GHS3D, tetrahedral mesh generator

INRIA - Simulog-Technologies

March 7, 2005

1 Abstract

This technical note describes the GHS3D[1] automatic mesh generator. The available version is the so-called V3.0 release[2]. Are defined the aim of the software, the input files (a surface mesh as input data) and the output files[3] (the resulting tet mesh). The causes of error are listed and possible corrections are proposed. Typical application examples are shown and briefly discussed. A short technical description of the algorithm and the various ways to obtain the software are given.

GHS3D has been developed by the team *Gamma* (formerly *Modulef*) of INRIA in collaboration with Distène (the former Simulog-Technologies). The GHS3D software has been included in different commercial software packages proposed by various companies such as ANSYS, Matra-Datavision, MSC, E.S.P., Samtech, Distène, ..., and many others. It is also intensively used directly by different end-users such as E.D.F., Renault, SNECMA, Dassault-Aviation, and many others (such as dozen of university labs in France, in the States and in a number of other countries).

2 Aim of the mesh generator

GHS3D is a *fully automatic mesh generator* which aims at providing a tetrahedral mesh within a domain defined by a conforming mesh of its boundary. This surface mesh is usually made up of triangles (while quad elements are also supported). Surface mesh creation is not included in the generator. However, the user is responsible to provide this surface mesh, for instance using its favorite CAD-CAM software package or using other surface meshing tools (such those developed within the team *Gamma*).

[1] Also referred to as TetMesh-GHS3D.

[2] The current release offers a **series of improvements** as compared with the previous releases. Improvements and changes include CPU cost, less number of tet, multi-material domain, shape quality, robustness, extra delaunisation, self-centering and a series of changes that makes the implementation easier. Some of these improvements (see Section 7.5) have been motivated by microelectronics concerns and developed with the support of the EEC Magic_Feat project (IST Project 1999-11433).

[3] In the case of a brute usage of the software.

The volume mesh is governed by the properties of the surface mesh given as input. The latter clearly governs the element density together with the mesh quality. In fact, a major feature of the GHS3D algorithm is to preserve the surface entities meaning that any surface face will be present in the resulting tet mesh and encountered as a tet face.

Remark : in addition to the surface mesh, (internal) points, edges and faces can be explicitly specified to govern the mesh generation process. These specified items will be present in the resulting mesh and, in other words, may give some degree of user's guidance.

Other usages of GHS3D.

Three different usages of GHS3D can be made for specific applications.

GHS3D can be used as *a mesh optimizer*. In such a case, a previously created tet mesh is entered as data input and the algorithm optimizes this mesh in terms of element quality (aspect ratio, see below).

GHS3D also produces diagnostics regarding the correctness of a surface mesh (see ERR 3*** in Section 10) and thus can be used to check the validity of a given surface mesh. In case of a wrong surface mesh (self-intersection, overlapping elements, holes, lack of conformity, etc.), these diagnostics help the user to correcting the surface mesh. Here is a more "exotical" usage of the GHS3D algorithm that allows for a *very fast detection* of surface mesh failures (this could be obtained in a few seconds or minutes based on the number of faces).

GHS3D can be also serve to construct a *hull* of a given set of points. In this case, the data consists of a set of points and no faces (meaning that the number of faces, NF in the above *xxx.faces*, is null). The output is a *xxx.boite* file (as described below) which indeed is a triangulation of a certain hull of the points. In some cases, this hull is the well-known convex hull, in others it may be a little bit different.

3 Data

Two ASCII files describing the surface mesh defining the volume of interest are input as data. The surface mesh must be conformal. The input files are called *xxx.points* and *xxx.faces* (*xxx* being a basename) respectively.

3.1 The *xxx.points* file

This file stores the point coordinates. See Annex.

Remark : usually, file *xxx.points* contains $NP + 1$ records where NP is the number of points. In case the user wants to specify some internal points, they can be entered in lines $NP + 2$, ..., $NP + n$. (meaning that $n - 1$ extra points are provided). Such points can be used to specify the density in such or such area around them in the domain, thus, a stepsize (element size) must be given for each of them (see Annex).

3.2 The *xxx.faces* file

This file stores the boundary facets. See Annex.

The boundary mesh can be made of triangular facets but quad facets are also supported. Prescribed edges may be specified and referred to as a type 2 facet.

A triangle is a type 3 facet with three vertices, a quadrilateral is a type 4 facet while an edge is a type 2 face. Other types of facet are not supported. Note that a quadrilateral facet is automatically subdivided into two triangles.

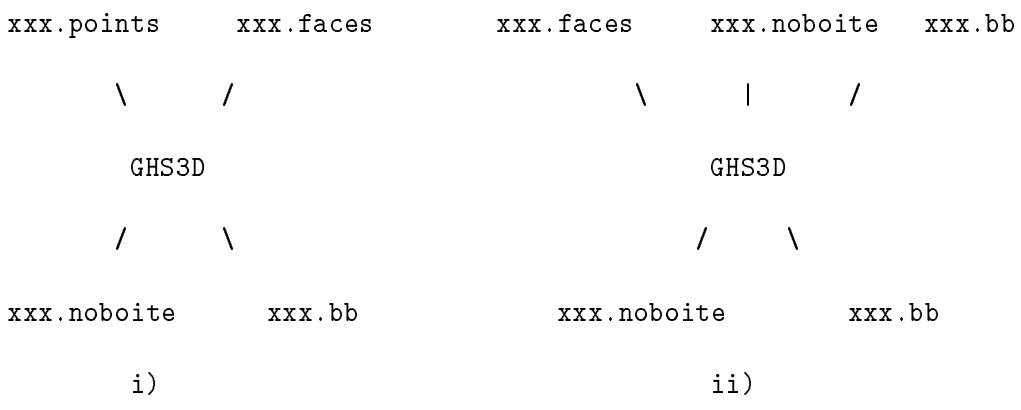
3.3 More about the data files

A few remarks about the input files and the file organization scheme.

Remark : GHS3D also supports a *xxx.mesh* file instead of the *points* and *faces* files.

Remark : as optimizer, GHS3D uses the *xxx.faces* file, a *xxx.noboite* file and a *xxx.bb* file, the two latter being described below. In this case, the output files *xxx.noboite* and *xxx.bb* replace the corresponding input files (with the same name).

Remark : if incorporated in an existing software, GHS3D is seen in a rather different way. Actually, for efficiency reason, it is clear that the data must be provided in a different way. Obviously, the two above input files must be avoided and replaced by a direct call while the output files must be also avoided since it is possible to have a direct access to the tables of interest (element vertices, vertex coordinates, ...).



File organization : i) standard use of GHS3D, ii) GHS3D used as a mesh optimizer.

4 Output

As indicated in the above flowchart, there are two output files, a *noboite* and a *bb* file. The *noboite* file basically reports the tets and the point coordinates while the *bb* file reports the corresponding sizing information (*i.e.,* a size is associated with the mesh vertices, these scalar values being automatically produced by `GHS3D`).

For a *noboite* file, binary as well as ASCII output can be produced while the *bb* file is an ASCII file. See Annex.

Remark : in some case, the algorithm does not complete the mesh. Most of these cases of failure are encountered at the time the boundary items are regenerated (see the general scheme of the algorithm).

In such a situation, an output file can be created, the so-called *xxx.boite*, this file can be used as data for a new run of the software. The *xxx.boite* file has the same structure as the *xxx.noboite* but does not contain the three last records. See Annex.

4.1 Bb ASCII file

The *bb* output file reports, for each mesh vertex, a stepsize. This ASCII file is organized as described in the Annex.

Note : The stepsize in a mesh vertex denotes, in some sense, the expected lenght of all the edges emanating from this vertex.

4.2 Sub-domain (sub-region) affectation

In the case where several regions are meshed by `GHS3D`, the question is to make it possible to know to which sub-region a tet belongs. In this way, it will be easy to assign a region flag (a material property) to each tet. Records 4 and 5 of the "noboite" output file (see Annex) have been designed in this purpose.

First, an integer value is given to each mesh element (record 6). Second, a triangular face is associated with each sub-region (record 5). Therefore, the responsability of the user is to establish the correspondence between the above integer value and its own system of sub-domain characterization.

What to do ? Pick a given face (for instance that characterizing the sub-domain with number 1, *i.e.,* the first face triple in record 5. Find where it is, then retrieve your own number (for example 3) or system of sub-domain definition. Then all the tets with the given sub-domain number (here 1) must be considered as subjected to the treatment related to your own number (here 3) or any equivalent way as defined in your sub-region definition system.

Remark : Clearly, when $SUBNUMBER = 1$, all this task is obvious and not really useful. This is why records $4, 5$ and 6 are organized as previously described (see the above corresponding note).

Remark : When $SUBNUMBER > 1$ or in case of holes, see parameter $OPTION$ described in Section 8. This value must be appropriately related to what is expected.

5 Errors

At the time the GHS3D has completed an execution, three cases can be encountered :

- a *xxx.noboite(b)* file exists : the resulting mesh is correct,
- a *xxx.boite(b)* file exists : the algorithm has failed during the boundary regeneration phase,
- no output file has been created : the data (surface mesh) is not correct (the boundary is self-intersecting in various ways or the initial set of points contains, at least, two points which have been found identical).

More precisely, the reasons for which the algorithm failed have been catalogued as follows :

- the surface mesh contains "facets" other than edge, triangle or quadrilateral,
- the surface is self-intersecting (at least, a surface edge intersects a surface facet, an edge intersects another edge or a facet intersects another facet),
- the surface mesh includes hole(s) so that the domain is not properly defined,
- at least, two given points are identical or judged too close,
- the memory size is not sufficient,
- the boundary regeneration is not achieved (mostly due to the case where the surface quality is too poor or when incompatible size face occurs for two faces that are very close).

Section 10 lists the different possible errors that can be encountered and gives a coarse indication regarding the type of the detected error.

Remark : Most of the failures of the mesh generation algorithm are due to rather bad surface mesh (in terms of element aspect ratios). To prevent this type of failures, it is advised to input nice surface mesh (noticing at the same time that such a demand is quite natural for finite element computing since the tet quality is strongly related to the surface facet quality).

6 Technical description of the algorithm

In this section are indications about the general scheme of the method together with some remarks about how to include GHS3D in an host package.

6.1 Flowchart of the algorithm

In brief, the mesh generation algorithm is of a *Delaunay* type. The general scheme of the method is given by :

- reading of the two input data files,
- insertion process of the given points (by means of a constrained Delaunay type method),
- boundary surface regeneration (basically by repeated face-edge swapping and various other techniques including the insertion of a few Steiner points),
- creation of the field points (based on the distribution of the surface vertices),
- insertion process of the field points (using a constrained Delaunay type method),
- optimization (by relocating the field points or by means of topological operators),
- writing of the output files.

Technical details as well as algorithms used in the mesh generation procedure can be found in a list of references provided in the bibliography section of this document.

6.2 Including GHS3D as an algorithm

Actually, the software package GHS3D consists of about 240 subroutines and, roughly speaking, is organized in three main groups.

- INPUT : is related to the input data reading (the two input files),
- ALGO : is related to the mesh generation algorithm,
- OUTPUT : is devoted to writing the resulting mesh and the corresponding step-sizes onto two files.

It is thus possible to extract the second step in order to include the mesh generation algorithm in a given software package (thus taking benefit from the current background available while reducing the CPU cost).

Note also the subroutine PRIERR in which are the print orders. See also the subroutines LIRNBO and ECRNBO designed for reading and writing a "noboite" type file.

7 Application examples

7.1 Goal

The aim of the method is to construct an isotropic mesh consisting of tetrahedral elements (tet in short) enjoying the best possible quality (the optimum being the regular ("equilateral") element). Thus, the expected resulting mesh must include a maximum of well shaped elements and, on the other hand, must have as worst element(s) that (those) possessing a bad shaped surface face. This demonstrates how the surface mesh quality is a crucial factor. When the mesh generation process is completed, an histogram helps the user to appreciate the mesh quality. This histogram reports the number of elements falling into various ranges of quality.

The quality function retained in `GHS3D` is defined, for tet K, as follows :

$$\mathcal{Q}_K = \alpha \frac{h_{max}}{\rho_K}, \tag{1}$$

where h_{max} is the element *diameter*, *i.e.,* its longest edge while ρ_K is the inradius of element K. Notice that this value varies between 1 to ∞ and that the more $\mathcal{Q}_K$ is close to 1, the best is tet K. Indeed, $\mathcal{Q}_K$ measures the so-called *degradation* of element K.

7.2 Programming language and supported computers

The software is written in F77 (one routine being in C).

Basically, the software has been developed using HP workstations. Nevertheless, a lot of different types of workstations and mainframes have been used to validate the software. Among them are SUN, Silicon Graphics, DEC alpha, IBM 6000, Cray C90, ..., as well as various PC's.

A main program, in C, namely *ghs3d*, is available. Options are possible including

```
-h [-formatted|-f] -m k -o opt -p print PrefixFile
```

by typing -h, information is provided about the meaning of the various options that are available.

7.3 Memory requirement

About 100 integer values are required to process one point, thus it is possible to assign the memory space which is expected. A typical memory space needed for up to 600 000 elements is about 16 000 000 words meaning about 64 MegaBytes (32 bits architecture).

The main program, written in C, allows the dynamic memory allocation. The default value is 16 000 000 words for the memory size. According to the expected size of the mesh, it is possible to change this value by adding on the command line (in Unix system) the desired value in million of words. Thus typing

```
ghs3d -m 10
```

stands for the allocation of 10 millions words (where ghs3d is the name of the main), while

```
ghs3d
```

stands for the default value.

7.4 Cpu time expected

The CPU time expected for creating a mesh (including i/o) is a function of different parametres including, in particular, the size (the volume) of the domain and the quality of the surface mesh. It is reasonable to expect (using a modern workstation) a CPU time (including I/O) of :

- few seconds to create a mesh of one to several thousands elements,
- about one minute for roughly one million elements.

Note that a 50 million tet mesh has been successfully constructed (using a 64 bits version of `GHS3D`. The CPU time is then impeded by some cache arguments.

7.5 More about quality concerns

The main concern of `GHS3D` is to construct good quality elements. To this end, the quality function is that previously defined. On the other hand, various other properties are emphatised (since Version 3.0).

Delaunay criterion. While internally based on a Delaunay type method (for point connection), the mesh generator does not guarantee that the resulting mesh is fully Delaunay. In this way, non Delaunay surface mesh can be handled with success. Nevertheless, most of the tet inside the mesh are Delaunay tets apart in regions where the input surface mesh is not suitable to guarantee such property. However, the current release includes an algorithm that favor this property.

Self-centered elements. Elements opposite a boundary face have received a special treatment that aims to make them self-centered (a tet is self-centered when its circumcenter falls inside it). This property is of interest, in specific, when a finite volume method is used furthermore.

7.6 Application examples

The `GHS3D` automatic mesh generator has been tested on more than 500 significant examples resulting in meshes ranging from 4 elements to more than 10 million elements. Two of them are depicted in the following figure, while Tables 1 and 2 report the cost (in terms of cpu time) to process these two cases and larger ones. Table 1 recall the previous versions capabilities while Table 2 refers to the current release (the CPUs are those obtained using a HP9000-C3600 (PA8600-552Mhz)).

In the first two tables, ne is the number of elements, np is the number of vertices, t denotes the CPU time in sec. (including i/o) while v indicates the number of elements created within one minute. Del is the percentage of effort used in the Delaunay part of the mesh generation algorithm (*i.e.,* Del corresponds to the insertion of both the given points and the bulk points created inside the given domain).

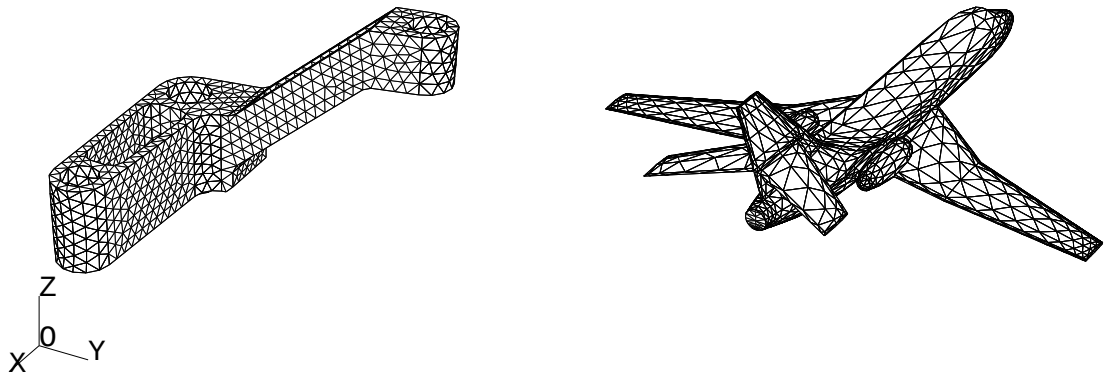


Figure 1: *Mesh example in solid mechanics (courtesy of SDRC, Cincinatti, OH). Mesh example in CFD (courtesy of Dassault-Aviation).*

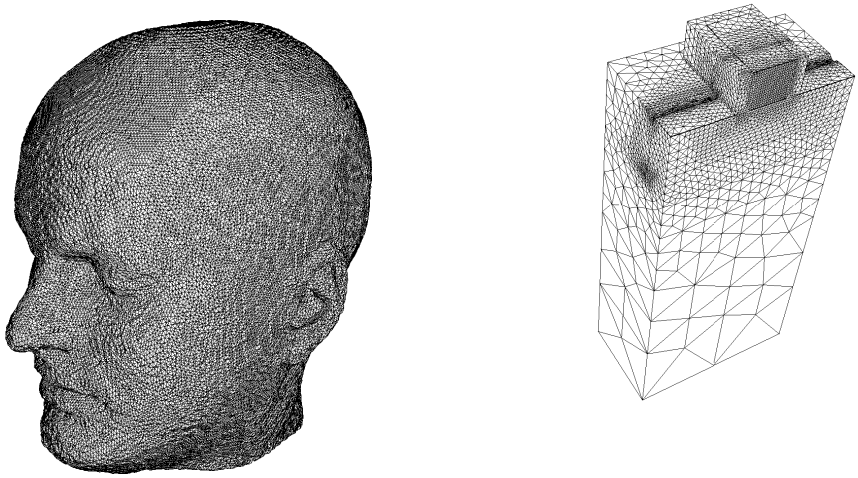


Figure 2: *Mesh example in biomedicine (courtesy of Low Temp. Lab., Helsinki Univ.). Mesh example im microelectronics (EEC Magic_Feat IST project).*

-	*np*	*ne*	*t* (in sec., HP 9000)	*v*	*Del*
Example 1	1 014	3 601	1.54	140 000	17
V2.0	1 077	3 986	0.84	284 000	16
Example 2	36 252	191 279	29.39	390 000	49
V2.0	29 630	151 457	15.60	582 000	53
Example 3	62 495	369 304	58.93	376 000	44
V2.0	22 871	131 668	13.40	590 000	39
Example 4 (V2.0)	110 292	614 691	64.60	571 000	53
Example 5 (V2.0)	269 993	1 562 489	182.51	513 000	48

Table 1: `GHS3D`: Cpu requirements, V1.0 (HP 9000/735) and V2.0 (HP 9000/780).

-	np	ne	t (in sec., HP 9000)	v	Del
Example 1	1 040	3 751	0.45	500 000	14
Example 2	15 293	85 764	3.86	1 300 000	22
Example 3	25 689	127 404	5.07	1 500 000	31
Example 4	79 764	430 033	18.30	1 400 000	38
Example 5	157 261	879 173	43.83	1 200 000	40
Example 6	337 945	1 951 845	97.67	1 200 000	37

Table 2: `GHS3D`: Cpu requirements, V3.0 HP 9000/C3600.

-	np	ne	t (in sec., HP 9000)	v
Example 1	1 050	3 804	0.17	1 300 000
Example 2	15 509	86 93 4	1.45	3 600 000
Example 3	25 984	129 060	2.81	2 700 000
Example 4	80 278	432 815	8.49	3 000 000
Example 5	158 500	885 914	26.36	2 000 000
Example 6	540 356	2 931 116	161.47	1 100 000

Table 3: `GHS3D` as an optimizer : Cpu requirements, V3.0 HP 9000/C3600.

The third table demonstrates how `GHS3D` can be used as an optimizer. Note, in this case, that the CPU includes the i/o which can be time consuming (based on the size of the mesh under optimization).

Version V3.0, Table 2, benefits from various improvements leading to better quality meshes with less nodes and rather less CPU requirements (while biased by the change of computer, this point has been demonstrated using again some old machines in principle out of service).

Remark : The above statistics (mostly for t) are strongly dependent on the computer where the program runs. Nevertheless, computers of various computer-makers with almost similar capabilities lead to similar characteristics.

7.7 A typical output

```
../ghs3d
   PRINT  (0 10(advised) -10) ?
10
  -- GHS3D (INRIA) Release V2.0
  -- FILE BASENAHE ?
bielletest
  %% bielletest.points    FOUND
  %% bielletest.faces     FOUND
  **  points AND  faces
   OPTION ? : 0 ==> STANDARD
  -1   , -2      , -3               , -4    , -5     , - 6       , -7  ==> ** NUHBER OF ELEHENTS WORSE THAN TARGET VALUE  7
  NOPO, - optim, before integrity, sd  1, all sd, boundary , private
0
  -- READING THE DATA
  -- READING THE POINT FILE bielletest.points
     OPERATION COHPLETED
  -- READING THE FACE FILE bielletest.faces
     OPERATION COHPLETED
     NUHBER OF GIVEN VERTICES   758
     NUHBER OF GIVEN TRIANGLES  1512
  -- DATA READING COHPLETED

  -- SURFACE HESH QUALITY  1512
     WORST ELEHENT QUALITY  11.32872
     BEST  ELEHENT QUALITY  1.00757
     WORST ELEHENT   220 226 308

     HISTOGRAH

    1  < Q <    2    81%      1231
    2  < Q <    3    14%       222
    3  < Q <    4     2%        43
    4  < Q <    5     0%         8
    5  < Q <    6     0%         2
    6  < Q <    7     0%         0
    7  < Q <    8     0%         2
    8  < Q <    9     0%         0
    9  < Q <   10     0%         0

   10. < Q <  100.     0%         4


  ** TARGET VALUE  8.30351


  ##################################################
   HODULE GHS3D-INRIA : V2.0
  ##################################################

   HAXIHUH NUHBER OF POINTS    149177
   HAXIHUH NUHBER OF ELEHENTS  596708

  -- PHASE 1 : SPECIFIED POINTS
  -- PHASE 1 COHPLETED

  -- PHASE 2 : BOUNDARY REGENERATION
  --  HISSING FACES  136 AHONG  1512
  --  HISSING EDGES  69 AHONG  2268
  --  HISSING EDGES  4  AHONG  7
  --  HISSING EDGES  2  HISSING FACES  4
  --  HISSING EDGES  2  HISSING FACES  4
  --  HISSING EDGES  0  HISSING FACES  0
  -- PHASE 2 COHPLETED

  -- PHASE 3 : FIELD POINTS
  --  34 POINTS PROPOSED  0 POINTS DISCARDED
  --  222 POINTS PROPOSED  1 POINTS DISCARDED
  --  48 POINTS PROPOSED  0 POINTS DISCARDED
  --  6 POINTS PROPOSED  0 POINTS DISCARDED
   -- PHASE 3 COHPLETED

  -- PHASE 4 : OPTIHIZATION
  -- PHASE 4 COHPLETED
```

```
-- RESULTING HESH QUALITY  3986
   WORST ELEHENT QUALITY  11.06646
   BEST  ELEHENT QUALITY  1.04676
   WORST ELEHENT   466 501 502 462

   HISTOGRAH

  1  < Q <    2    78%      3124
  2  < Q <    3    16%       668
  3  < Q <    4     2%       114
  4  < Q <    5     1%        41
  5  < Q <    6     0%        12
  6  < Q <    7     0%        12
  7  < Q <    8     0%         3
  8  < Q <    9     0%         6
  9  < Q <   10     0%         4

 10. < Q <  100.     0%         2

##############################################################################
 END OF HODULE GHS3D-INRIA
##############################################################################


-- WRITING THE OUTPUT FILE
   NUHBER OF GIVEN VERTICES   758
   NUHBER OF CREATED VERTICES 319
   TOTAL NUHBER OF VERTICES   1077
   NUHBER OF TETRAHEDRA       3986
   NUHBER OF SUB-DOHAINS      1
-- WRITING COHPLETED (  6 RECORDS ) bielletest.noboiteb
-- WRITING COHPLETED (  2 RECORDS ) bielletest.bb

 TOTAL CPU   .9800000000000001 SEC.
 244040 ELEHENTS ( 265733 ) WITHIN A HN.  FOR bielletest


##############################################################################
 HODULE GHS3D-INRIA : V3.0
##############################################################################

........


-- RESULTING HESH QUALITY  3751
   WORST ELEHENT QUALITY  9.15487
   BEST  ELEHENT QUALITY  1.06492
   WORST ELEHENT   308 655 226 220

   |------  HISTOGRAH  ------|

  1. < Q <    2.   77%      2873
  2. < Q <    3.   19%       718
  3. < Q <    4.    2%        79
  4. < Q <    5.    1%        38
  5. < Q <    6.    1%        23
  6. < Q <    7.    0%        13
  7. < Q <    8.    0%         1
  8. < Q <    9.    0%         4
  9. < Q <   10.    0%         2


** NUHBER OF ELEHENTS WORSE THAN TARGET VALUE  3


##############################################################################
 END OF HODULE GHS3D-INRIA
##############################################################################


-- WRITING THE OUTPUT FILES
   NUHBER OF GIVEN VERTICES   758
   NUHBER OF CREATED VERTICES 282
   TOTAL NUHBER OF VERTICES   1040
   NUHBER OF TETRAHEDRA       3751
   NUHBER OF SUB-DOHAINS      1
-- WRITING COHPLETED (  6  RECORDS ) bielle.noboiteb
-- WRITING COHPLETED (  2  RECORDS ) bielle.bb

   TOTAL CPU      .57000     SEC.
 394842  ELEHENTS ( 432807  ) WITHIN A HN.  FOR bielletest
```

8 Options and physical description

This section concerns the possible options and the way in which the physics can be associated with the resulting mesh.

8.1 Options

The mesh generation algorithm is fully automatic, no particular options or values are required to run the program.

Depending on the installation, the sole available options concern the printout rate and the selection of such or such type of output.

The printout rate (verbosity), namely $PRINT$, is defined as follows :

- PRINT = 0: Minimum rate,
- PRINT = 10: Reasonable rate (advised),
- PRINT = -10: Maximal rate (not advised, except for debugging purpose).

The output is defined by the so called $OPTION$ as follows:

- OPTION 0 : standard case, a *xxx.noboite* file is created as output,
- OPTION -1: a $NOPO$ file is created in addition ($NOPO$ is the mesh data structure of software package Modulef[4]),
- OPTION -2: (private) output before processing the optimization phase,
- OPTION -3: (private) output at the time the given points have been inserted, a *xxx.boite* file contains the mesh of an enclosing box of the domain,
- OPTION -4 and OPTION -5: cases where the domain includes several connected components (see below),
- OPTION -6: (private) output after the boundary regeneration phase, a *xxx.noboite* file is created,
- OPTION -7: (private) output after the boundary regeneration phase, a *xxx.boite* file is created.

8.2 Entering the physics

By itself, the mesh generation algorithm does not consider the physical attributes included in the *xxx.points* file as well as in the *xxx.faces* file. Thus, the *xxx.noboite* file does not contain these information. To obtain the latter, a post-processing is needed based on the fact that *the numbering of the initial points is not affected* by the mesh generation process. This post-processing is in charge of associating the physics with

[4] Available only on request.

the points, edges, faces and elements in the mesh in such a way as to make the desired computation possible.

As default option, it is assuming that the domain to be meshed is of one of the following types :

- a domain with *one* connected component (one material),
- a domain with one connected component but having *one or several* holes.

In these cases, the output file will contains exactly the domain so-defined. In the case where several connected components exist and such or such must be retained, $OPTION$ must be fixed as follows:

- OPTION -4: component number 1 is selected (the component defined as component number one is that immediately encountered when starting from the outside of the whole domain (in case of ambiguous situations, this option is not advised).
- OPTION -5: all the connected components are selected, in particular, possible holes and sub-domains are not considered as distinct[5].

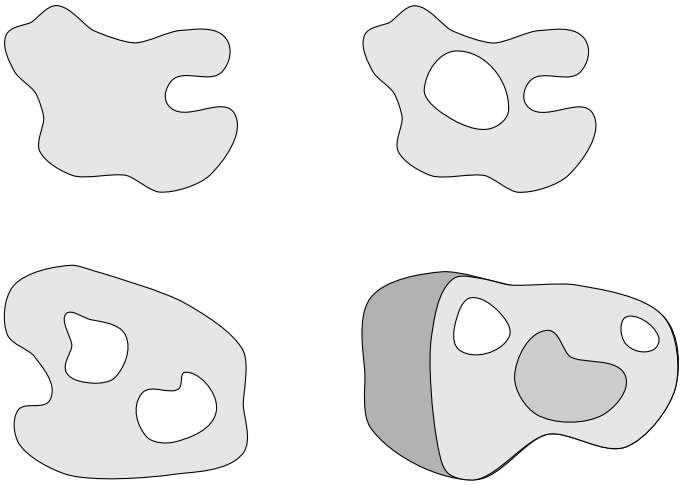


Figure 3: *From top to bottom, from left to right : i) one region, ii) one region including one hole, iii) one region with two holes and iv) three sub-domains and two holes.*

[5] An easy to write program is then necessary to identify the elements in such or such component, see also Section 4.2.

9 Diffusion

There are roughly two possibilities to obtain the GHS3D software package One entry point could be INRIA (in connection with Distène), the other is Distène, a commercial former subsidiary of INRIA.

9.1 From Distène

Distène (the former Simulog-Technologies) is in charge of licensing the mesh generation algorithm. Several possibilities are available including the license of the algorithm as itself or the license of SIMAIL, a powerful mesh generation software including the GHS3D algorithm as an option. Simulog-Technologies offers a series of services connected to the software including integration, portage, maintenance, hot line, etc.

- Distène S.A.S.
- Pôle Teratec - BARD-1, Domaine du Grand Rué
- 91680 Bruyères le Chatel
- FRANCE

Fax number : (33) 1 69 26 62 30 (attention Mark Loriot or Laurent Anné)
Fax number : 01 69 26 90 33 (attention Mark Loriot or Laurent Anné)
E-mail : mark.loriot@distene.com or Laurent.Anne@distene.com

9.2 From INRIA

INRIA (in connection with Distène) has the license of diffusing the GHS3D algorithm under some conditions. The address is as follows:

- Relations Industrielles
- INRIA, domaine de Voluceau
- Rocquencourt
- BP 105, 78153 Le Chesnay Cedex
- FRANCE

E-mail : paul-louis.george@inria.fr
Fax number : (33) 1 3963 5882 (attention Paul Louis George)
or could be : (33) 1 3963 5034 (attention Dominique Begis)

10 List of errors

Warnings, errors or causes of failure are identified and detailed in the following. Diagnostics are provided corresponding mainly to

- the correctness of the surface mesh serving as input data,
- the lack of computer memory,

- others.

The list of available diagnostics is as follows :

- ERR 0000 : The surface mesh includes a facet of type *type* other than edge, triangle or quadrilateral. Table *list* gives the vertices of this facet. This facet type is not supported. ** Subdivide this facet into an available type facet.
- ERR 0001 : Not enough memory allocation for the facet table. There are nf facets while the table is $nfmax$ long. ** Modify the memory resource allocation.
- ERR 0002 : Not enough memory, there are np points and $npmax$ points have been allocated. ** Modify the memory resource allocation.
- ERR 0003 : Not enough memory, there are ne elements and $nemax$ elements have been allocated. ** Modify the memory resource allocation.
- ERR 0004 : Facet number j with vertices *list* is not considered.
- ERR 0005 : end of file.
- ERR 0006 : failure when reading the file.
- ERR 0007 : the metric file is inadequate (dimension other than 3).
- ERR 0008 : the metric file is inadequate (values not per vertices).
- ERR 0009 : the metric file contains more than one field.
- ERR 0010 : the number of values in the metric file is incompatible with the expected value (the number of mesh vertices).
- ERR 0012, LIRNBO, $ndsd$, $ndsdmax$.
  Too much sub-domains (more than $ndsdmax$).
- ERR 0022 : incompatible data (see also ERR 3122).

- ERR 1000 : Facet $(f1, f2, f3)$ appears more than once in the input surface mesh (warning).
- ERR 1001 : Edge $(e1, e2)$ appears more than once in the input surface mesh (warning).

- ERR 2000 : Not enough available memory (stop).
- ERR 2002 : n initial points cannot be inserted (stop). ** The surface mesh is probably very bad in terms of quality or the input list of points is wrong (see also the following ERR).
- ERR 2003 : Vertex $v1$ and vertex $v2$ are too close or coincident (stop).
- ERR 2004 : Vertex $v1$ and vertex $v2$ are too close or coincident (warning).
- ERR 2012 : Vertex $v1$ cannot be inserted (warning).
- ERR 2014 : There is at least two points whose distance is *dist*, *i.e.,* judged coincident (stop).
- ERR 2103 : Vertex $v1$ and vertex $v2$ are too close or coincident (warning).

- ERR 3000 : The surface mesh regeneration step has failed. A *xxx.boite* file is created (stop). ** Try again with this file as input or modify (optimize, subdivide, etc.) the surface mesh.
- ERR 3009 : Constrained edge $(e1, e2)$ cannot be enforced (warning).
- ERR 3019 : Constrained facet $(f1, f2, f3)$ cannot be enforced (warning).
- ERR 3029 : Number of missing facets (warning).
- ERR 3100 : No guess to start the definition of the connected component(s) (stop).
- ERR 3101 : The surface mesh includes at least one hole. The domain is not well defined (warning). ** Check your surface mesh.
- ERR 3102 : Impossible to define a component (stop). Are respectively reported the number of external tet, internal tet, unclassified tet and the number of connected components.
- ERR 3103 : The surface edge $(e1, e2)$ intersects another surface edge $(e3, e4)$ (stop). ** Check your surface mesh.
- ERR 3104 : The surface edge $(e1, e2)$ intersects the surface facet $(f1, f2, f3)$ (stop). ** Check your surface mesh.
- ERR 3105 : One boundary point (say $p1$) lies within a surface facet $(f1, f2, f3)$ (stop). ** Check your surface mesh.
- ERR 3106 : One surface edge (say $e1, e2$) intersects a surface facet $(f1, f2, f3)$ (stop). ** Check your surface mesh.
- ERR 3107 : One boundary point (say $p1$) lies within a surface edge $(e1, e2)$ (stop). ** Check your surface mesh.
- ERR 3108 : Insufficient memory resources detected due to a bad quality surface mesh leading to too much swaps. (stop). ** Check the surface mesh. Improve its quality and/or allocate more memory size.
- ERR 3109 : Edge $(e1, e2)$ is unique (i.e., bounds a hole in the surface) (warning).
- ERR 3122 : Presumably, the surface mesh is not compatible with the domain under treatment (warning). ** Check that you are doing what you really think.
- ERR 3123 : the number of sub-domains (materials) exceeds the allocated limit (1024).
- ERR 3209 : The surface mesh includes at least one hole. Thus there is no domain properly defined (stop). ** Check the surface mesh and suppress the hole(s).
- ERR 3300 : Statistics.
- ERR 3400 : Statistics.
- ERR 3500 : Warning, it is dramatically tedious to enforce the boundary items (warning). ** Check the surface mesh. Improve its quality.

- ERR 4000 : Not enough memory at this time, n evertheless, the program continues (warning). ** The expected mesh will be correct but not really as large as required.
- ERR 4002 : see above error code.

- ERR 8000 : Unknown facet type.

- ERR 8001, $f1, f2, f3$.

  Facet $(f1, f2, f3)$ appears more than once in the surface mesh.

- ERR 8002, $e1, e2$.

  Edge $(e1, e2)$ appears more than once in the surface mesh.

- ERR 9000 : A too small volume element is detected. Are reported the index of the element, its four vertex indices, its volume and the tolerance threshold value (warning).
- ERR 9001 : There exist at least a null or negative volume element. The resulting mesh will be inappropriate (warning).
- ERR 9002 : There exist n null or negative volume element. The resulting mesh will be inappropriate (warning).
- ERR 9003 : A too small volume element is detected. A facet is judged degenerated, its minimum edge length is reported (warning).
- ERR 9100 : see above ERR 9000.
- ERR 9101, t, $list$, Q.

  The aspect ratio of element t with vertices $list$ is Q. This element is suspected to be very bad shaped or wrong.

- ERR 9102 : A too bad quality facet is detected. This facet is judged degenerated, its index, its three vertex indices together with its quality value are reported (warning).
- ERR 9112 : A too bad quality facet is detected. This facet is judged degenerated, its index, its three vertex indices together with its inradius are reported (warning).
- ERR 9122 : see above ERR 3122.
- ERR 9999 : Bug symptom, contact the hot-line.

11 Coming soon

Right now, the GHS3D software is widely used in various types of calculations (automotive, aerospace, mechanical engineering, etc.) by numerous end-users. It is also included in some major commercial software packages. This allows for concrete experiences and ideas of improvement in all aspects of the program.

Adaptive versions are in development including the so-called GAMHIC3D software that concerns the construction of isotropic adaptive tet meshes based on a metric size (defined by a background mesh). Anisotropic problems are also considered and an anisotropic version (unnamed at this time) is expected in the coming future.

References

[1] P.L. George, F. Hecht, E. Saltel, Fully automatic mesh generator for 3d domains of any shape, *Impact of Comp. in Sci. and Eng.*, 2, pp 187-218, 1990.

[2] P.L. George, *Génération automatique de maillages. Applications aux méthodes d'éléments finis*, RMA 16, Masson, Paris, 1991. Also as *Automatic mesh generation. Applications to finite element methods*, Wiley, 1991.

[3] P.L. George, F. Hecht, E. Saltel, Automatic mesh generator with specified boundary, *Comp. Meth. in Appl. Mech. and Eng.*, vol 92, pp. 269-288, 1991.

[4] P.L. George, F. Hecht, M. G. Vallet, Creation of internal points in Voronoi's type method, Control and adaptation, *Adv. in Eng. Soft.*, 13, n° 5/6, pp. 303-313, 1991.

[5] P.L. George, F. Hermeline, Delaunay's mesh of a convex polyhedron in dimension d. Application to arbitrary polyhedra, *Int. Jour. Num. Meth. Eng.*, vol 33, pp. 975-995, 1992.

[6] E. Briere de l'Isle, P.L. George, Optimization of tetrahedral meshes, *IMA Volumes in Mathematics and its Applications*, ed. I. Babuska, W.D. Henshaw, J.E. Oliger, J.E. Flaherty, J.E. Hopcroft and T. Tezduyar, vol 75, pp. 97-128, 1995.

[7] H. Borouchaki, F. Hecht, E. Saltel, P.L. George, Reasonably efficient Delaunay based mesh generator in 3 dimensions, *in 4th International Meshing Roundtable, Albuquerque, New Mexico*, pp. 3-14, 1995.

[8] P.L. George, Improvement on Delaunay based 3D automatic mesh generator, *Finite Elements in Analysis and Design*, vol 25(3-4), pp. 297-317, 1996.

[9] P.L. George et H. Borouchaki, *Triangulation de Delaunay et maillage. Applications aux éléments finis*, Hermès, Paris, 1997. In english, *Delaunay triangulation and meshing. Applications to Finite Elements*, Hermès, 1998.

[10] P.J. Frey et P.L. George, *Maillages. Applications aux éléments finis*, Hermès, Paris, 1999. In english, *Mesh Generation*, Hermès Science Publication, 2000.