MeshGems

MG-Cleaner

The automatic mesh fixing tool

of the

MeshGems® Suite

User Manual

MG-Cleaner Version: 1.0

Documentation Revision: 1.0

June 2012

Customer Service

Should you encounter any problem with MG-Cleaner or the MeshGems® Suite, please contact the customer service by:

Address: DISTENE S.A.S.
Customer Support Service
Campus Teratec
2, rue de la Piquetterie
91680 Bruyeres-le-Chatel
FRANCE

Telephone: +33(0)164-908-596
Fax: +33(0)169-269-033
Electronic mail: support@distene.com

1st edition, June 2012

Information

This document is the user manual of the MG-Cleaner software. It applies to all platforms running the software.

The MG-Cleaner software is the automatic mesh fixing tool.

> **Important notice**: In order to use MG-Cleaner, you must first install the tool and, if necessary, the license server. Please refer to the installation guide in the delivered CD-ROM or in the installed distribution.

Document History

Version	Date	Comment
1.0	June 2012	1st edition

Foreword

The MeshGems® Suite is a comprehensive set of meshing components, which provides CAD/CAE application developers with reliable, fast and high quality meshing technologies. These components address all aspects of automatic meshing for real life industrial 3D numerical simulations, using very efficient algorithms:

- surface meshing from CAD,
- correction of data, whether geometrical or triangulated,
- volume mesh generation,
- remeshing,
- adaption driven by user-given local criteria.

The MeshGems® Suite is designed to make integration and maintenance easy by third party developers. The MeshGems® Suite today includes:

- **MG-Tetra** for tetrahedral volume mesh generation
- **MG-CADSurf** for triangular/quadrilateral mesh generation
- **MG-PreCAD** and **MG-Cleaner** for CAD and mesh correction respectively

and will soon include:

- **MG-Hexa** for hexahedral volume mesh generation
- **MG-SurfOpt** for surface remeshing and optimization
- **MG-Adapt** for surface and/or volume mesh adaption

All these components can be used independently or as a compatible whole.

See section 1.5 for more information about coupling MG-Cleaner with the other MeshGems® Suite components and section 3.3 for more information about MG-Cleaner relationship with the physics and/or the CAD description.

Contents

Introduction

This user manual describes the main features of MG-Cleaner[1], the automatic mesh fixing tool.

MG-Cleaner is an automatic meshing component designed to correct automatically surface meshes, in order to make them suitable for volume meshing. MG-Cleaner aims at generating valid surface meshes out of faulty ones, and being automatic, does not require any interactive manipulations from the user.

MG-Cleaner will either make an analysis of the surface mesh and output information highlighting possible faults, or make this analysis and proceed with corrective actions accordingly.

Typical problems detected and corrected include, but are not limited to:

- Overlaps
- Non conformities
- Holes or gaps preventing water-tightness
- Self-intersections
- Bad shaped elements
- Incorrect assemblies, up to a certain level of complexity.

MG-Cleaner's behaviour and precision can be controlled with several user-tunable runtime parameters.

MG-Cleaner is simply run as a command line executable with online options.

The input and output files, the interaction with the user, and the list of error messages are described in this document. Several typical application examples are provided to explain the various possibilities of the code. Several appendixes describe thoroughly the input/output formats and other useful information.

[1] Developed by M.Bory and Distene SAS

Chapter 1

Getting started

1.1 Installation

Please refer to the installation guide which describes MeshGems® Suite component installation and its use. This guide is available both from the delivered media and from the distribution once installed under "`.../DISTENE/Docs/`"; in particular you may want to check out the section "Use of a Distene Tool" located in file "`05-use-tool.html`".

1.2 Requirements

The program is entirely written in C++. MG-Cleaner is currently available on 32-bit and 64-bit Windows and Linux architectures.

The executables have been designed to have minimal dependencies towards the Operating System.

1.2.1 Disk Space

The disk space requirement for this tool installation is about 30 MB.

Content	**Size (MB*)**
Common part for all modules (installation documentation, scripts)	0.1
Common part for MG-Cleaner (documentation, scripts, examples)	2.7
Linux 32bit MG-Cleaner	7.4
Linux 64bit MG-Cleaner	9.2
Windows 32bit MG-Cleaner	5.6
Windows 64bit MG-Cleaner	6.1

* MB=1,000,000 bytes

1.2.2 CPU time

The CPU time required to obtain the final mesh (including the I/O) is related to the parameters and options specified. Nevertheless, one can reasonably expect a few minutes to process a mesh with several hundred thousand elements.

1.2.3 Memory usage

One can reasonably expect a memory usage about one million triangles per 250 Mb.

1.3 Launching the software

Once the installation variables[1] and the execution path are set, MG-Cleaner can be called by typing the command "mg-cleaner.sh" on Unix platforms and "mg-cleaner.bat" on Windows platforms.

If the command exits with a message such as the following:

```
Error: License not available for MG-CLEANER version 1.0
```

it means that one or more of the following is true:

1. the environment variables are incorrectly set;
2. the license is obsolete or invalid;
3. the license server is incorrectly installed.

The use of MG-Cleaner and the other MeshGems® Suite components requires various license keys, depending on the activated features.

1.4 Errors and diagnostics

MG-Cleaner has its own advanced error management. It intercepts them and prints them on screen.

1.5 Coupling with other MeshGems® Suite components

MG-Cleaner can be used efficiently with the following companion tools: MG-Tetra, MG-SurfOpt, MG-Adapt, MG-CADSurf.

MG-Cleaner has been initially designed as a **preprocessor to make surface meshes suitable for MG-Tetra,** that is water-tight and not self-intersecting.

MG-Cleaner may also be used as a **postprocessor of MG-CADSurf or MG-SurfOpt.** Indeed, depending on the quality and characteristics of their input, MeshGems® Suite MG-CADSurf or MG-SurfOpt may produce such invalid meshes for MG-Tetra.

MG-Cleaner can also be used as a **preprocessor for MG-SurfOpt or MG-Adapt.** Both try to keep all the geometrical structures they detect in their input. Because MG-Cleaner improves the surface mesh quality and gets rid of unwanted small structures, numerical errors are avoided and confusing or unwanted small structures count is reduced.

1.6 Integration within third party products

MG-Cleaner is delivered as an executable and meshes come in and out of MG-Cleaner as files.

Contact Distene SAS if you wish to embed more tightly MG-Cleaner in your own software.

[1] The variable "DISTENE_LICENSE_FILE" is usually set through a script.

Chapter 2

MG-Cleaner principles

MG-Cleaner scans the input mesh and detects several types of misfeatures such as overlaps, intersections, holes or gaps, ...

Once detected, it applies several corrective strategies in sequence and stops as soon as the problem has been corrected. If none of the strategies is successful, MG-Cleaner proceeds with the other errors and will come back to this one again later.

Chapter 3

MG-Cleaner utilisation

Contents

3.1 Input Files

To describe the input surface mesh, the ".mesh" file format is the main format used with MeshGems® Suite. A ".mesh" file can be written either in ASCII or binary format.

Limitation: However, only the ASCII version of the ".mesh" format is currently usable with MG-Cleaner. For the time being MG-Cleaner also reads the deprecated ASCII format "MSH2".

The several types of files used by MG-Cleaner are briefly described below, and detailed in appendix B, 'File format description' on page 35.

The input surface mesh must be formed with triangles or quadrilaterals only. In the case of quadrilaterals, these will be momentarily converted to triangles to be able to be processed with MG-Cleaner, but they will be converted back to the original quadrilaterals if they were left untouched by the cleaning process. Also, the attributes of the surface will be kept in the fixed mesh.

3.1.1 The mesh format

This format is composed of a single file, `mybasename.mesh` (ASCII) or `mybasename.meshb` (binary). This file contains the basic information needed to describe entirely the surface mesh. It is organized as a series of fields, identified by keywords. In addition to the vertex coordinates and the topology represented by a list of faces, it allows one to specify additional information such as faults detected in the mesh.

Refer to appendix B, 'File format description' on page 35 for a detailed presentation of this format.

Limitation: Please note that MG-Cleaner does not support `.meshb` binary files yet.

3.1.2 The MSH2 format

This format is composed of two files, `mybasename.faces` and `mybasename.points` (both ASCII). This format was the original native format of MG-Tetra and is now deprecated.

The `mybasename.points` file contains the coordinates of the surface mesh vertices and the `mybasename.faces` file contains the list of triangles of the surface mesh, described by their vertices.

Refer to appendix B, 'File format description' on page 35 for a detailed presentation of this format.

3.2 Output Files

A single output file is produced at completion. This file, corresponding to the resulting surface mesh, is written using the `.mesh` format.

The resulting mesh will be composed of triangles and untouched quadrilaterals only, and it may contain additional keywords depending on the runtime option selected.

Unless option `--check` has been used, this resulting mesh will have as many faults corrected as possible. It will conform to the given surface within the allowance given by the user or, if this tolerance has been left up to the program, computed automatically from the input data.

3.3 Relationship with the physics and/or the CAD description

Each entity (vertex, edge, face, tetrahedron, ...) carries an integer (often called «attribute» or «colour») that, depending on the runtime option selected, MeshGems® Suite components preserve.

One defines an attribute integer to identify:

- a physical property such as material, boundary condition, ...
- a CAD group affiliation such as CAD patch, construction line, ...
- or a unique combination of properties.

These attributes are set in view of a particular treatment and entities with the same attribute are usually processed the same way on the client side usually but sometimes on MeshGems® Suite components side as well.

The user must simply set these attributes to suit his needs such that:

- two entities belonging to two different CAD patches must have two different attributes if one needs to keep track of which CAD patch entities belong to;
- two entities belonging to two different physical property sets must have two different attributes if one needs to keep track of which physical property set entities belong to;

This may need a pre/post treatment of the user data.

Advice: when possible, save CAD construction lines (surface normal discontinuities as ridges and surface curve changes as coloured edge lines) in the input mesh file as this helps rebuilding the geometrical information.

For example attribute numbers on vertices can be used:

- to associate a certain number of variables to apply boundary conditions,
- to analyse the results corresponding to a particular attribute.

MG-Cleaner preserves the attributes specified in the input file (vertex, edge and face attributes) unless the entity underwent a corrective operation.

Limitation: In a next version, a history of attribute modifications will be maintained that will help the user keep track even after corrective operations took place.

As can be easily understood, the numbering of the entities is affected by MG-Cleaner algorithms.

3.4 Invoking MG-Cleaner and using Options

This section describes how to run MG-Cleaner and the possible options and parameters needed to control the meshing process.

3.4.1 Running MG-Cleaner

The usual way to start MG-Cleaner is to type in the following line:

```
mg-cleaner --in ⟨filein⟩ [⟨extra options⟩]
```

where:

- the parts between angle brackets are ⟨set by the user⟩;
- the parts between square brackets are [optional].

Moreover, the order in which the options are given may be altered at user's will, that is the following line will work as well:

```
mg-cleaner [⟨extra options 1⟩] --in ⟨filein⟩ [⟨extra options 2⟩]
```

More precisely,
in the case of `.mesh` files, the syntax is the following:

```
mg-cleaner.exe --in ⟨my_case⟩.mesh [--out ⟨my_clean_case⟩.mesh] [⟨extra options⟩]
```

in the case of `MSH2` files, the syntax is the following:

```
mg-cleaner.exe --in ⟨my_case⟩.faces [--out ⟨my_clean_case⟩.mesh] [⟨extra options⟩]
```

Note: The use of the `.mesh` format rather than the `MSH2` format is highly recommended, as the `MSH2` format is deprecated and may become unsupported in future releases.

Although the program is mostly automatic, it requires a few options. The default options were chosen so as to comply with most of the realistic models.

3.4.2 Generic options

3.4.2.1 Choosing the input file: "`--in` ⟨file⟩" (MANDATORY)

This option tells to the mesher the name of the input surface mesh, which should be in the `.mesh` ASCII format.

3.4.2.2 Setting the verbosity level: "`--verbose` ⟨level⟩"

This option (0 <= ⟨level⟩ <= 10) prints information on meshing steps. Default level is 3.

This allows the user to increase (higher ⟨level⟩ value) or decrease (lower ⟨level⟩ value) the amount of information that appears on the screen while MG-Cleaner processes the mesh.

3.4.2.3 Setting the output file name: "`--out` ⟨file⟩"

This option tells to the mesher the name of the output mesh, which will be in the `.mesh` ASCII format.

By default, the name of the output mesh will be the name of the input file with `_cleaner` inserted before the suffix.

3.4.2.4 Getting online help: "--help"

```
    ========================================
    MG-Cleaner -- MeshGems 1.0-0 (June, 2012)
    ========================================

        Distene SAS
           Campus Teratec
           2, rue de la Piquetterie
           91680 Bruyeres le Chatel
           FRANCE
        Phone: +33(0)164-908-596     Fax: +33(0)169-269-033
        EMail: <support@distene.com>

        Running MG-Cleaner (Copyright 2004-2012 by Distene SAS)
           date of run: 22-Jun-2012 AT 10:24:23
           running on : Linux 3.1.10-1.9-desktop x86_64

        MeshGems and MG-Cleaner are Registered Trademarks of Distene SAS



MG-Cleaner USAGE

Synopsis:
    ../../bin/Linux_64/MeshCleaner-DLIM-1.0-0.exe [--help] [--in <filein>] \
        [--out <fileout>] [--check] [--fix2pass] [--fix1pass] \
        [--topology <behaviour>] [--min_hole_size <size>] \
        [--tolerance_displacement <size>] [--resolution_length <size>] \
        [--folding_angle <angle>] [--remesh_planes] \
        [--overlap_distance <size>] [--overlap_angle <angle>] \
        [--verbose <verblevel>]

Description:
     Short option (if it exists)
    /    Long option
   |    /   Description
   |   |   /
   v   v  v

     --help
          gives this help.

  -i --in <filein>
          sets the input file.
          (MANDATORY)

  -o --out <fileout>
          sets the output file.

  -c --check
          performs checks only (no fixing).
          Writes diagnostics into the output file.
          Default is to fix with two passes.

  -R --fix2pass
          analyses and fixes mesh with the two stage cleaning procedure.
          Does not write diagnostics into the output file.
          Default is to fix with two passes.

  -f --fix1pass
          analyses and fixes mesh with only the first stage of the cleaning
          procedure.
          Does not write diagnostics into the output file.
          Default is to fix with two passes.

     --topology <behaviour>
          sets the applicable fixing operations.
            'ignore' : applies all fixing operations
            'respect': disables fixing operations which induce topology
                       modifications
```

```
          Default is 'ignore'.

  -h --min_hole_size <size>
          sets the surface size threshold below which holes are filled.
          Default is not to fill holes.

  -t --tolerance_displacement <size>
          sets the displacement threshold below which modification is allowed.
          Unused in collision resolution.
          tolerance_displacement is set to resolution_length if it is lower.
          Default is computed from model.

  -s --resolution_length <size>
          sets the distance threshold above which 2 points are considered
          distinct.
          Sets the tolerance_displacement to 1/5 of this size.
          Default is computed from model.

  -a --folding_angle <angle>
          sets the threshold angle below which 2 connected triangles are
          considered overlapping.
          Reduce this value if model contains sharp angles below this threshold
          that must be kept.
          overlap_angle is set to this angle if it is higher.
          Default is 15 degrees.

  -p --remesh_planes
          inserts vertices on planes to improve mesh quality.
          May be useful for poor quality triangulations (eg. STL or DXF
          triangulations).
          Default is not to mesh planes.

  -d --overlap_distance <size>
          sets the distance below which 2 unconnected triangles are considered
          overlapping.
          Reduce this value if too many overlaps are detected.
          Default is computed from model.

  -A --overlap_angle <angle>
          sets the angle below which 2 unconnected triangles are considered
          overlapping.
          folding_angle is set to this angle if it is lower.
          Default is 15 degrees.

  -v --verbose <verblevel>
          sets the verbosity level.
          From 0 (no detail) to 10 (very detailed).
          Default is 3.


================================================================================
                  MG-Cleaner -- MeshGems 1.0-0 (June, 2012)
       END OF SESSION - MG-Cleaner (Copyright 2004-2012 by Distene SAS)
     compiled Jun 19 2012 17:58:29 on Linux 2.6.27.56-0.1-default x86_64
       MeshGems and MG-Cleaner are Registered Trademarks of Distene SAS
================================================================================
      ( Distene SAS
        Phone: +33(0)164-908-596     Fax: +33(0)169-269-033
        EMail: <support@distene.com> )
```

3.4.3 Module specific options and control parameters

3.4.3.1 Checking the mesh: "--check" (default is to fix with two passes)

This option will deactivate the mesh fixing, and will only perform detection of potential faults.

The output mesh will be written in the .mesh ASCII format.

The output mesh will contain the data read from the input mesh and, for each class of error, a new section containing the list of triangles potentially faulty or in an area of problem.

3.4.3.2 Fixing the mesh with two passes: "--fix2pass" (default)

This option analyses and fixes the mesh with a two stage cleaning procedure. The first stage is identical to the one pass fixing procedure below. However, as the second stage applies different strategies, it should improve the mesh more than the first stage procedure alone.

The output fixed mesh will be in the .mesh ASCII format. There will be no added sections.

3.4.3.3 Fixing the mesh with one pass: "--fix1pass" (default is to fix with two passes)

This option analyses and fixes mesh with only the first stage of the cleaning procedure.

The output fixed mesh will be in the .mesh ASCII format. There will be no added sections.

3.4.3.4 Preserving topology: "--topology ⟨behaviour⟩" (default is to modify topology)

This option may be used to prevent the mesh cleaning step from performing global corrections who are deemed to induce a topology change.

This option handles the following values of ⟨behaviour⟩ parameter:
ignore: allows changes to the topology when necessary for corrections and is the default value;
respect: disables corrections implying a topology change (only the resolution_length is kept in effect).

This option reduces considerably the spectrum of corrections that can be applied, so should be used with care.

Important note: even though this option aims at reducing significantly the topology changes, it does not guarantee that the topology will always be preserved. This can happen on some occasions when local corrections induce a local topology change.

3.4.3.5 Setting hole min size: "--min_hole_size ⟨size⟩" (default is to not fill holes)

This option activates automatic hole filling. If this option is not used, no holes will be filled. The size parameter gives the threshold above which the holes will not be filled.

Important note: only holes for which their contour can be held in a plane can be filled with this option.

3.4.3.6 Setting tolerance parameter: "--tolerance_displacement ⟨size⟩" (default from the input surface)

The tolerance_displacement parameter specifies the maximum size of modification allowed. That is, no modification will be made if it implies a vertex displacement exceeding this value, unless this is to correct a collision (a vertex crossing a neighbouring surface).

It should be used with great care to avoid destroying the geometry.

The default value for tolerance_displacement parameter is computed from the input surface.

3.4.3.7 Setting resolution length: "--resolution_length ⟨size⟩" (default from the input surface)

This option sets the minimal distance between two vertices, or between one vertex and an edge. If two vertices are closer than this distance, they will be considered the same vertex and will be "glued" together. If one vertex is closer than this distance of an edge, the vertex will be considered part of the edge which will be split.

Important note: the tolerance induced by this option will take precedence over any parameter set by the user for the tolerance.
This implies that activating this parameter will set the tolerance value to a 1/5th of this new resolution size, if this new tolerance is smaller or equal to the default tolerance. This is necessary to prevent the tolerance from being too large compared to the resolution size.

The default value for resolution_length parameter is computed from the input surface.

3.4.3.8 Setting minimum angle: "--folding_angle ⟨size⟩" (default is 15 degrees)

This option sets the threshold angle below which two *connected* triangles will be considered as being overlapping each other.

See also options --overlap_distance and --overlap_angle.

The default value is set to 15 degrees.

3.4.3.9 Activating the remeshing of planar surface: "--remesh_planes" (default is to not remesh planes)

This option will activate remeshing of planar surfaces. This is only useful when dealing with poorly discretized surfaces where the number of elements was reduced very significantly (such as STL geometries). This option will generate more and better quality triangles on these automatically detected planes.

This option is not activated by default.

3.4.3.10 Setting the overlapping distance: "--overlap_distance ⟨size⟩" (default from the input surface)

This option gives another control on overlap detection. This sets the minimal distance between two *unconnected* triangles below which they will be considered as being overlapping.

You may reduce this value, depending on the input surface, if too many overlaps are detected.

The default value for overlap_distance parameter is computed from the input surface.

See also option --folding_angle.

3.4.3.11 Setting the overlapping angle: "--overlap_angle ⟨angle⟩" (default is 15 degrees)

This option sets the threshold angle below which two *unconnected* triangles will be considered as being overlapping each other.

See also options --overlap_distance and --folding_angle.

The default value is set to 15 degrees.

3.4.4 Using MG-Cleaner with its default parameter values

By default, the sole option that needs to be specified is the name of input data file:

```
mg-cleaner --in (filein)
```

3.5 Detection of faults

This section describes the potential faults detected and saved by MG-Cleaner when used together with the `--check` option.

MG-Cleaner optionally performs checks on the surface mesh to detect potential problems. The resulting areas are gathered as sets of triangles grouped by fault category, and are added to the surface mesh.

The produced mesh file contains – along with the data read from the input mesh – new keywords and mesh fields (refer to appendix B), which can be used by the embedding environment to visualize these potential faults.

Note that some problems may be highlighted by several different ways, so the same triangles may appear in several categories (eg, a hanging triangle on the surface may appear as "overlap", "near tri" and "free edge").

The categories of faults are:

SmallTri This highlights very small sized triangles. These may induce hard configurations for tet meshers, or very small simulation time steps.

BadShape This highlights narrow triangles. As for small triangles, these may induce hard configurations for tet meshers, and both very small simulation time steps and precision issues.

Overlap This highlights triangles which overlap. This particularly detects similar triangles on top of one another. General overlap areas are also detected by **NearTri** below.

FreeEdge This highlights free edges. This may not always be a real fault (in case of hanging surfaces for example), but there should not be any free edges on an outer surface.

Inter This highlights intersections.

NearTri This highlights triangles which are very close without being connected. This may also be an artifact of overlapping areas.

Chapter 4

MG-Cleaner examples of use

4.1 Example 1: Cleaning to get a volume mesh

This section gives a typical example of what MG-Cleaner can achieve. This is what is obtained on the case7 test case, provided in the installation.
The case7 test case uses the mesh format.

4.1.1 Mesh checking

Copy the file "case7.mesh" provided in the Examples directory of the distribution to your local work directory, then type:

```
mg-cleaner.exe --in case7.mesh --out case7-test.mesh --check
```

to perform some checks on the surface mesh and detect potential problems. The resulting areas are gathered as a set of triangles and are added to the surface mesh.
The produced .mesh file indeed contains new keywords and mesh fields, which can be used by the embedding environment to visualize these potential faults.

The two meshes, the initial surface mesh, and the mesh produced are shown in the pictures below (the embedding environment used here is Ensight, by CEI).

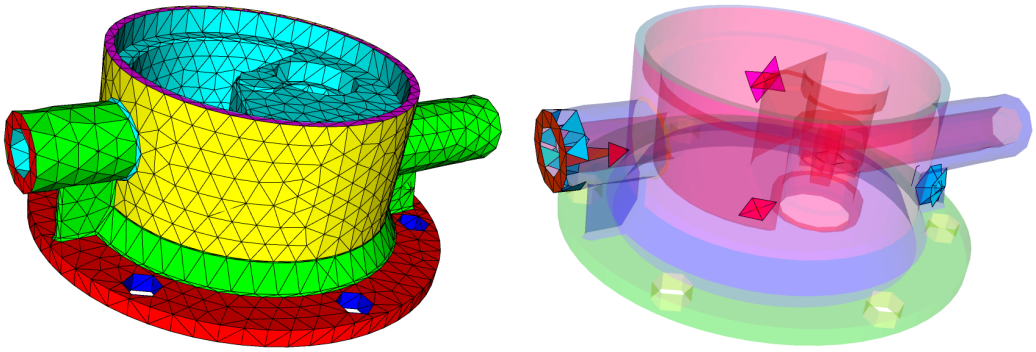


Figure 4.1: Original surface mesh (left) and "Checked" mesh (right)

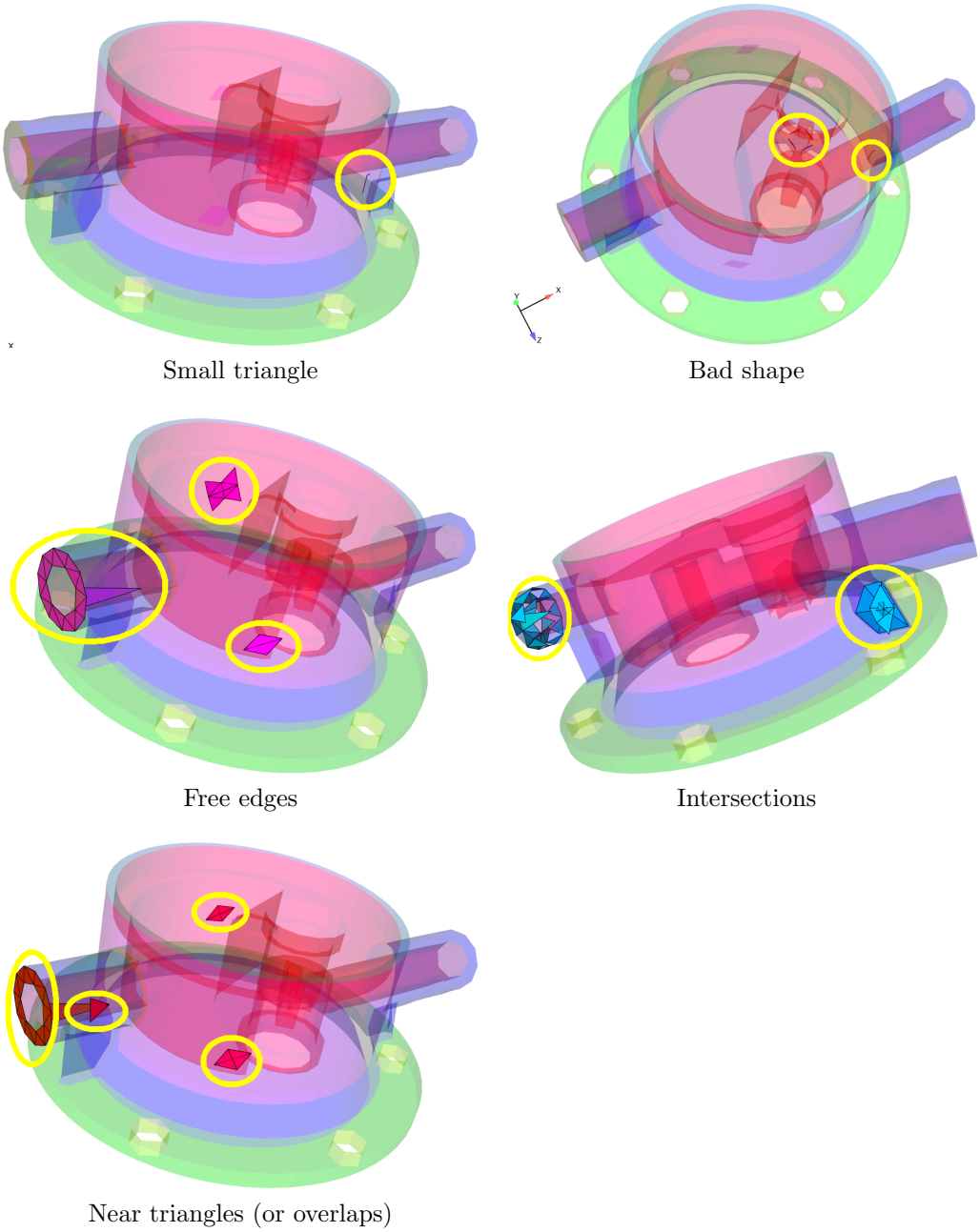


Figure 4.2: Detected faults

This run will produce the following output in the console window:

```
    ========================================
    MG-Cleaner -- MeshGems 1.0-0 (June, 2012)
    ========================================

        Distene SAS
           Campus Teratec
           2, rue de la Piquetterie
           91680 Bruyeres le Chatel
           FRANCE
        Phone: +33(0)164-908-596     Fax: +33(0)169-269-033
        EMail: <support@distene.com>

        Running MG-Cleaner (Copyright 2004-2012 by Distene SAS)
           date of run: 06-Jul-2012 AT 12:34:22
           running on : Linux 3.1.10-1.16-desktop x86_64

        MeshGems and MG-Cleaner are Registered Trademarks of Distene SAS

info: dlim, getting token info from <192.168.2.1>

 ************* MG-Cleaner --in case7.mesh --out case7-test.mesh --check
     *****************

 *** INPUT FILE    case7.mesh   ***

  Cleaning parameters: sizeMin  0.500000  tolGeom 0.100000 sizeHole 0.000000
      SewingSize 1.000000
                       ovAngles 15.000000 15.000000 ovDist 0.500000

  nb_Tri: 4475  smallEdge average size 18.675639 smallest size 0.098646
  colliding triangles   74  near triangles 139
   Worst ratio  228.356140 badTri 2177

  nb FreeContours: 5  nb Overlaps 0 nb SmallSize 2  nb BadRatio 6

 number badTri tangent and secant  137 16

 **** CPU time  :         0.7

*******************************************************************************

===============================================================================
                  MG-Cleaner -- MeshGems 1.0-0 (June, 2012)
       END OF SESSION - MG-Cleaner (Copyright 2004-2012 by Distene SAS)
     compiled Jul  5 2012 17:42:14 on Linux 2.6.27.56-0.1-default x86_64
       MeshGems and MG-Cleaner are Registered Trademarks of Distene SAS
===============================================================================
      ( Distene SAS
        Phone: +33(0)164-908-596     Fax: +33(0)169-269-033
        EMail: <support@distene.com> )
```

and the following file in the directory: "case7-test.mesh".

Let's see all detected faults one by one (see figure 4.2):

1. Small tri : there are very small elements in the mesh.
2. Bad shape : several bad shaped elements are detected.
3. Free edges: a surface is overlapping another surface on an inlet, and is detected as free edge. At the same time, some overlapping triangles appear on the main body.
4. Intersections : the inlet is detected as intersection because some vertices of the overlapping triangles are very close to other triangles. At another location (on the right), a point went through the surface to emerge on the other side.
5. Near tri : overlaps are also detected because some triangles are very close to others, without having more than one point in common.

4.1.2 Correcting the mesh with standard options

Using the same case as above, type this time :

```
mg-cleaner.exe case7.mesh case7-fix.mesh --fix
```

This will produce a corrected mesh, without the additional analysis keywords. Note that the mesh attributes are preserved.

The two meshes, the initial surface mesh, and the corrected mesh produced are shown in the pictures below:

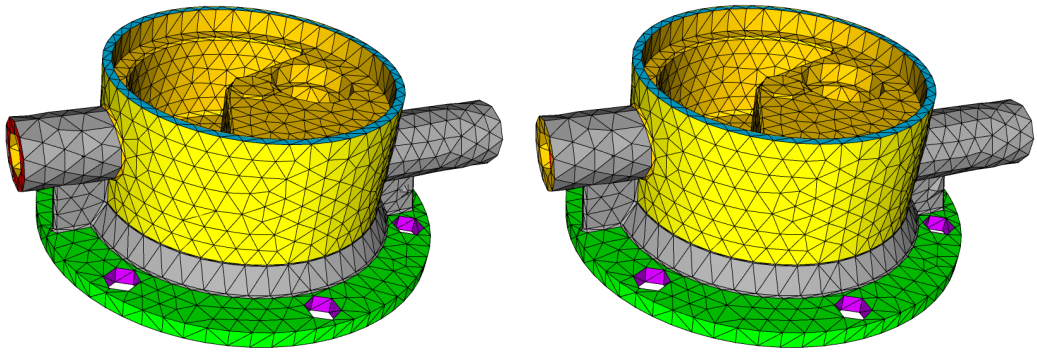
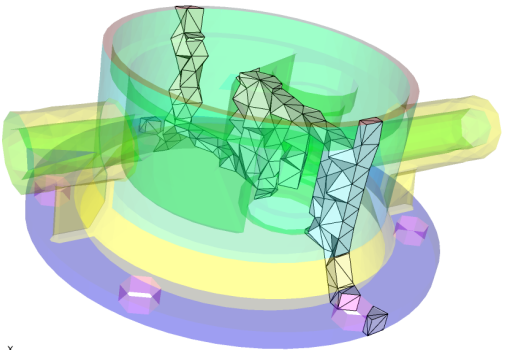


Figure 4.3: Original surface mesh (left) and corrected mesh (right)

We are now able to launch the tet mesher MG-Tetra to produce a tetrahedral mesh:

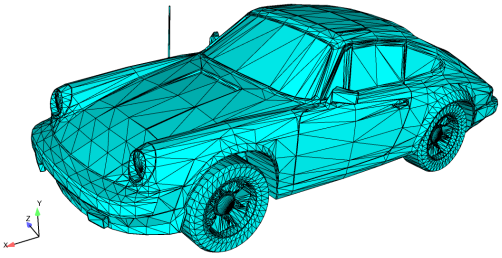


4.2 Example 2: Cleaning to remesh and get a volume mesh

This section gives another example of what MG-Cleaner can do. We will use it as a correction tool prior to performing surface remeshing, then eventually volume meshing. We will also use it as a means to improve the quality of the surface mesh.

4.2.1 Mesh checking

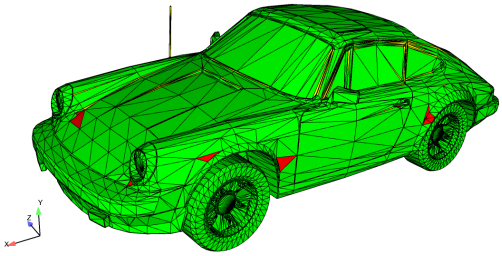
The mesh represents the geometry of a car, generated by a tessellation program, and contains many errors.



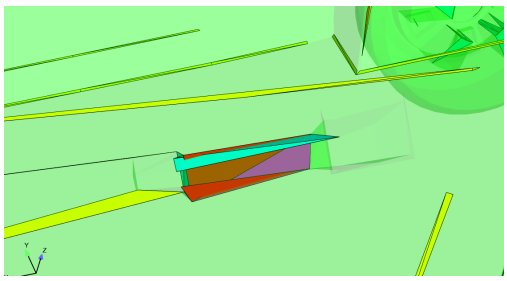
We will first start with the error checking:

```
mg-cleaner.exe --in Porsche.mesh --out Porsche-test.mesh --check
```

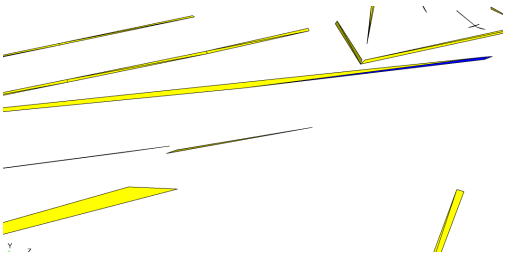
This is what we get:



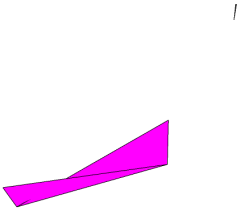
Let's analyze what is happening near the door handle on the front left of the car:



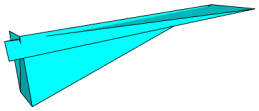
There are several problems. We will now visualize each in turn, at the same location (left door handle).



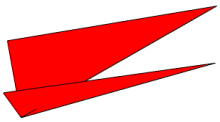
Fault_SmallTri (in blue) and Fault_BadShape (in yellow)



Fault_Overlap (purple)



Fault_Inter (blue)



Fault_NearTri (red)

1. badly shaped elements (very narrow triangles) or very small triangles. This is highlighted with the categories Fault_SmallTri (in blue) and Fault_BadShape (in yellow).

2. overlapping triangles. This happens when two or more triangles are overlapping, with shared vertex connections . This is highlighted with category Fault_Overlap (purple) as shown below. We see one triangle on top of one another.
   The overlapping detection has here a default threshold value of 15 degrees.

3. Intersections : This happens when two or more triangles intersect. This is highlighted with category Fault_Inter (blue) as shown below. We see two triangles intersecting each other.

4. Triangles too close : This happens when one vertex of a triangle is almost contained in another triangle. This is highlighted with category Fault_NearTri (red) as shown below. We see two triangles too close to one another (to the point of intersection).

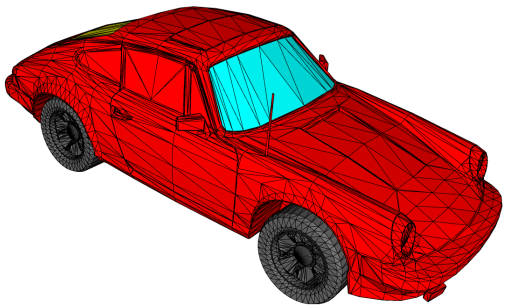
4.2.2 Fixing the mesh

Let's now try to correct the mesh above, typing this time :

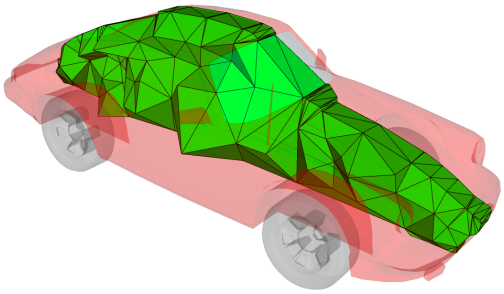
```
mg-cleaner.exe --in Porsche.mesh --out Porschefix.mesh --fix
```

This will produce a corrected mesh, without the additional analysis keywords. Noe that the mesh attributes are preserved.

Here is the corrected mesh:



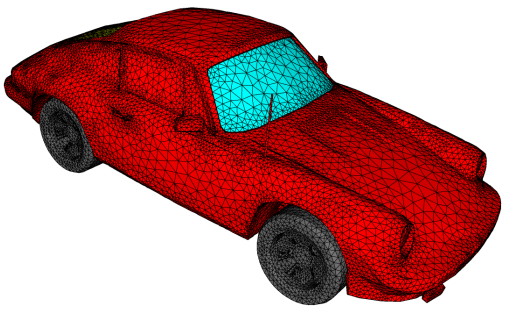
We are now able to run the tet mesher to produce a tetrahedral mesh from the surface, using for example MG-Tetra:



Note that MG-Cleaner cannot guarantee that it will be able to produce a fixed mesh correcting all faults, especially in the case of complex assemblies.

4.2.3 Producing a surface mesh with better aspect ratio of triangles

In our example, even if corrected and valid, the mesh is not suitable for classical finite elements computations because of the high aspect ratios of triangles. This can be improved with other tools, such as MG-SurfOpt. When running MG-SurfOpt on the surface mesh above to produce better triangles, this is what one obtains:



4.2.4 Improving mesh quality with MG-Cleaner

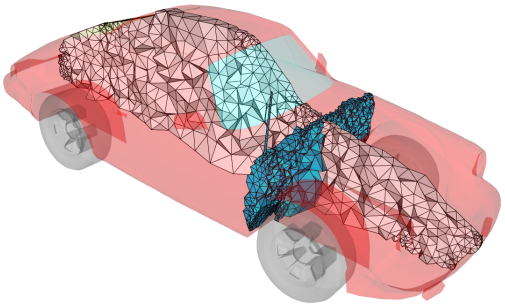
Even though this case can be meshed in volume, the quality of the worst element is quite poor. Such elements may occur when the input geometry is very "noisy", generating artificial geometrical constraints for MG-SurfOpt. These cannot be corrected by MG-SurfOpt because it always conforms to the input geometry.

These elements cannot be removed without altering the geometry, so we will use the mesh fixing component to achieve this goal.
For this, we can proceed with MG-Cleaner, for example as follows:

```
mg-cleaner.exe --in PorscheNew.mesh --out PorscheNewfix.mesh --fix --
    resolution_size  0.03
```

This will collapse all vertices closer than the given threshold size. A "good" value may be hinted from the mesh fixing analysis. In this case, the minimal size computed by the mesh fixing analysis (as shown in the standard output) is 0.003. Enlarging this size by a factor of 10 may be a good guess (hence the 0.03 value above).
In this example, the worst quality is improved by a factor of 10 in one run of MG-Cleaner.
Note that changing these values may very well destroy the geometry, so one must be careful with these values. In particular, small details might be lost!

Once the clean-up is performed, we can produce the tetrahedral mesh using MG-Tetra:



Appendix A

MG-Cleaner diagnostics

MG-Cleaner has its own advanced error management. It intercepts them and prints them on screen.

You may get some help about valid options and values by typing:

```
mg-cleaner --help
```

Only some errors and possible cause of failure are identified and listed here, the rest of the messages are informative messages and should be self explanatory.

In the list of some errors and possible cause of failure appearing below, we indicate after "ϟ" the diagnostic and after "✓" a possible way of correcting the problem if possible.

Detailed explanation and possible cure are provided only when necessary.

ϟ ERROR: required argument not found for option '--(string)'

ϟ ERROR: Missing input file.

ϟ ERROR: Unable to create output file name.

✓ Check if the destination directory is writable and if the destination file is not read only.

ϟ ERROR: Missing output file.

✓ As the output file name is computed by default, this means there has been a memory problem.

ϟ ERROR: file not found or empty '(string)'

ϟ ERROR: No input file...

ϟ ERROR: Incorrect value (string) for option (string)

ϟ ERROR: Unknown option (string)

Appendix B

File format description

Contents

In this chapter, we describe different types of data files used with MeshGems® Suite.

Refer to the main sections of the User Manual of the component you wish to use to know which types are actually handled and if there are any limitations.

The surface mesh formats are:

- ".mesh" (see B.1); and
- **[deprecated]** "MSH2" which is formed by ".points" and ".faces" files (see B.2).

B.1 the ".mesh" format

The ".mesh" file format is the main format used with MeshGems® Suite.

Unlike the other file description formats, the ".mesh" format is a flexible structure that is composed of facultative fields. We will describe the structure and keywords below.

A ".mesh" file can be written either in ASCII or binary format.

Note: Refer to the main sections of the User Manual of the component you wish to use to know if there are any limitations and what are the keywords handled by the software code.

B.1.1 Notation

In the following description, the terms or expressions written in `policy` represent file items. The blanks, "newline" (or Carriage Return) and Tabs are considered as item separators. The comment lines start with the character # and end at the end of the line, unless if they are in a string. The comments are placed between the fields and not inside fields.

The syntactic entities are:

keywords or field names

integer values (int),

floating values (float),

booleans (bool): 0 (resp. 1) for false (resp. true),

strings (char*) (up to 1024 characters) being placed between quotes (").
The blanks and carriage returns are significant when they are placed between quotes.
To have a quote defined in a string, one has to type it twice as in FORTRAN or in C languages: 'a"b' defines the string a"b.

The mesh file starts with a descriptor:

```
MeshVersionFormatted 1
```

Then, the various fields follow (required or facultative).

Note: all field keyword (except the `MeshVersionFormatted`) are alone on a line to simplify the file decoding.

B.1.2 Dimension (required)

```
Dimension
d
```

`int` d`;` `//` the space dimension, 2 or 3, must be specified

B.1.3 Vertices (required)

This field describes the vertex coordinates.

The first line contains the keyword and the next one the number of vertices contained:

```
Vertices
N^v
```

`int` N^v`;` `//` the number of vertices
Note: the numbering of these items is implicit *i.e.*, from 1 to N^v.

N^v lines then follow, one for each vertex. They are of the form:

```
x_i^1 ... x_i^d τ_i^v
```

`float` x_i^k`;` `//` the coordinates of vertex i with $1 \leq k \leq d$, d being the space dimension

`int` τ_i^v`;` `//` one integer characterising the attribute of point i

B.1.4 Elements (Triangles or Quadrilaterals required)

The elements are then given with respect to their geometric nature (edge, triangle, etc.).

Note: MG-Cleaner only works on 2D element sets *i.e.* at least one field `Triangles` or `Quadrilaterals` must be defined.

B.1.4.1 Edges

The field `Edges` includes usually only the edges with a non-null attribute or edges that are used in the list of `Ridges` and/or `RequiredEdges`.

Note: some MeshGems® Suite may rely on such assumption.

```
Edges
N^{e,e}
```

`int` $N^{e,e}$; `//` the number of edges

Note: the numbering of these items is implicit *i.e.*, from 1 to $N^{e,e}$.

$N^{e,e}$ lines then follow, one for each edge, of the form:

```
v_{i,1}^{e,e}  v_{i,2}^{e,e}  τ_i^{e,e}
```

`int` $v_{i,k}^{e,e}$; `//` the k-th vertex defining element edge i with $1 \leq k \leq 2$

`int` $\tau_i^{e,e}$; `//` one integer characterising attribute of element edge i

B.1.4.2 Triangles

```
Triangles
N^{e,t}
```

`int` $N^{e,t}$; `//` the number of triangles

Note: the numbering of these items is implicit *i.e.*, from 1 to $N^{e,t}$.

$N^{e,t}$ lines then follow, one for each triangle, of the form:

```
v_{i,1}^{e,t}  v_{i,2}^{e,t}  v_{i,3}^{e,t}  τ_i^{e,t}
```

`int` $v_{i,k}^{e,t}$; `//` the k-th vertex defining element triangle i with $1 \leq k \leq 3$

`int` $\tau_i^{e,t}$; `//` one integer characterising the attribute of element triangle i

B.1.4.3 Quadrilaterals

```
Quadrilaterals
N^{e,q}
```

`int` $N^{e,q}$; `//` the number of quadrilaterals

Note: the numbering of these items is implicit *i.e.*, from 1 to $N^{e,q}$.

$N^{e,q}$ lines then follow, one for each quadrilateral, of the form:

```
v_{i,1}^{e,q}  v_{i,2}^{e,q}  v_{i,3}^{e,q}  v_{i,4}^{e,q}  τ_i^{e,q}
```

`int` $v_{i,k}^{e,q}$; `//` the k-th vertex defining element quadrilateral i with $1 \leq k \leq 4$

`int` $\tau_i^{e,q}$; `//` one integer characterising the attribute of element quadrilateral i

B.1.4.4 Extra Vertices

Extra vertices may be added to elements. Those extra vertices are added in `Vertices` section and used in the following two new keywords, `ExtraVerticesAtTriangles` and `ExtraVerticesAtEdges`.

B.1.4.4.1 Extra Vertices At Edges

```
ExtraVerticesAtEdges
N^{e,e,e}
```

`int` $N^{e,e,e}$; // the number of edges with extra vertices.
Note: as of now, all edges have an extra vertex; so $N^{e,e,e} = N^{e,e}$.

$N^{e,e,e}$ lines then follow, one for each edge with extra vertices, of the form:

$$e_i^{e,e,e} \quad k_i^{e,e,e} \quad v_{i,1}^{e,e,e} \quad \dots \quad v_{i,k_i^{e,e,e}}^{e,e,e}$$

`int` $e_i^{e,e,e}$; // element edge index (implicit numbering of `Edges`).

`int` $k_i^{e,e,e}$; // number of extra vertices added to edge $e_i^{e,e,e}$
Note: as of now, only one vertex may be added to an edge leading to a quadratic curvilinear edge, so $k_i^{e,e,e} = 1$.

`int` $v_{i,j}^{e,e,e}$; // the j-th vertex added to edge $e_i^{e,e,e}$ (implicit numbering of `Vertices`).

B.1.4.4.2 Extra Vertices At Triangles

```
ExtraVerticesAtTriangles
N^{e,t,e}
```

`int` $N^{e,t,e}$; // the number of triangles with extra vertices.
Note: as of now, all triangles have extra vertices; so $N^{e,t,e} = N^{e,t}$.

$N^{e,t,e}$ lines then follow, one for each triangle with extra vertices, of the form:

$$e_i^{e,t,e} \quad k_i^{e,t,e} \quad v_{i,1}^{e,t,e} \quad \dots \quad v_{i,k_i^{e,t,e}}^{e,t,e}$$

`int` $e_i^{e,t,e}$; // element triangle index (implicit numbering of `Triangles`).

`int` $k_i^{e,t,e}$; // number of extra vertices added to triangle $e_i^{e,t,e}$
Note: as of now, only three vertices may be added to the triangle edges leading to a quadratic curvilinear triangle, so $k_i^{e,t,e} = 3$.

`int` $v_{i,j}^{e,t,e}$; // the j-th vertex added to triangle $e_i^{e,t,e}$ (implicit numbering of `Vertices`).

B.1.4.4.3 Extra Vertices At Tetrahedra

```
ExtraVerticesAtTetrahedra
N^{e,t,e}
```

`int` $N^{e,t,e}$; // the number of tetrahedra with extra vertices.
Note: as of now, all tetrahedra have extra vertices; so $N^{e,t,e} = N^{e,t}$.

$N^{e,t,e}$ lines then follow, one for each tetrahedron with extra vertices, of the form:

$$e_i^{e,t,e} \quad k_i^{e,t,e} \quad v_{i,1}^{e,t,e} \quad \dots \quad v_{i,k_i^{e,t,e}}^{e,t,e}$$

`int` $e_i^{e,t,e}$; // element tetrahedron index (implicit numbering of `Triangles`).

`int` $k_i^{e,t,e}$; // number of extra vertices added to tetrahedron $e_i^{e,t,e}$
Note: as of now, only six vertices may be added to the tetrahedron edges leading to a quadratic curvilinear tetrahedron, so $k_i^{e,t,e} = 6$.

`int` $v_{i,j}^{e,t,e}$; // the j-th vertex added to tetrahedron $e_i^{e,t,e}$ (implicit numbering of `Vertices`).

B.1.5 Geometric Information

Known geometric information about the surface may be given with the following fields.

In particular, a corner point `Corner` is a point where there is C^0 continuity between the edges sharing it (this type of item is necessary a mesh vertex).

By analogy, a `Ridge` is an edge where there is a C^0 continuity between the adjacent faces.

B.1.5.1 Corners

```
Corners
N^{g,c}
```

`int` $N^{g,c}$`;` `//` the number of corners

$N^{g,c}$ integers then follow in the form:

```
v_i^{g,c}
```

`int` $v_i^{g,c}$`;` `//` vertex index of the i-th corner (implicit numbering of `Vertices`).

B.1.5.2 Ridges

```
Ridges
N^{g,r}
```

`int` $N^{g,r}$`;` `//` the number of ridges

$N^{g,r}$ integers then follow in the form:

```
e_i^{g,r}
```

`int` $e_i^{g,r}$`;` `//` element edge index of the i-th ridge edge (implicit numbering of `Edges`).

B.1.5.3 Normals

The following fields concern the specification of normals. When specified, the normals are given as a list of vectors.

```
Normals
N^{g,n}
```

`int` $N^{g,n}$`;` `//` the number of normals

Note: the numbering of these items is implicit *i.e.*, from 1 to $N^{g,n}$.

$N^{g,n}$ lines then follow, one for each normal, of the form:

```
n_{i,1}^{g,n} ... n_{i,d}^{g,n}
```

`float` $n_{i,k}^{g,n}$`;` `//` the k-th coordinate of the normal vector i with $1 \leq k \leq d$, d being the space dimension

B.1.5.3.1 Normal At Vertices

When the normal is C^1 at a vertex, it may be specified using the following field:

```
NormalAtVertices
N^{g,n,v}
```

`int` $N^{g,n,v}$`;` `//` the number of normal at vertices

$N^{g,n,v}$ lines then follow, one for each vertex normal, of the form:

```
v_i^{g,n,t} n_i^{g,n,t}
```

`int` $v_i^{g,n,t}$`;` `//` vertex index of the i-th normal at vertex (implicit numbering of `Vertices`).

`int` $n_i^{g,n,t}$`;` `//` normal index of the i-th normal at vertex (implicit numbering of `Normals`).

B.1.5.3.2 Normal At Triangle Vertices

When the normal is C^0 at a triangle vertex, it may be specified using the following field:

`NormalAtTriangleVertices`
$N^{g,n,t}$

`int` $N^{g,n,t}$; // the number of normals at triangle vertex

$N^{g,n,t}$ lines then follow, one for each triangle vertex normal, of the form:

$e_i^{g,n,t}$ $k_i^{g,n,t}$ $n_i^{g,n,t}$

`int` $e_i^{g,n,t}$; // element triangle index of the i-th normal at triangle vertex (implicit numbering of `Triangles`).

`int` $k_i^{g,n,t}$; // local triangle index of the i-th normal at triangle vertex between 1 and 3 (local numbering of triangle vertices).

`int` $n_i^{g,n,t}$; // normal index of the i-th normal at triangle vertex (implicit numbering of `Normals`).

B.1.5.3.3 Normals At Quadrilateral Vertices

When the normal is C^0 at a quadrilateral vertex, it may be specified using the following field:

`NormalAtQuadrilateralVertices`
$N^{g,n,q}$

`int` $N^{g,n,q}$; // the number of normals at quadrilateral vertex

$N^{g,n,q}$ lines then follow, one for each quadrilateral vertex normal, of the form:

$e_i^{g,n,q}$ $k_i^{g,n,q}$ $n_i^{g,n,q}$

`int` $e_i^{g,n,q}$; // element quadrilateral index of the i-th normal at quadrilateral vertex (implicit numbering of `Quadrilaterals`).

`int` $k_i^{g,n,q}$; // local quadrilateral index of the i-th normal at quadrilateral vertex between 1 and 4 (local numbering of quadrilateral vertices).

`int` $n_i^{g,n,q}$; // normal index of the i-th normal at quadrilateral vertex (implicit numbering of `Normals`).

B.1.5.4 Tangents

The following fields concern the specification of tangents. When specified, the tangents are given as a list of vectors.

`Tangents`
$N^{g,t}$

`int` $N^{g,t}$; // the number of tangents
Note: the numbering of these items is implicit *i.e.*, from 1 to $N^{g,t}$.

$N^{g,t}$ lines then follow, one for each tangent, of the form:

$t_{i,1}^{g,t}$ ... $t_{i,d}^{g,t}$

`float` $t_{i,k}^{g,t}$; // the k-th coordinate of the tangent vector i with $1 \leq k \leq d$, d being the space dimension

B.1.5.4.1 Tangents At Edges

When several boundary lines emanate from a given point, it may be specified using the following field:

```
TangentsAtEdges
N^{g,t,e}
```

`int` $N^{g,t,e}$`;` `//` the number of tangents at edge vertex

$N^{g,t,e}$ lines then follow, one for each edge vertex tangent, of the form:

```
e_i^{g,t,e}  k_i^{g,t,e}  n_i^{g,t,e}
```

`int` $e_i^{g,t,e}$`;` `//` element edge index of the i-th tangent at edge vertex (implicit numbering of `Edges`).

`int` $k_i^{g,t,e}$`;` `//` local edge index of the i-th tangent at edge vertex between 1 and 2 (local numbering of edge vertices).

`int` $t_i^{g,t,e}$`;` `//` tangent index of the i-th tangent at edge vertex (implicit numbering of `Tangents`).

B.1.6 Required Entities

The user may want to keep some entities from the input mesh into the output mesh. The fields of type `Required`something make this possible to specify any type of entity that must be preserved by the meshing algorithm.

B.1.6.1 Required Vertices

```
RequiredVertices
N^{r,v}
```

`int` $N^{r,v}$`;` `//` the number of required vertices

$N^{r,v}$ integers then follow in the form:

```
v_i^{r,v}
```

`int` $v_i^{r,v}$`;` `//` vertex index of the i-th required vertex (implicit numbering of `Vertices`).

B.1.6.2 Required Edges

```
RequiredEdges
N^{r,e,e}
```

`int` $N^{r,e,e}$`;` `//` the number of required edges

$N^{r,e,e}$ integers then follow in the form:

```
e_i^{r,e,e}
```

`int` $e_i^{r,e,e}$`;` `//` element edge index of the i-th required edge (implicit numbering of `Edges`).

B.1.6.3 Required Triangles

```
RequiredTriangles
N^{r,e,t}
```

`int` $N^{r,e,t}$`;` `//` the number of required triangles

$N^{r,e,t}$ integers then follow in the form:

```
e_i^{r,e,t}
```

`int` $e_i^{r,e,t}$`;` `//` element triangle index of the i-th required triangle (implicit numbering of `Triangles`).

B.1.6.4 Required Quadrilaterals

```
RequiredQuadrilaterals
N^{r,e,q}
```

`int` $N^{r,e,q}$`; //` the number of required quadrilaterals

$N^{r,e,q}$ integers then follow in the form:

```
e_i^{r,e,q}
```

`int` $e_i^{r,e,q}$`; //` element quadrilateral index of the i-th required quadrilateral (implicit numbering of `Quadrilaterals`).

B.1.7 Extra Information in output mesh

Some extra information is written by MG-Cleaner in the output mesh.

B.1.7.1 Angle of Corner

```
AngleOfCornerBound
α
```

`float` α`; //` angle value, expressed in degrees $0 \leq \alpha \leq 180$, corresponding to the ridge angle defined by "`-Dridge=`α". It allows an automatic determination of the type of continuity between two edges or two faces that was not clearly defined or not explicitly specified in the input mesh.

B.1.7.2 Bounding Box

The size of the domain, *i.e.*, the extrema of its vertex coordinates will be stored in the following field:

```
BoundingBox
x^1_min  x^1_max
⋮
x^d_min  x^d_max
```

`float` $x^k_{\min}$,$x^1_{\max}$`; //` lower and upper bound for the k-th coordinate of vertices

B.1.7.3 End of File

The data structure ends with the keyword:

```
End
```

B.1.7.4 Contents of binary file

The file structure is slightly different than that of the Ascii file. It will be described in a next release of this document.

B.1.8 Specific keywords for MG-Cleaner

B.1.8.1 Fault_SmallTri

This highlights very small sized triangles.

```
Fault_SmallTri
N^{f,s,t}
```

`int` $N^{f,s,t}$`;` `//` the number of these triangles

$N^{f,s,t}$ integers then follow in the form:

```
e_i^{f,s,t}
```

`int` $e_i^{f,s,t}$`;` `//` element triangle index of the i-th triangle.

B.1.8.2 Fault_BadShape

This highlights narrow triangles.

```
Fault_BadShape
N^{f,b,s}
```

`int` $N^{f,b,s}$`;` `//` the number of narrow triangles.

$N^{f,b,s}$ integers then follow in the form:

```
e_i^{f,b,s}
```

`int` $e_i^{f,b,s}$`;` `//` element triangle index of the i-th triangle.

B.1.8.3 Fault_Overlap

This highlights triangles which overlap.

```
Fault_Overlap
N^{f,o,l}
```

`int` $N^{f,o,l}$`;` `//` the number of these triangles.

$N^{f,o,l}$ integers then follow in the form:

```
e_i^{f,o,l}
```

`int` $e_i^{f,o,l}$`;` `//` element triangle index of the i-th triangle.

B.1.8.4 Fault_FreeEdge

This highlights free edges.

```
Fault_FreeEdge
N^{f,f,e}
```

`int` $N^{f,f,e}$`;` `//` the number of free edges.

$N^{f,f,e}$ integers then follow in the form:

```
e_i^{f,f,e}
```

`int` $e_i^{f,f,e}$`;` `//` edge index of the i-th edge.

B.1.8.5 Fault_Inter

This highlights intersections.

```
Fault_Inter
N^{f,i,t}
```

`int` $N^{f,i,t}$`;` `//` the number of intersections.

$N^{f,i,t}$ integers then follow in the form:

```
e_i^{f,i,t}
```

`int` $e_i^{r,i,t}$`;` `//` element triangle index of the i-th triangle.

B.1.8.6 Fault_NearTri

This highlights triangles which are very close without being connected.

```
Fault_NearTri
N^{f,n,t}
```

`int` $N^{f,n,t}$`;` `//` the number of these triangles.

$N^{f,n,t}$ integers then follow in the form:

```
e_i^{f,n,t}
```

`int` $e_i^{r,n,t}$`;` `//` element triangle index of the i-th triangle.

B.1.9 Sample

We show below mostly the parts of the file where there is a field change and we put an ellipsis marks (...) in place of the most part of a long field.

```
MeshVersionFormatted 1

Dimension
3

# Set of mesh vertices
Vertices
682
94.444450378418 94.4444427490234 -3.99593202971005e-09 0
89.3760375976562 84.1671295166016 -2.91678103891968e-09 0
84.1671295166016 89.3760375976562 -3.46373307813508e-09 0
```

...

```
53.5949287414551 37.586181640625 100 0
91.6666564941406 25 100 0

# Set of mesh triangles (v1,v2,v3,tag)
Triangles
1364
1 2 3 1
4 2 1 1
5 6 7 1
```

...

```
656 621 622 1
658 656 622 1

# Set of corners
Corners
16
22
32
```

...

```
676

# Set of mesh edges (v1,v2,tag)
Edges
160
9 7 0
22 23 0
24 22 0
27 28 0
```

...

```
646 671 0

# Set of ridges (iv,tag)
Ridges
154
1
2
3
4
```

...

```
160

# List of normal vectors
Normals
842
2.07356437575967e-10 1.9641839543727e-10 -1
4.09822488445233e-11 -1.45596243394941e-10 -1
-6.54994461446456e-11 9.58556411329026e-12 -1
4.38153485715631e-10 -1.17958004475227e-10 -1
```

...

```
0 0 1

# Normals at vertices
NormalAtVertices
528
1 1
2 2
3 3
5 5
```

...

```
682 842

# Normals at triangle vertices
NormalAtTriangleVertices
937
2 1 4
3 3 7
4 2 9
4 3 10
```

...

```
1364 2 797

# Angle of corner bound (degrees)
AngleOfCornerBound
45.0005515606517

# Bounding box
BoundingBox
0 100
0 100
-3.99593202971005e-09 100

End
```

B.2 the msh2 format

Important notice: Please note that this format is deprecated and `.mesh` format should be used from now on (see section B.1).

This file format is composed of two ASCII files with the suffixes "`.points`" and "`.faces`".

B.2.1 File "`.points`"

This file contains in order:

Line 1 :

N^v

`int` N^v; // the number of points

Line $i+1$: for point number i, $1 \leq i \leq N^v$:

$x_i \;\; y_i \;\; z_i \;\; \tau_i^v$

`float` x_i,y_i,z_i; // the three (real, single precision) point i coordinates

`int` τ_i^v; // one integer characterising the point i attribute

B.2.2 File "`.faces`"

This file contains in order:

Line 1 :

N^e

`int` N^e; // the number of elements

Line $j+1$: for facet number j, $1 \leq j \leq N^e$:

$t_j \;\; v_{j,1} \;\; \ldots \;\; v_{j,t_j} \;\; \tau_j^e \;\; \tau_{j,1}^f \;\; \ldots \;\; \tau_{j,t_j}^f$

`int` t_j; // facet type

- a triangle is a type 3 face with three vertices and three frontiers (edges)
- a quadrilateral is a type 4 face with four vertices and four frontiers (edges)
- an edge is a type 2 face with two vertices and two frontiers (points)

Other face types can be described but are not supported by the software.

`int` $v_{j,k}$; // the k-th vertex numbers defining facet j with $1 \leq k \leq t_j$

`int` τ_j^e; // one integer characterising the facet j attribute

`int` $\tau_{j,k}^f$; // integers characterising the k-th frontier attributes for facet j with $1 \leq k \leq t_j$

B.2.3 Sample

bielle .points:

```
758
6592.270 558.9722 4673.861 0
7003.462 0.3326008 4673.861 0
6259.994 0.3326008 4673.861 0
6154.081 978.4711 4673.861 0
5447.993 0.3326008 4555.487 0
```

...

```
26136.35 8.639509 3558.658 0
26686.68 10.71624 649.1638 0
26562.08 -5.897580 291.9667 0
26582.84 10.71624 4366.505 0
26686.68 10.71624 4023.845 0
```

bielle . faces:

```
1512 0
3 17 43 39 0 0 0 0
3 113 144 504 0 0 0 0
3 437 60 463 0 0 0 0
3 711 721 722 3 0 0 0
3 173 174 207 6 0 0 0
```

...

```
3 317 758 757 4 0 0 0
3 404 430 439 0 0 0 0
3 628 578 631 6 0 0 0
3 711 722 714 3 0 0 0
```

Appendix C

List of FAQ's

Contents

In this section, we will attempt to answer some frequently asked questions about the MG-Cleaner product.

C.1 My mesh contains quads, is MG-Cleaner capable of correcting these?

In theory, no. However, MG-Cleaner is capable of processing quad meshes. Quads are indeed split into two triangles automatically once read by MG-Cleaner, so that it can process the mesh and clean it up. A pointer is kept to be able to revert these split quads into their original form, once the cleaning process is over, unless some of their triangles were removed or swapped. Attributes are also reverted to the original in the process

C.2 Is it a good idea to iterate MG-Cleaner on itself?

It may prove useful on some cases, but in general this does not help so much because failures are often caused by very complex geometries or assemblies. One can think of using the `--fix2pass` option alternatively.

C.3 How do I improve the quality of triangles?

One can use the `--resolution_length` ⟨size⟩ parameter to force the removal of bad quality triangles, or use MG-SurfOpt to improve the surface mesh quality. While MG-SurfOpt will always conform to the geometry and thus may be limited in its scope of mesh improvement, MG-Cleaner is less limited than MG-SurfOpt, even though there are some geometry accuracy considerations taken into account.

C.4 Is MG-Cleaner capable of guaranteeing a corrected mesh suitable for a Delaunay or front advancing tet mesher?

Unfortunately no, as the current approach tries to conform as much as possible to the original geometry. There are cases where the geometry is composed of a heap of intersecting triangles which cannot even define a geometry.

C.5 How is associativity taken care of ?

Since MG-Cleaner only works on triangulated surfaces, there is no direct associativity with the CAD system. However, since MG-Cleaner preserves attributes of vertices, elements and since release 1.1, edges, it is possible to keep track of modified items throughout the meshing process from CAD. One can then identify where the corrected mesh entities are originating from

C.6 Can my geometry be altered ?

MG-Cleaner may indeed modify the geometry, as it aims at correcting several issues preventing the tet mesher from succeeding. However , it contains a failsafe built-in automatic parameter set-up which prevents the geometry to be altered too significantly, as long as the automatically computed default parameters are used.

C.7 How can I preserve as much as possible my input topology ?

MG-Cleaner provides an option `--topology` respect[1] to prevent topology changes. This however reduces significantly the amount of issues which can be corrected successfully by MG-Cleaner.

[1] Was previously `-T`.

Appendix D

Release Notes

Released versions are described below.

D.1 Version 1.0

This version is the first version of MG-Cleaner, which is part of of the MeshGems® Suite and which replaces MeshCleaner. The changes and new features introduced by this new module are detailed below.

D.1.1 New features

- Transmission of input edge attributes is now supported

D.1.2 Improvements

- Significantly increase of the number of cases which can be corrected successfully, compared to the previous release

D.1.3 Corrections

- Several bugs and issues are corrected, in particular option "`--remesh_planes`" adds vertices as it should.

Appendix E

References

MG-Tetra: User's manual

List of Figures