MESH.

KeyWord Name	Default value	Keyword description
add_walls	0 (int)	The voro++ library offer the possibility of cutting the domain by 3D planes. When used, this function need of a file named "wallslist.dat". The file has to contain a list of 4 space separated values ([A] [B[[C[[D]), that represent a plane using the implicit formulation Ax+By+Cz+D>0 (the cutting wall). Range values are: 0:no; 1:yes.
assign_infinite_volume_to_boundary	0 (int)	In an ustructured grid is not so immediate understand which blocks are on the boundary (geometrically speaking) or not. This option allow to assign a inifinite volume to blocks that are “touching” a wall plane or on of the six domain planes. Range values are: 0:no; 1:yes.
assign_roktype_to_gener	0 (int)	When calculating GENER elements using some criterion (like depth or primary values threshold), the rocktype is modified, for helping in visualization. Range values are: 0=no modify; 1=modify rocktype to . rocktype= assign_roktype_to_gener
auto_nxnynz	1 (int)	The voro++ library divide the domain in sub domains to speed up the node points search during voronoi cell computation. The user can choose if give in input the subdivision (along the 3 cartesian axis) Range values are: 0=manual; 1=auto.
block_names_format	0 (int)	The default (standard) block name format of TOUGH2 is A3I2 (3 characters, 2 numbers). This option allow to increase the number of numbers to be used for block naming as: 0 (default),standard A3I[len_char_eleme_name-3]; 1 use I[len_char_eleme_name]
blocks_thick	70.0 (double)	Used when fit_surface keyword>0. It define default block thickness.
check_4_duplicates	1 (int)	The Voronoi discretization assume that no more of one point can be located in the same position. If the

KeyWord Name	Default value	Keyword description
		distance between two pairs of points is lower of toler_dist2, then execution of VORO2MESH is stoped.
coarse_mesh	1 (int)	Not implemented
create_gener	0 (int)	For special cases, a GENER file is created. Range values are: 0=no; 1=create.
create_incon	0 (int)	A ready TMGAS compatible file is created, with variable assignation driven by linear interpolation of the coordinates: X1=A1x+B1y+C1z+D1. If create_incon>0, then a file incon.dat must be created. The file incon.dat must contain 4 rows with 4 double space separated representing A1 B1 C1 D1 for thec calculation of the primary variable as defined previously. Range values are: 0=no; 1=yes, create.
cut_model_top_bottom	0 (int)	When fit_surface>0, the domain is divided in n_of_surfaces+1 layer with automated rocktype asignation. If cut_model_top_bottom>0, then uppermost and bottommost layer are excluded from the mesh FILE.
CVT_max_iter	500 (int)	Maximum number of iteration of CVT calculation.
delta_surface_cut_off	1.0 (double)	When usinf fit_surface>0, a check of the surface is performed. For each xy node, the distance between 2 surface have to be greather than delta_surface_cut_off. If not, the corresponding mesh generationis skypped generating : A pinch out A mesh refinement
dist_mode_por_perm	1 (int)	For special cases, VORO2MESH can compute cell by cell permeability values (TMGAS special input feature). The compute of the distance between the location of the node and the referenced table of permeability and porosity can be done on 2D or 3D mode. Range values are: 1: 2D 2: 3D
divide_control_inside_points	0 (int)	If fitsurface>0, for each layer between two surfaces, a different rocktype is assigned to top drive nodes, bottom drive nodes and inside node. Range values are: 0: no distinction between drive and inside node; 1: the rocktype assignation consider drive nodes and inside nodes.
End	(string)	The **voro.par** termination line is “end=end”.

KeyWord Name	Default value	Keyword description
exclude_out_pts	1 (int)	During reading of in.dat file, all points outside the domain (out from xmin,ymin,zmin and out of xmax,ymax and zmax, and outside from the walls domain are excluded from input. As the same way, when using fit_surface>0 some points can be outside from the domain. This option allow also skip writing outside points. Range values are: 0: no 1: skip
export_surface_as_ply	0 (int)	The input surface, that have to be in GRASS ASCCI format (see **Errore. L'origine riferimento non è stata trovata.**), can exported as PLY file, usefuel for visualization. Range values are: =: no export 1: yes, export,
fit_surface	0 (int)	This option activate automated seed points generation. If fit_surface>0, then a file named SurfaceList.dat have to present, plus a set of surface file as specified in the SurfaceList.dat file. The SurfaceList.dat file must contain a list of file name that represent the geological separation between geological layers. The format of the file SurfaceList.dat is for each line: [surface name file] [block_thickness] [offset from surfaces]. Note that at list 2 surfaces have to present to compute the discretization. Range values are: 0: no 3: ->fit_surface3() execution 5: ->fit_surface5() execution: an improved version of fit_surface3() function with local refinement 7: ->fit_surface7() execution: an improved version of fit_surface5() function with resolved bugs and fixes. Range values are: 0: no fit 3: first version implemented 5: classical 7: advanced
format_por_perm_tables	0 (int)	The format of the txt file containing porosity and permeability information. The actual versin is just for reading [x] [y] [z] [por] [perm]. Under construction other formats. Range values are 0;-> [x] [y] [z] [por] [perm].
HLBFGS_FLAG	0	Not used

KeyWord Name	Default value	Keyword description
	(int)	

iCVT	0
	(int)
		VORO2MESH is able to compute a Centroidal Voronoi Tessellation (CVT) (under development).
		By activating this option, a iterative (number of iteration= CVT_max_iter) Lloyd (or more complicated methods not described here) relaxation of the seed points is performed to obtain a CVT tessellation.
		Range values are:
		iCVT=1 -> CVT_Lloyd();
		Under developments:
		iCVT=2 -> not used
		iCVT=3 -> CVT_Lloyd_surface()
		iCVT=4 -> Antonio De Lorenzo()
		iCVT=11 -> CVT_Lloyd_weighted()
		iCVT=12 -> CVT_withHLBFGS();
		iCVT=13 -> CVT_withHLBFGS_with_Lp();

i_CVT_example	1 (int)	When using iCVT=11, special density functions for weighted CVT computation is set by using this parameter. To be documented in a future manual "Weighted CVT with VORO2MESH".
k_cvt	0.0 (double)	Not used
k_s	0.0 (double)	Not used
len_char_coordinates	10 (int)	The standard length for coordinates in the MESH file is 10 characters. To improve precision, a different value can be used. Note that tough2viewer is only able to read MESH file with standard length.
len_char_cosine	10 (int)	
len_char_d1d2	10 (int)	
len_char_eleme_name	5 (int)	
len_char_eleme_surface	10 (int)	
len_char_surface	10 (int)	
len_char_volume	10	
local_refine_number	3 (int)	If fit_surface>3 (eg 5 or 7), a block is refined when the ration thickness/edge dimension is lower than block_thickness/(min(dx,dy).
max_dist_por_perm	3400.0 (double)	In special cases, the block by block porosity/permeability assignation take a set of neighbouring values with distance lower than max_dist_por_perm.
min_2d_dist	0.1 (double)	

KeyWord Name	Default value	Keyword description
min_distance_from_walls	0.1 (double)	Points near walls are skipped from input if distance is lower than min_distance_from_walls.
n_layers	5 (int)	When fit_surface>0 and vertical<2, the space between the upper and the lower drive nodes is divided in n_layer blocks.
n_x	7 (int)	Voro++ option. The number of subdivision domain for n_x node search. Ignored if auto_nxnynz=1
n_y	7 (int)	Voro++ option. The number of subdivision domain for n_y node search. Ignored if auto_nxnynz=1
n_z	7 (int)	Voro++ option. The number of subdivision domain for n_z node search. Ignored if auto_nxnynz=1
number_of_points_por_perm	4 (int)	In special cases, the block by block porosity/permeability computation is done with the specified number of data. Range values: 4.
offset	70.0 (double)	If fit_surface>0, is the default values for the drive_node points distance from the surfaces. A value for each surface(up and down) is specified in the surfacelist.dat file.
offset_rocktype_boundary	10 (int)	When a block is geometric boundary, then it can have a different rocktype (usefull for visualization or identification) defined as rocktype=rocktype+offset_roccktype_boundary
por_perm_upscaling	1 (int)	1:average 2.Logarithmic average
print_vtkXML_file	1 (int)	Write a vtu file of the model
r_max	2.0 (double)	Obsolete. Not used
read_por_perm_tables	0 (int)	For special case, compute the block by block porosity/permeability computation for TMGAS input files
read_rocktype	1 (int)	The in.dat file can contain the rocktype material for each block. The rocktype is assigned in the MESH file (see TOUGH2 manual for details). Range value for the variable are: 0=no (in.dat with 4 columns); 1=yes(in.dat with 5 columns).
refine_mesh	1 (int)	Not used
toler	1.0E00 (double)	Interface area tolerance. If the area is lower than toler, then connection is skipped from output CONNE
tolerance_walls	1.0E-03 (double)	To evaluate if a point belong to the domain, is evaluated the implicit equation ax+by+cz-d>tolerance_walls.
two_digit	1 (int)	In the previous version of visual studio, the default scientific notation was wth 3 digit (1.0E+001). The two digit option give 1.0E+01) set the number of digit to 2.

KeyWord Name	Default value	Keyword description
use_w_as_first_letter_rocktype	0 (int)	For the T2well is needed to know what are wells elements by naming with a “w” as first character of the eleme name.
variable_n_layers	0 (int)	If fit_surface>0 the number of block between drive nodes can be fixed or variables.
verbose	0 (int)	Define the grade of printing message to the logfile and screen. Range values are: 0: limited printout; 1: detailed printout.
vertical	0 (int)	The position of the inside points. 0: along conjunction line, are inserted n_layers blocks. 1: along the vertical line passing for the centre of the gridded surface, are inserted n_layers blocks. 2: a regular grid is build internally to the drive nodes. The number of blocks can vary.
wall_type	0 (int)	Obsolete, not used.
write_boundary_info	0	In the MESH file, a string of 0 or 1 is written to identify ia a block “touch” a boundary or a cutting wall.
write_boundary_surface	0 (int)	The area of the face created by the walls is written in the MESH file. Useful for boundary conditions assignation.
write_preview_vtu	0 (int)	After node point creation with fit_surface is created to take a preview of the mesh Range values are: 0: no; 1: yes.
write_tough2viewer_dat	1 (int)	Range values are: 0: no; 1: yes; (%g) 2: yes (%.17f).
x_min	(double)	Domain definition
y_max	(double)	Domain definition
y_min	(double)	Domain definition
z_max	(double)	Domain definition
z_min	(double)	Domain definition
x_max	(double)	Domain definition