Data Structure : "mesh" file

INRIA, Gamma project

21 janvier 2000

1 Abstract

General description of a mesh in two and three dimensions. The mesh can be composed of triangles, quadrilaterals, tetrahedra, pentahedra or hexahedra. The structure can also handle surface meshes.

2 Usage

- GHS3D, GAMHIC3D : tetrahedral mesh generation,
- Medit, Visu : visualization of meshes,
- Yams : Surface simplification and remeshing.
- BLSURF : Parametric surface meshing.

3 Description

3.1 Identification

Let *xxx* be a basename, the file name is :

- *xxx*.mesh for an Ascii file,
- *xxx*.meshb for a binary file.

3.2 Characteristics

- Ascii
- Binary.

Remark : A binary file is less demanding in terms of memory occupation, size and CPU time but is also less easy to transfer from one type of computer to another.

3.3 Contents

Contrary to the other file description formats, the 'mesh' format is a flexible structure that is composed of facultative fields. We will not describe the structure *in extenso* here, but only specify the fields used by the software codes described above.

Notation. In the following description, the terms or expressions in `policy` represent file items. The blancks, "newline" or <CR> 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.

The syntactic entities are field names, integer values `(I)` , (double) floating values `(R)` , booleans `(B)` : 0 (resp. 1) for false (resp. true), char strings `(C*)` (up to 1024 characters) being placed between "". The blanks and "new lines" are significant when used between quotes and to use a quote " in a string, one has to type it twice " " (as in FORTRAN or in "C"). Finally, numbers, for instance a vertex number, is denoted by `@Vertex`.

The notation (..., i=1,n) stands for an implicit DO loop as in FORTRAN.

Content (ASCII file). Usually, the mesh file contains :

- `MeshVersionFormatted 1`

The string `MeshVersionUnformatted` is used to identify binary files.

Then, follow the various fields (required or facultative). At first, the dimension (2 or 3) must be specified :

- `Dimension`
  `(I)` dim

The next field corresponds to the vertex coordinates, that induces an implicit numbering of these items (*i.e.*, from 1 to NbOfVertices) :

- `Vertices`
  `(I)` NbOfVertices
  ((`(R)` x_i^j , j=1,dim) , `(I)` $Ref\phi_i^v$, i=1 , NbOfVertices)

The elements are then given with respect to their geometric nature (edge, triangle, tetrahedron, etc.). Notice that the field `Edges` includes only the edges with a non-null reference or edges that are used in the list of `Ridges` and/or `RequiredEgdes`.

- `Edges`
  `(I)` NbOfEdges
  (`@Vertex`$_i^1$, `@Vertex`$_i^2$, `(I)` $Ref\phi_i^e$, i=1 , NbOfEdges)
- `Triangles`
  `(I)` NbOfTriangles
  ((`@Vertex`$_i^j$, j=1,3) , `(I)` $Ref\phi_i^t$, i=1 , NbOfTriangles)
- `Quadrilaterals`
  `(I)` NbOfQuadrilaterals
  ((`@Vertex`$_i^j$, j=1,4) , `(I)` $Ref\phi_i^t$, i=1 , NbOfQuadrilaterals)
- `Tetrahedra`
  `(I)` NbOfTetrahedra
  ((`@Vertex`$_i^j$, j=1,4) , `(I)` $Ref\phi_i^t$, i=1 , NbOfTetrahedra)
- `Pentahedra`
  `(I)` NbOfPentahedra
  ((`@Vertex`$_i^j$, j=1,6) , `(I)` $Ref\phi_i^t$, i=1 , NbOfPentahedra)

- `Hexahedra`
  `(I)` NbOfHexahedra
  ((`@Vertex`$_i^j$,j=1,8) , `(I)` $Ref\phi_i^t$, i=1 , NbOfHexahedra)

Then, arrives the description of constrained entities or singularities (for example in a surface). In particular, a corner point `Corner` is a point where there is a 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. The fields of type `Requiredxx` make it possible to specify any type of entity that must be preserved by the meshing algorithm.

- `Corners`
  `(I)` NbOfCorners
  (`@Vertex`$_i$, i=1 , NbOfCorners)
- `Ridges`
  `(I)` NbOfRidges
  (`@Edge`$_i$, i=1 , NbOfRidges)
- `RequiredVertices`
  `(I)` NbOfRequiredVertices
  (`@Vertex`$_i$, i=1 , NbOfRequiredVertices)
- `RequiredEdges`
  `(I)` NbOfRequiredEdges
  (`@Edge`$_i$, i=1 , NbOfRequiredEdges)
- `RequiredTriangles`
  `(I)` NbOfRequiredTriangles
  (`@Tria`$_i$, i=1 , NbOfRequiredTriangles)
- `RequiredQuadrilaterals`
  `(I)` NbOfRequiredQuadrilaterals
  (`@Quad`$_i$, i=1 , NbOfRequiredQuadrilaterals)

The following fields concern the specification of normals and tangents. The normals (resp. tangents) are given as a list of vectors. The normal at a vertex, `NormalAtVertices`, is specified using the vertex number and the index of the corresponding normal vector. The normal at a vertex of a triangle, `NormalAtTriangleVertices`, corresponds to the combination of the triangle number, the index of the vertex in the triangle and the index of the normal vector at this vertex. Similarly for the field `NormalAtQuadrilateralVertices`. The tangent vectors `Tangents` are given in the same way. The tangent at an edge, `EdgesTangent`, is specified using the edge number, the index of the vertex of the edge (*i.e.*, the endpoint) and the index of the tangent vector at the vertex. Giving the tangent vector of an edge by means of the tangent vector at a point enables us to deal with the case where several edges (boundary lines) emanate from a given point.

- `Normals`
  `(I)` NbOfNormals
  ((`(R)` x_i^j ,j=1,dim) ,i=1 , NbOfNormals)
- `NormalAtVertices`
  `(I)` NbOfNormalAtVertices
  (`@Vertex`$_i$, `@Normals` ,), i=1 , NbOfNormalAtVertices
- `NormalAtTriangleVertices`
  `(I)` NbOfNormalAtTriangleVertices
  (`@Tria`$_i$, `(I)` VertexInTrian. , `@Normals` ,), i=1 , NbOfNormalAtTriangleVertices

- `NormalAtQuadrilateralVertices`
  `(I)` NbOfNormalAtQuadrilateralVertices
  (`@Quad`$_i$, `(I)` VertexInQuad. , `@Normals` ,), i=1 , NbOfNormalAtQuadrilateralVertices

- `Tangents`
  `(I)` NbOfTangents
  ((`(R)` x_i^j , j=1,dim) , i=1 , NbOfTangents)

- `TangentAtEdges`
  `(I)` NbOfTangentAtEdges
  (`@Edge`$_i$, `(I)` VertexInEdge , `@Tangents` ,), i=1 , NbOfTangentAtEdges

The field `AngleOfCornerBound` is a value that determines the type of continuity between two edges or two faces that was not clearly defined or not explicitely specified. It is an angle value, expressed in degrees : $0. \leq \theta \leq 180.$.

- `AngleOfCornerBound`
  `(R)` θ

The next three fields allow to define cracks, by specifying that an edge (a face, respectively) is geometrically identical to another edge (face).

- `CrackedEdges`
  `(I)` NbOfCrackedEdges
  (`@Edge`$_i^1$, `@Edge`$_i^2$, i=1 , NbOfCrackedEdges)

- `CrackedTriangles`
  `(I)` NbOfCrackedTriangles
  (`@Tria`$_i^1$, `@Tria`$_i^2$, i=1 , NbOfCrackedTriangles)

- `CrackedQuadrilaterals`
  `(I)` NbOfCrackedQuadrilaterals
  (`@Quad`$_i^1$, `@Quad`$_i^2$, i=1 , NbOfCrackedQuadrilaterals)

The size of the domain, *i.e.,* the extrema of its point coordinates will be stored in the field `BoundingBox`:

- `BoundingBox`
  (`(R)` Min_i `(R)` Max_i , i=1 , dim)

The data structure ends with the keyword :

- `End`

Content (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.

4 Example

Example of an Ascii triangular surface mesh file :

```
# Mesh generated by YAMS (INRIA) V1.1
MeshVersionFormatted 1

Dimension
3
```

```
# Set of mesh vertices
Vertices
7902
-3.299989938735962 1.829990029335022 -4.139999866485596 0
-3.209990024566650 1.620000004768372 -3.939990043640137 0
-3.170000076293945 0.629989981651306 -4.149990081787109 0
-3.099989891052246 0.629989981651306 -3.939990043640137 0
-3.079989910125732 1.509999990463257 -3.890000104904175 0
-3.049989938735962 -0.489989995956421 -4.139999866485596 0
.....

# Set of mesh edges (v1,v2,ref)
Edges
891
401 13 0
13 389 0
410 15 0
413 17 0
478 12 0
510 1 0
.....

# Set of mesh triangles (v1,v2,v3,ref)
Triangles
15346
13 389 401 0
410 15 406 0
17 412 413 0
445 441 446 0
.....

# Set of corners
Corners
82
1
2
9
13
17
.....

# Set of ridges
Ridges
891
1
2
3
4
5
6
.....
```

```
# List of normal vectors
Normals
8299
0.244684616817435 0.031587128807099 -0.969088072151664
0.254205560291861 0.033309848788757 -0.966576425892113
0.251236808287723 0.034918265235304 -0.967295601620495
0.298865337781927 0.043201363979187 -0.953316921082823
0.321752194211001 0.055406270683441 -0.945201391603588
0.319152976658203 0.055225513509790 -0.946092765085850
.....

# Normals at vertices
NormalAtVertices
7006
406 1391
412 6
445 16
441 15
446 1423
.....


# Normals at triangle vertices
NormalAtTriangleVertices
4119
1 1 1
1 2 2
1 3 3
2 1 1395
2 2 4
3 1 5
3 3 7
5 1 17
.....

# Angle of corner bound (degrees)
AngleOfCornerBound
45.001000714163766

# Bounding box
BoundingBox
-3.775069952011108 3.697506904602051
-7.363364696502686 7.399990081787109
-4.302670001983643 0.042770355939865

End
```