Introduction à Gmsh

Nicolas Kielbasiewicz

23 octobre 2013

Gmsh est un logiciel libre créé par Christophe Geuzaine [1] et Jean-François Remacle [2], permettant de générer des maillages 2D et 3D de type éléments finis avec des outils de pré/post-traitement. C'est un logiciel multi-plateforme (Windows, Linux et Mac OS X) téléchargeable sur :

http://geuz.org/gmsh

L'objectif ici est de présenter les commandes élémentaires permettant de générer des maillages 2D ou 3D au format msh à l'aide de cet outil. Je présenterai aussi des exemples permettant, entre autres, de générer des maillages de toutes les natures possibles, avec les outils nécessaires et les contraintes qu'ils imposent. Pour des informations détaillées sur les fonctionnalités de Gmsh, je vous renvoie à la documentation disponible sur le site susnommé.

Table des matières

1. Professeur à l'Université de Liège : http://www.montefiore.ulg.ac.be/~geuzaine
2. Professeur à l'Université Catholique de Louvain http://perso.uclouvain.be/jean-francois.remacle/

1 Le format msh

Un fichier de maillage au format msh est organisé en blocs de données. Chacun de ces blocs est décrit dans une sous-section, dans l'ordre d'apparition dans le fichier de maillage (au moins pour les blocs obligatoires).

1.1 Informations du fichier de maillage

Le premier bloc concerne les informations générales sur le fichier :

```
$MeshFormat
version-number file-type data-size
$EndMeshFormat
```

`version-number` est un réel valant actuellement 2.1 ou 2.2 (dernières versions du format msh).

`file-type` vaut 0 pour un fichier texte.

`data-size` précise le nombre de décimales significatives d'un flottant double précision (=sizeof(double)).

1.2 Définition des nœuds

Le bloc suivant concerne la définition des nœuds, à savoir leur nombre et la liste de leurs coordonnées :

```
$Nodes
number-of-nodes
node-number x-coord y-coord z-coord
...
$EndNodes
```

1.3 Définition des éléments

Le bloc suivant concerne la définition des éléments (triangles, quadrangles, tétraèdres, ...) :

```
$Elements
number-of-elements
elm-number elm-type number-of-tags <tags> node-number-
    list
...
$EndElements
```

`elm-type` désigne le type d'éléments finis (géométrie de la maille et ordre). Consulter le Tableau 1.

`number-of-tags` désigne le nombre de paramètres de l'élément.

`<tags>` désigne la liste de ces paramètres, de taille `number-of-tags`. Par défaut, le premier paramètre est la référence de l'entité physique à laquelle appartient l'élément, le second est la référence de l'entité géométrique élémentaire contenant cet élément. Quand on partitionne le maillage (Gmsh propose cette fonctionnalité), un troisième tag est ajouté et représente la référence de la partition du maillage contenant cet élément.

`node-number-list` désigne la liste ordonnée des références des nœuds de l'élément (cf. sous-section 1.6).

1.4 Définition des labels

La section suivante concerne la définition des labels physiques des différentes régions du maillage. Ce bloc n'est utilisé que pour spéficier des labels par des chaînes de caractères.

`elm-type`	géométrie	nb de nœuds	ordre	position
1	segment	2	1	2S
2	triangle	3	1	3S
3	quadrangle	4	1	4S
4	tétraèdre	4	1	4S
5	hexaèdre	8	1	8S
6	prisme	6	1	6S
7	pyramide	5	1	5S
8	segment	3	2	2S+1A
9	triangle	6	2	3S+3A
10	quadrangle	9	2	4S+4A+1F
11	tétraèdre	10	2	4S+6A
12	hexaèdre	27	2	8S+12A+6F+1V
13	prisme	18	2	6S+9A+3FQ
14	pyramide	14	2	5S+8A+1FQ
15	point	1	N/A	1S
16	quadrangle	8	2	4S+4A
17	hexaèdre	20	2	8S+12A
18	prisme	15	2	6S+9A
19	pyramide	13	2	5S+8A
20	triangle	9	3	3S+6A
21	triangle	10	3	3S+6A+1F
22	triangle	12	4	3S+9A
23	triangle	15	4	3S+9A+3F
24	triangle	15	5	3S+12A
25	triangle	21	5	3S+12A+6F
26	segment	4	3	2S+2A
27	segment	5	4	2S+3A
28	segment	6	5	2S+4A
29	tétraèdre	20	3	4S+12A+4F
30	tétraèdre	35	4	4S+18A+12F+1V
31	tétraèdre	56	5	4S+24A+24F+4V

TABLE 1 – Quelques valeurs possibles pour le paramètre `elm-type`. Le nombre de nœuds est entre parenthèses. Pour la position : A=arête, F=face, FQ=face quadrangulaire, S=sommet, V=volume.

```
$PhysicalNames
number-of-names
physical-dimension physical-number "physical name"
...
$EndPhysicalNames
```

1.5 Définition des données de post-traitement

Les blocs de post-traitement servent) tracer une quantité sur le maillage ainsi défini. Ces blocs peuvent être définis dans des fichiers séparés de celui contenant la définition du maillage. Chaque fichier représente alors un pas de temps. Ils contiendront donc les valeurs à tracer sur chacun des nœuds/éléments du maillage, répartis en 3 catégories.

1.5.1 Les données de nœuds

```
$NodeData
number-of-string-tags
<"string-tag">
...
number-of-real-tags
<real-tag>
...
number-of-integer-tags
<integer-tag>
...
node-number value ...
...
$EndNodeData
```

Les 3 premières séries de données concernent les étiquettes sur les nœuds, rangés par leur type : chaîne de caractères, entier ou flottant.

La quatrième et dernière série de données concerne les valeurs affectées à chacun des nœuds.

`value` désigne la valeur de la quantité à tracer sur chacun des nœuds.

1.5.2 Les données des éléments

```
$ElementData
number-of-string-tags
<"string-tag">
...
```

```
number-of-real-tags
<real-tag>
...
number-of-integer-tags
<integer-tag>
...
elm-number value ...
...
$EndElementData
```

Il s'agit de la même organisation que pour les nœuds. Ce type de données sert à tracer des fonctions constantes par maille.

`value` désigne ici la valeur de la quantité à tracer sur chacun des éléments.

1.5.3 Les données des nœuds d'éléments

```
$ElementNodeData
number-of-string-tags
<"string-tag">
...
number-of-real-tags
<real-tag>
...
number-of-integer-tags
<integer-tag>
...
elm-number number-of-nodes-per-element value ...
...
$EndElementNodeData
```

Là encore, on retrouve la même organisation de données. Il s'agit d'une verison plus générale que les données d'éléments, car ici, plutôt que d'affecter la même valeur à chacun des nœuds d'une maille, on va spécifier ici la valeur pour chacun de ces nœuds, maille après maille.

`value` désigne la valeur de la quantité à tracer sur chacun des nœuds de chacun des éléments.

1.6 Numérotation des éléments

Je vas présenter ici la numérotation des éléments utilisée par Gmsh sur les 3 familles d'éléments plans, à savoir les segments (Figure 1), les triangles (Figure 2) et les quadrangles (Figure 3). Les exemples porteront sur les éléments finis de Lagrange d'ordres respectifs 1, 2 et 4. On remarquera que l'ordre de numérotation des éléments concerne les sommets, puis les nœuds

d'arêtes, psui les nœuds de faces. En 3D viendront s'ajouter ensuite les nœuds de volume.

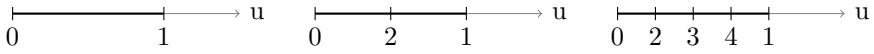

FIGURE 1 – Numérotation des nœuds sur un élément de type segment (Lagrange P1 à gauche, P2 au centre et P4 à droite).

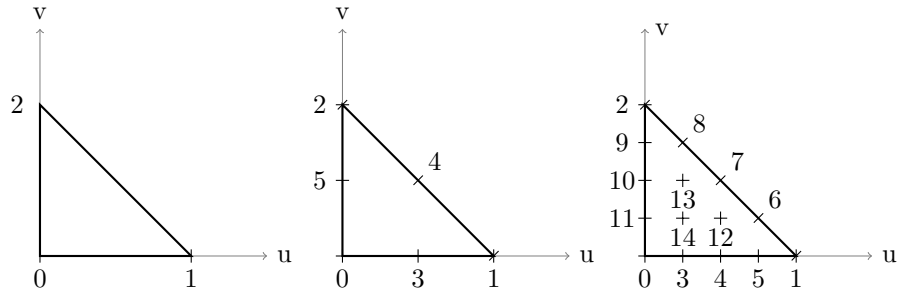

FIGURE 2 – Numérotation des nœuds sur un élément de type triangle (Lagrange P1 à gauche, P2 au centre et P4 à droite).

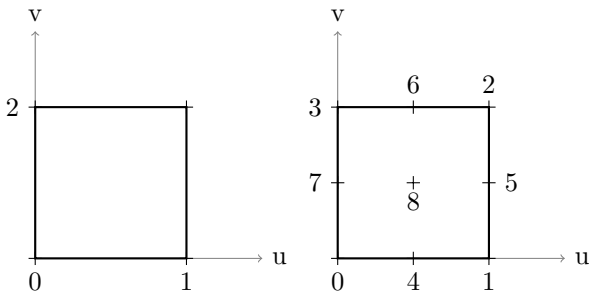

FIGURE 3 – Numérotation des nœuds sur un élément de type quadrangle (Lagrange P1 à gauche, P2 à droite).

Après cette présentation du format du maillage généré par GMSH, je vais à présent vous expliquer le fonctionnement de GMSH, en commençant par son interface graphique.

2 GMSH, outil graphique

GMSH est un logiciel possédant une interface graphique minimaliste, mais assez fonctionnelle. Elle se compose de deux frames : un menu, articulé en 4 modules, et une fenêtre principale (Figure 4). Les boutons du menu donnent accès à des sous-menus.

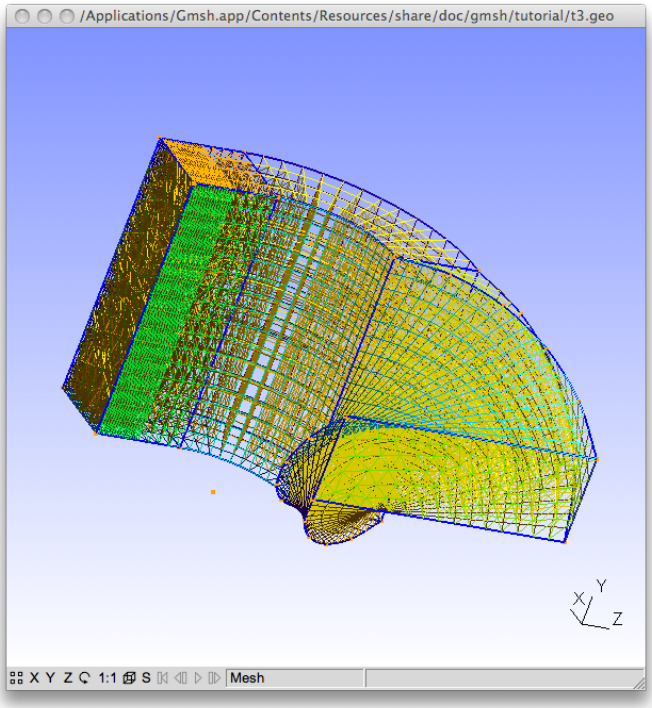

FIGURE 4 – La fenêtre principale

2.1 Le module `geometry`

Le module `geometry` (Figure 5) est disponible sur le menu par la liste de sélection ou le raccourci `g`.

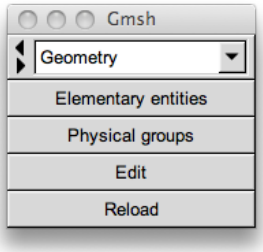

FIGURE 5 – Menu du module `geometry`.

Il est dédié à la construction géométrique du maillage, avec la distinction entre les éléments purement géométrique utiles à la construction, et les éléments physiques du maillage à générer (avec leurs références, ...). C'est aussi par ce menu que l'on peut accéder au fichier de script (extension .geo) servant à générer le maillage, ainsi qu'à le recharger après modification directe. Je parlerai plus en détails de la syntaxe des instructions présentes dans ces fichiers dans la section suivante.

Considérons que nous voulons définir un nouveau point. On va donc cliquer respectivement sur `Elementary entities`, `Add`, `New` et `Point`. Une fenêtre apparaît alors (Figure 6). Hormis les coordonnées spatiales du point, un champ très important à remplir est celui de la longueur caractéristique. Sa valeur représente la valeur du pas de maillage en ce point. Cela permet ainsi de définir des maillages graduellement plus raffinés d'un point à un autre.

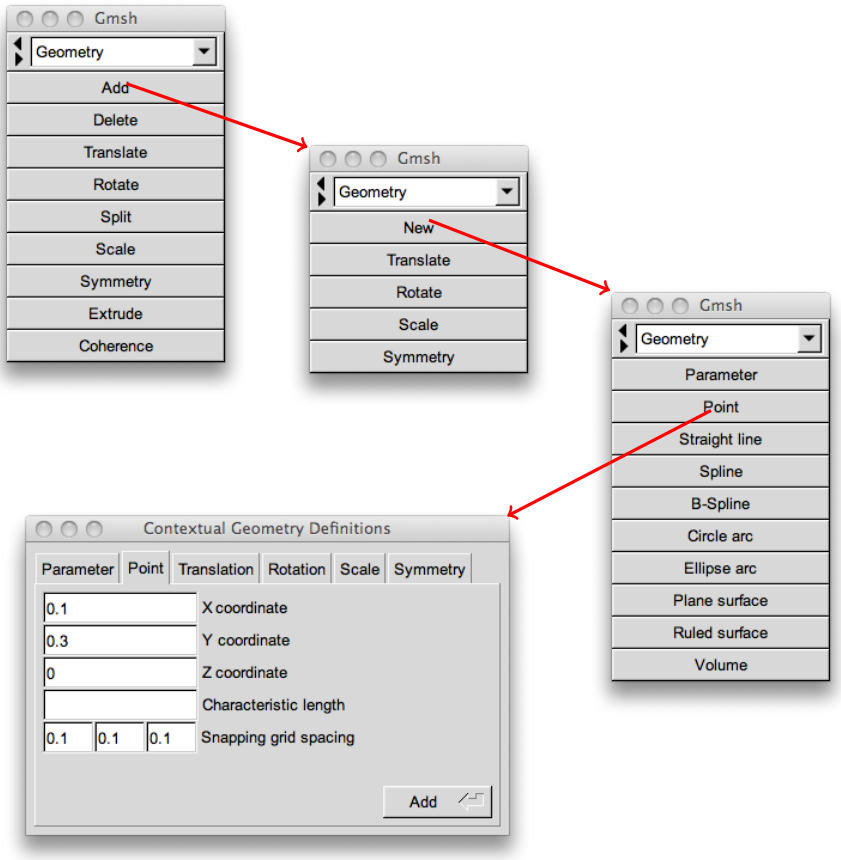

FIGURE 6 – Aspect du menu à chaque étape conduisant à la création d'un point par ouverture de la fenêtre contextuelle dédiée.

À partir de ces points, on peut alors tracer des segments en reliant deux points, des arcs de cercles à partie de 3 points, ...

2.2 Le module `mesh`

Une fois la surface géométrique définie, on va générer le maillage. Pour cela, on choisit le module `mesh`, soit grâce à la liste de sélection, soit avec le raccourci `m` (Figure 7). À partir de ce menu, on va pouvoir mailler les contours, surfaces et volumes en précisant l'ordre d'approximation des éléments finis, effectuer des opérations de partitionnement, de raffinement et d'homogénéisation des mailles. C'est ici également que l'on va sauvegarder le maillage au format msh.

Remarque : Pour des éléments finis d'ordre supérieur, lorsqu'il s'agit de bords courbes, les points supplémentaires sont par défaut, si nécessaires déplacés sur le bord au lieu de rester sur l'arête de la maille.

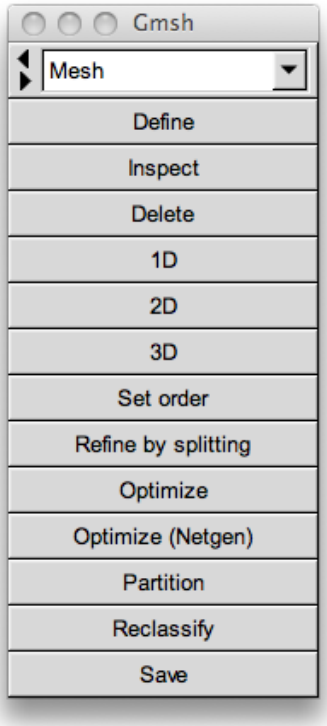

FIGURE 7 – Menu du module `mesh`.

2.3 Le module `solver`

Gmsh peut être interfacé avec un solveur C++. Pour cela, il suffit d'inclure GmshSocket.h dans le code. 5 solveurs peuvent être interfacés simultanément avec Gmsh.

2.4 Le module `post-processing`

Comme nous l'avons vu en présentant le format msh, il existe la possibilité de tracer des données à condition qu'elles soient données dans le bon format. Le module `post-processing` permet de tracer ces données et de les visualiser dans l'interface de Gmsh.

3 Gmsh, outil en ligne de commande

3.1 Exécution en ligne de commande

Comme nous l'avons vu, la construction du maillage en mode graphique s'accompagne de la génération d'un fichier de script d'extension .geo. Gmshpeut être utilisé en ligne de commande moyennant une série d'options fort pratique. Voici quelques exemples de combinaisons d'options :

gmsh -help cette commande affiche l'aide avec la liste complète des options

gmsh -g file.geo cette commande démarre le script *file.geo* avec le module `geometry` (comportement par défaut)

gmsh -m file.geo cette commande démarre le script *file.geo* avec le module `mesh`

gmsh -s file.geo cette commande démarre le script *file.geo* avec le module `solver`

gmsh -p file.geo cette commande démarre le script *file.geo* avec le module `post-processing`

gmsh -1 file.geo -o maillage.msh cette commande génère le fichier *maillage.msh* à partir d'un maillage 1D sur la géométrie contenue dans *file.geo* et sort (mode console)

gmsh -2 -order 3 file.geo -o maillage.msh cette commande génère le fichier *maillage.msh* à partir d'un maillage 2D d'ordre 3 sur la géométrie contenue dans *file.geo* et sort (mode console)

gmsh -3 -clscale 0.1 file.geo -o maillage.msh cette commande génère le fichier *maillage.msh* à partir d'un maillage 3D de longueur caractéristique (=pas) 0.1 sur la géométrie contenue dans *file.geo* et sort (mode console)

3.2 Éléments de syntaxe

La syntaxe des scripts geo est très facile d'accès et est basée sur le principe suivant : tout objet géométrique est identifié par une référence, et ce sont ces références qui sont utilisées pour définir des objets plus complexes.

3.3 Points de départ

Un point est défini par un quadruplet formé par les 3 coordonnées x, y et z, et par une longueur caractéristique. Comme nous l'avons vu précédemment, cette longueur caractéristique s'apparente à la valeur du pas localement autour de ce point. La syntaxe est la suivante :

```
lc = 0.001;
Point(1) = {0.0, 0.0, 0.0, lc};
```

Il est impossible de définir 2 points différents avec le même identifiant (erreur à l'exécution). Par contre, la valeur de cet identifiant est un entier naturel non nul quelconque.

3.4 Lignes et contours

Les lignes sont des objets orientés qui peuvent être de différentes natures : segments, arcs de cercle, splines, arcs elliptiques, ...

```
Line(2) = {1, 3};
Circle(3) = {5, 1, 4};
Ellipse(4) = {8, 7, 6, 9};
Line Loop(2) = {2, 5, -3, 6};
```

Le segment référencé 2 va du point référencé 1 au point référencé 3. L'arc de cercle référencé 3 va du point 5 au point 4, et est basé sur un cercle de centre le point référencé 1. L'arc elliptique référencé 4 commence au point référencé 8 et se termine au point référencé 9. Ses paramètres sont le centre de l'ellipse (le point référencé 7), et la direction du grand axe (le point référencé 6). La dernière commande permet de définir un contour fermé. Dans l'exemple, sa référence est 2 et il est construit à partir de la ligne référencée 2 et parcourue dans le sens direct, de la ligne référencée 5 et parcourue dans le sens direct, de la ligne référencée 3 parcourue dans le sens indirect et de la ligne référencée 6 parcourue dans le sens direct.

Remarque importante : Quelles que soit leurs natures géométriques, deux lignes ne peuvent avoir le même identifiant. De même pour deux contours.

3.5 Surfaces et contours surfaciques

Il existe 2 types de surfaces : les surfaces planes, et les surfaces dites réglées. Elles s'appuient toutes deux sur des contours fermés et ne sont pas orientées. Les surfaces réglées sont des surfaces non planes définies à partir de la forme de leur contour. La seule option que l'on peut passer est un forçage de la construction de la surface comme portion de sphère.

```
Plane Surface(3)={3,4};
Ruled Surface(4)={1};
Ruled Surface(5)={10} In Sphere {1};
Surface Loop(3)={3,4,5};
```

La surface plane référencée 3 est définie à partir des contours référencés 3 et 4. Il faut bien entendu que ces contours soit contenus dans un même plan. On définit ainsi une surface possédant un trou. La surface reglée référencée 4 est définie à partir du contour référencé 1. La surface réglée rérérencée 5 est définie comme un élément de sphère de centre le point référencé 1, à partir du contour référencé 10. Enfin, le contour surfacique référencé 3 est défini comme l'union des 3 surfaces précédentes.

Remarque importante : Quelles que soit leurs natures géométriques, deux surfaces ne peuvent avoir le même identifiant. De même pour deux contours surfaciques.

3.6 Volumes

Par analogie, les volumes sont définis à partir de contours surfaciques.

```
Volume(1)={3};
```

3.7 Éléments physiques

Si l'on regarde la définition du format de maillage msh, on voit qu'il existe pour chaque élément une référence physique et une référence géométrique. Les identifiants que l'on a manipulés jusque là sont les références géométriques des éléments créés.

Les références physiques sont les identifiants des objets physiques du maillage. Ce sont ces références qui permettent de définir les domaines mathématiques utilisés dans l'établissement des formulations variationnelles. Ces éléments physiques sont définis à partir d'un ensemble d'éléments géométriques de même dimension.

```
PhysicalPoint(1)={1,2,4};
PhysicalSurface(2)={32,89};
PhysicalVolume(3)={3};
```

3.8 Transformations géométriques

Je vais ici parler de quelques commandes plus avancées pour la manipulation d'objets . Je vous renvoie pour cela à la documentation officielle.

Dilate ... : cette commande permet de réaliser une homothétie sur un ensemble d'entités géométriques.

Rotate ... : cette commande permet d'efectuer une rotation sur un ensemble d'entités géométriques.

Translate ... : cette commande permet d'effectuer une translation sur un ensemble d'entités géométriques.

Symmetry ... : cette commande permet d'effectuer une symétrie plane sur un ensemble d'entités géométriques.

Duplicata ... : cette commande génère un duplicata d'un ensemble d'entités géométriques. Très utile en argument des commandes précédentes pour conserver l'ensemble à transformer.

Extrude ... : cette commande permet de générer des lignes, surfaces ou volumes par extrusion de points, lignes ou surfaces respectivement. L'extrusion peut être obtenue par une translation et/ou une rotation et concerne également le maillage de l'objet source. Voir les exemples en sous-section 4.5.

3.9 Transformations du maillage

Par défaut, Gmsh génère des maillages non structurés : des triangles en 2D, des tétraèdres en 3D. Il est possible de structurer le maillage. Pour cela, on fait appel à l'algorithme transfinite 1D, 2D ou 3D pour transformer le maillage d'une ligne, d'une surface ou d'un volume en un maillage triangulaire/tétraédrique plus régulier. Par ailleurs, il est possible de générer un maillage quadrangulaire/hexaédrique en appelant également la commande de recombinaison.

```
Transfinite Line {1,3}=10;
Transfinite Surface ".";
Recombine Surface {2,3};
Transfinite Volume ".";
```

Les commandes **Transfinite** prennent en argument la liste des éléments dont on veut modifier le maillage. Si on les veut tous, on peut se contenter d'écrire "*". On peut également préciser le nombre de nœuds générés par la restructuration. C'est obligatoire pour les lignes.

La commande **Recombine Surface** associée à **Transfinite** convertit une fraction des triangles/tétraèdres en quadrangles/hexaèdres.

Remarque importante : Lorsqu'on utilise la commande **Recombine Surface**, on est obligé d'appeler la commande **Transfinite Volume**. En effet, dans le cas contraire, le mailleur va essayer de générer des tétraèdres, ce qui est impossible quand on dispose de quadrangles sur certaines surfaces!!!

Il existe beaucoup d'autres commandes dont je ne parlerai pas, permettant de générer d'autres types de lignes, d'effectuer d'autres manipulations sur les entités géométriques et les maillages, comme supprimer, masquer, sauvegarder, ... Je vous renvoie pour cela à la documentation officielle.

Je vais maintenant vous présenter quelques exemples, illustrant les différentes commandes présentées et leur utilisation :

1. Un maillage non structuré dans un domaine 2D rectangulaire contenant un trou elliptique.
2. Deux maillages structurés dans un domaine 2D rectangulaire plein.
3. Un maillage non structuré dans un domaine 3D de la forme d'un conesphère.
4. Un maillage non structuré dans un domaine 3D de la forme d'une sphère.
5. Un maillage non structuré dans un domaine 3D cylindrique.
6. Un maillage structuré prismatique dans un domaine 3D prismatique.
7. Un maillage structuré contenant des mailles pyramidales dans un domaine 3D conique.

4 Exemples

4.1 Maillage non structuré d'un domaine rectangulaire contenant un trou

Cet exemple permet d'appréhender les différentes instructions de base pour générer des maillages surfaciques.

```
lc=0.05;

// generation du rectangle
Point(1)={0,0,0,lc};
Point(2)={2,0,0,lc};
Point(3)={2,1,0,lc};
Point(4)={0,1,0,lc};

Line(1)={1,2};
Line(2)={2,3};
Line(3)={3,4};
Line(4)={4,1};

// generation de l'ellipse
Point(5)={1,0.5,0,lc};
Point(6)={1.5,0.5,0,lc};
Point(7)={1,0.75,0,lc};
Point(8)={0.5,0.5,0,lc};
Point(9)={1,0.25,0,lc};

Ellipse(5)={6,5,6,7};
Ellipse(6)={7,5,8,8};
Ellipse(7)={8,5,8,9};
Ellipse(8)={9,5,6,6};

// generation des contours
Line Loop(1)={1,2,3,4};
Line Loop(2)={5,6,7,8};

// generation de la surface
Plane Surface(1)={1,2};
```

Remarque : En donnant une valeur plus petite à la longueur caractéristique associée aux points 2 et 3, le maillage sera graduellement plus raffiné dans la partie droite.

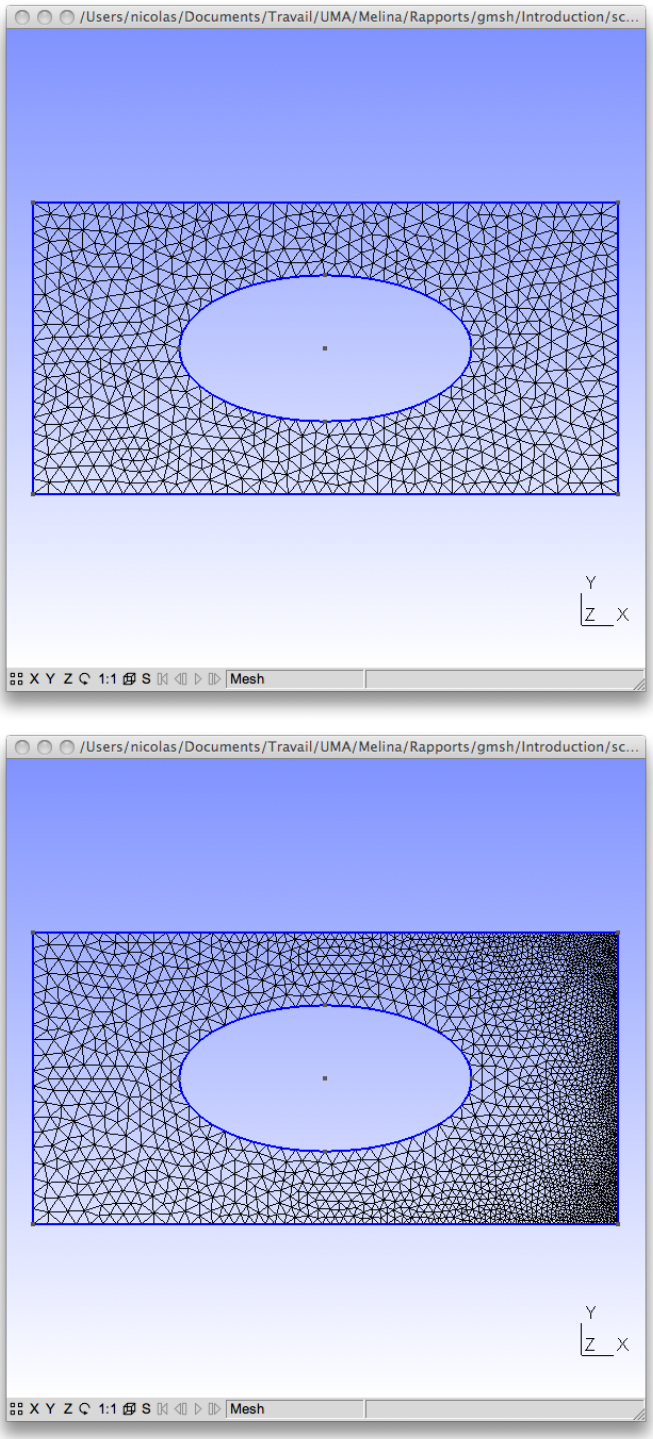

FIGURE 8 – Visualisation du maillage généré sur un domaine rectangulaire contenant un trou elliptique, sans et avec raffinement dans la partie droite

4.2 Maillages structurés d'un domaine rectangulaire

Cet exemple vise à montrer l'influence des commandes **Transfinite** et **Recombine Surface** sur la structure du maillage dans un domaine rectangulaire. Dans un domaine cubique, les mailles seront des tétraèdres réguliers avec **Transfinite**, et des cubes si l'on ajoute **Recombine Surface**.

```
lc = 0.05;

// generation du rectangle
Point(1) = {0,0,0,lc};
Point(2) = {2,0,0,lc};
Point(3) = {2,1,0,lc};
Point(4) = {0,1,0,lc};

Line(1) = {1,2};
Line(2) = {2,3};
Line(3) = {3,4};
Line(4) = {4,1};

// generation de la surface
Line Loop(1) = {1,2,3,4};
Plane Surface(1) = {1};

// structuration du maillage
Transfinite Line {1,3} = 40;
Transfinite Line {2,4} = 20;
Transfinite Surface "*";

// maillage quadrangulaire
// Recombine Surface "*";
```

Remarque : Dans cet exemple, on spécifie 40 nœuds sur l'union des lignes 1 et 3 en argument de Transfinite Line. Il ne s'agit pas de 40 nœuds pour la ligne 1 et 40 points pour la ligne 3, mais de 40 nœuds répartis sur les deux lignes, soit 20 nœuds pour la ligne 1 et 20 nœuds pour la ligne 3. Même raisonnement pour les 20 nœuds sur les lignes 2 et 4.

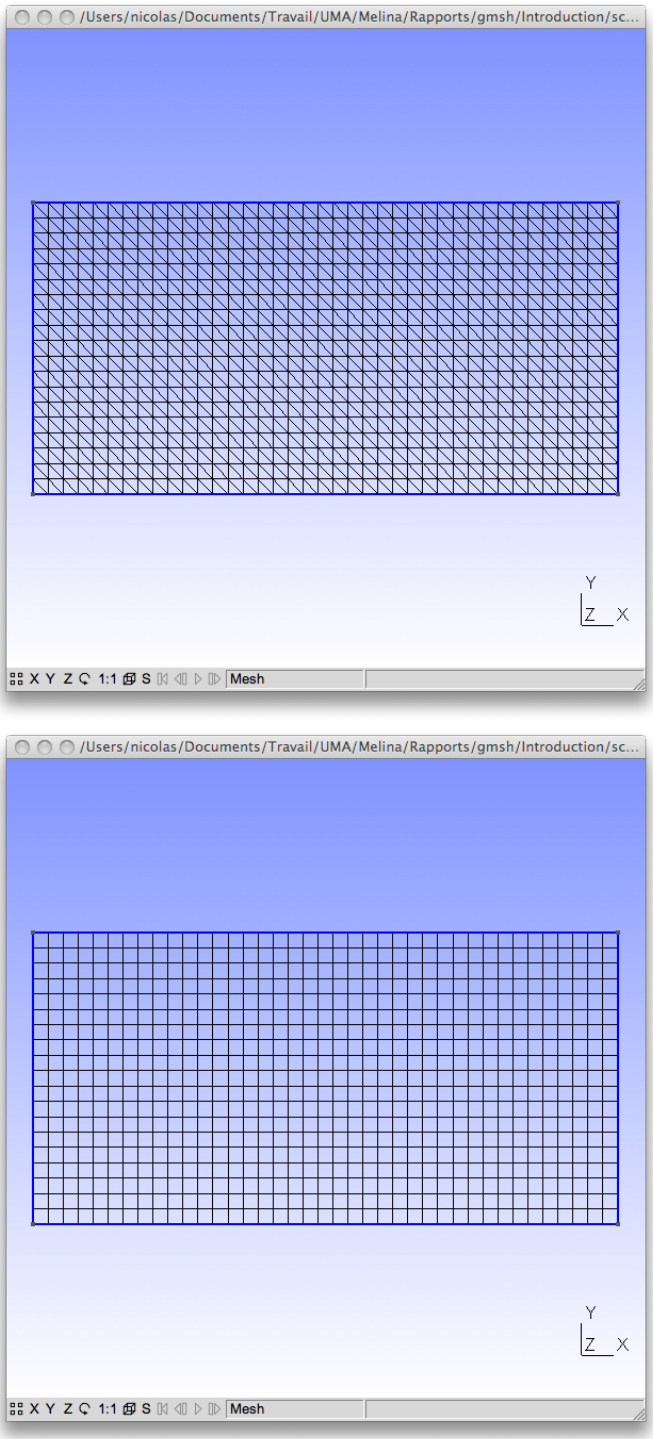

FIGURE 9 – Visualisation du maillage généré sur un domaine rectangulaire avec transfinite (à gauche) et avec transfinite et recombine (à droite).

4.3 Maillage non structuré d'un cônesphère

Cet exemple vise à montrer comment générer un maillage volumique composé d'éléments non polyédriques. Conformément à la démo proposée dans GMSH, on définit la sphère à l'aide du cercle principal et de deux demi-cercles orthogonaux 2 à 2, définissant ainsi les 4 éléments de la surface de la demi-sphère à l'aide des surfaces réglées. Même procédé pour le cône.

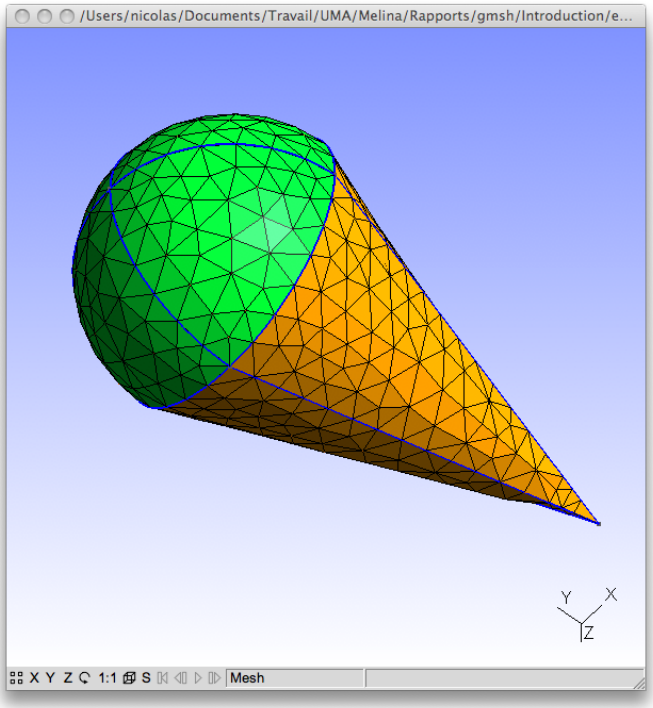

FIGURE 10 – Visualisation du maillage généré dans un cônesphère, défini comme l'union d'un cône de révolution et d'une demi-sphère.

```
lc=0.2;

// definition de la demi-sphere
Point(1)={0.0,0.0,0.0,lc};
Point(2)={1,0.0,0.0,lc};
Point(3)={0,1,0.0,lc};
Circle(1)={2,1,3};

Point(4)={-1,0,0.0,lc};
Circle(2)={4,1,3};

Point(5)={0,0,-1,lc};
```

```
Circle(3)={3,1,5};
Circle(4)={5,1,4};
Circle(5)={5,1,2};

Point(6)={0,0,1,lc};
Circle(6)={3,1,6};
Circle(7)={6,1,2};
Circle(8)={6,1,4};

// definition du cone
Point(7)={0,-3,0,lc};
Line(10)={7,2};
Line(12)={7,4};
Line(14)={7,5};
Line(16)={7,6};

// definition des surfaces de contour de la demi-
   sphere
Line Loop(20)={2,6,8};
Ruled Surface(21)={20} In Sphere{1};
Line Loop(22)={6,7,1};
Ruled Surface(23)={22} In Sphere{1};
Line Loop(24)={4,2,3};
Ruled Surface(25)={24} In Sphere{1};
Line Loop(26)={1,3,5};
Ruled Surface(27)={26} In Sphere{1};

// definition des surfaces de contour du cone
Line Loop(30)={-10,5,14};
Ruled Surface(31)={30};
Line Loop(32)={14,4,-12};
Ruled Surface(33)={32};
Line Loop(34)={16,8,-12};
Ruled Surface(35)={34};
Line Loop(36)={16,7,-10};
Ruled Surface(37)={36};

// definition du volume
Surface Loop(50)={31,33,35,37,21,25,27,23};
Volume(51)={50};
```

4.4 Maillage non structuré d'une sphère

Dans l'exemple précédent, nous avons pu voir que pour générer une demi-sphère, nous avions défini le cercle principal, plus 2 demi-cercles, de sorte à définir la demi-sphère en 4 morceaux à l'aide des surfaces réglées. Ceci est en effet basé sur la démo de la sphère disponible dans l'arborescence de Gmsh, basée sur la construction de 3 cercles orthogonaux 2 à 2. Il est toutefois possible de simplifier la procédure en ne définissant que 2 cercles orthogonaux.

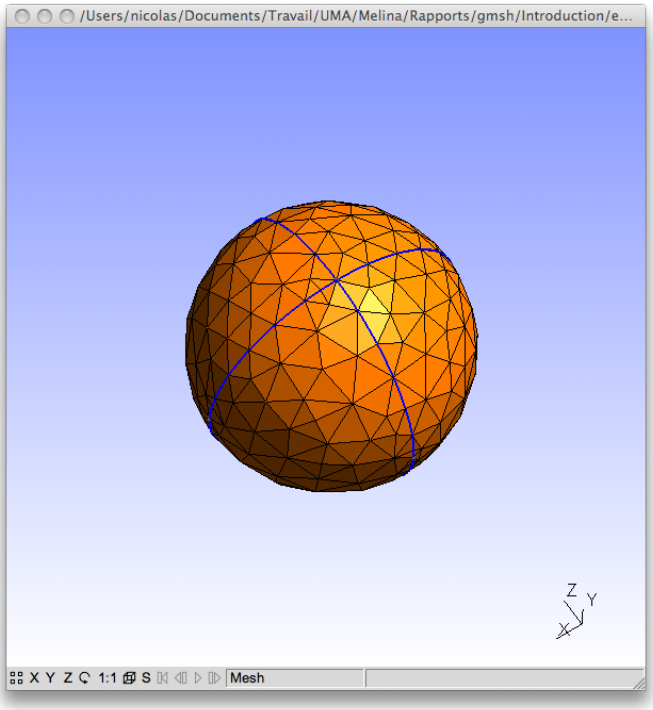

Figure 11 – Visualisation du domaine sphérique.

```
lc=0.25;

Point(1)={0,0,0,lc};
Point(2)={1,0,0,lc};
Point(3)={0,1,0,lc};
Point(4)={-1,0,0,lc};
Point(5)={0,-1,0,lc};
Point(6)={0,0,1,lc};
Point(7)={0,0,-1,lc};
```

```
Circle(1)={2,1,3};
Circle(2)={3,1,4};
Circle(3)={4,1,5};
Circle(4)={5,1,2};
Circle(5)={6,1,3};
Circle(6)={3,1,7};
Circle(7)={7,1,5};
Circle(8)={5,1,6};

Line Loop(1)={1,6,7,4};
Line Loop(2)={1,-5,-8,4};
Line Loop(3)={2,3,-7,-6};
Line Loop(4)={2,3,8,5};

Ruled Surface(1)={1} In Sphere{1};
Ruled Surface(2)={2} In Sphere{1};
Ruled Surface(3)={3} In Sphere{1};
Ruled Surface(4)={4} In Sphere{1};
Surface Loop(1)={1,2,3,4};
Volume(1)={1};
```

Une autre idée serait de définir un seul cercle et de définir la sphère par extrusion de ce cercle par rotation, mais pour des raisons évoquées en sous-sous-section 4.5.2, cela n'est pas possible.

4.5 Maillages structurés par extrusion

4.5.1 Maillage prismatique dans un prisme

Cet exemple vise à montrer comment générer un maillage prismatique régulier, à l'aide des commandes **Extrude** et **Recombine** dans un domaine prismatique. La commandes **Layers** permet dans cet exemple de spécifier non seulement que le maillage surfacique sera lui-aussi l'objet de la transformation, mais aussi le nombre de surfaces créées par extrusion. Il s'agit ici d'extrusion par translation.

```
lc=0.1;

// definition du triangle de base
Point(1)={0,0,0,lc};
Point(2)={1,0,0,lc};
Point(3)={1,1,0,lc};

Line(1)={1,2};
Line(2)={2,3};
```

```
Line(3)={3,1};

// definition de la surface de base
Line Loop(1)={1,2,3};
Plane Surface(1)={1};

// generation du volume par extrusion par translation
Extrude {0,0,-1}{Surface{1}; Layers{8}; Recombine;}
```

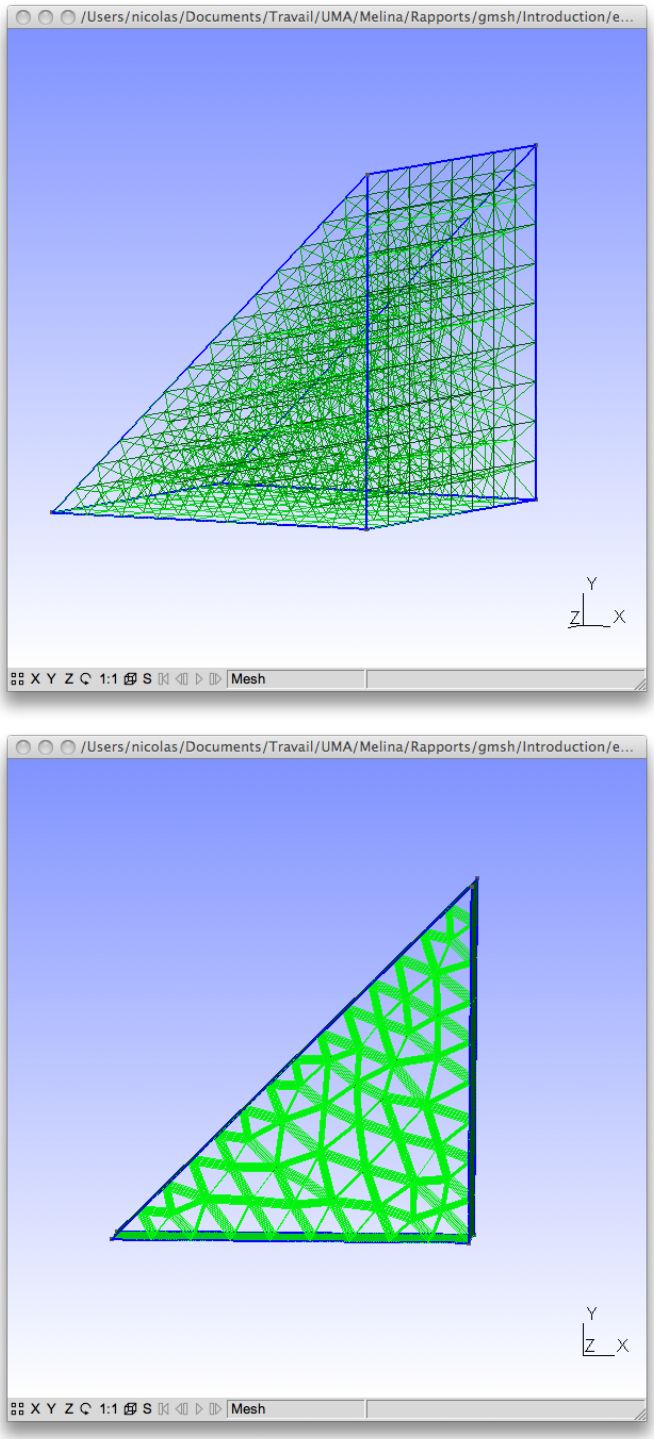

FIGURE 12 – Visualisation du maillage prismatique généré dans un prisme sous 2 angles de vue.

4.5.2 Maillage structuré (?) dans un cône

Cet exemple vise à montrer comment générer des mailles pyramidales dans un cône. À l'inverse des prismes, il n'est pas possible de générer un maillage constitué uniquement de mailles pyramidales. Dans cet exemple, on va trouver des tétraèdres, des prismes et des pyramides. Il s'agit ici d'extrusion par rotation.

```
lc=0.2;

// definition du triangle de base
Point(1)={0,0,0,lc};
Point(2)={1,0,0,lc};
Point(6)={0,0,3,lc};

Line(1)={1,2};
Line(2)={2,6};
Line(3)={6,1};

// definition de la surface
Line Loop(4)={2,3,1};
Plane Surface(5)={4};

// definition du volume par extrusion par rotations
   successives
Extrude { {0,0,1}, {0,0,0}, 2*Pi/3} {Surface{5};
   Layers{24}; Recombine;}
Extrude { {0,0,1}, {0,0,0}, 2*Pi/3} {Surface{17};
   Layers{24}; Recombine;}
Extrude { {0,0,1}, {0,0,0}, 2*Pi/3} {Surface{29};
   Layers{24}; Recombine;}
```

Remarque : Comme nous avons pu le voir avec les commandes de transformation géométrique, Gmsh numérote lui-même les éléments créés. Ici, on aurait pu s'attendre que la dernière surface générée par extrusion ait une référence décalée du nombre de calques (ici, 5+16=21). Or, le décalage est de 12, quelle que soit la valeur passée en arguments de **Layers**.

Remarque 2 : La valeur de l'angle de rotation n'est pas anodine. Si elle peut être négative ou supérieure à π en théorie, il faut savoir qu'en interne, cet angle est ramené modulo 2π dans l'intervalle $]-\pi, \pi]$. Par exemple, écrire $3\pi/2$ ou $-\pi/2$ donne le même résultat.

Par ailleurs, il n'est pas possible de générer le cône avec 2 extrusions d'angle π, il ne génère qu'un demi-cône. C'est la raison pour laquelle nous avons procédé en 3 étapes.

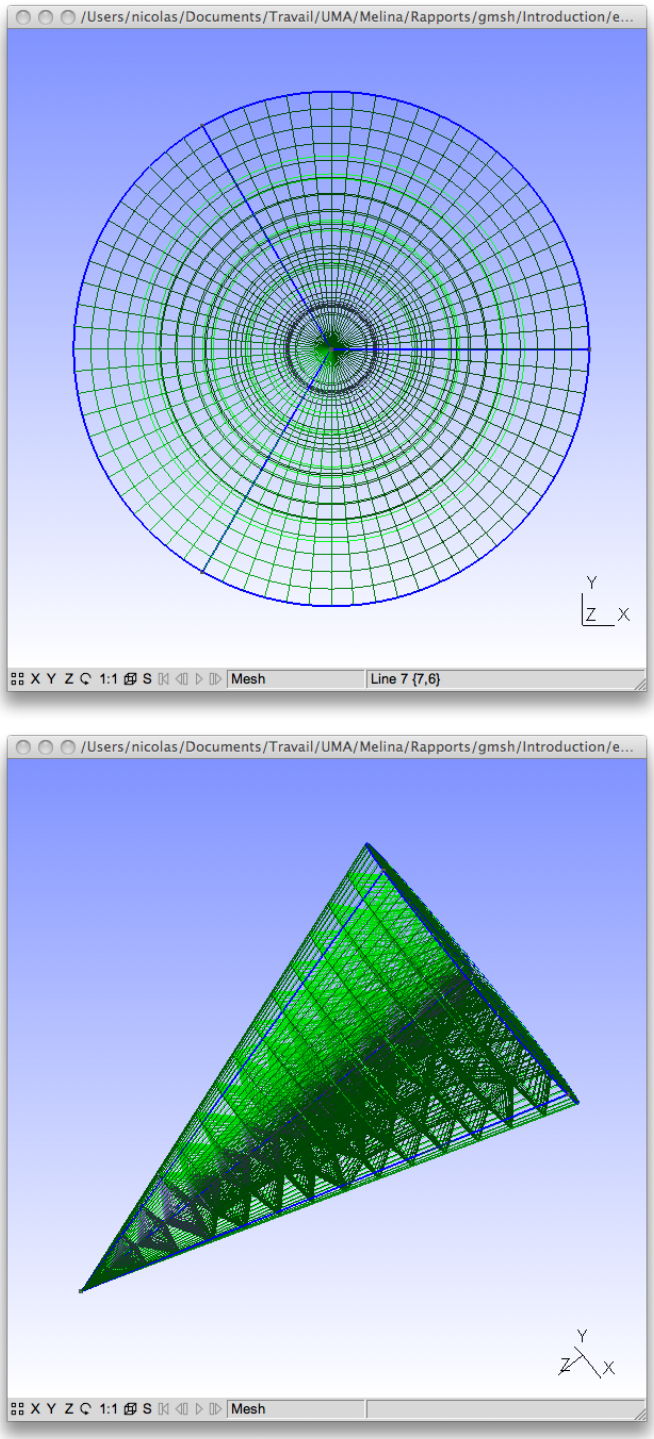

FIGURE 13 – Visualisation du maillage régulier généré dans un cône sous 2 angles de vue.

4.6 Maillage non structuré d'un cylindre contenant 2 tuyères

Cet exemple plus complexe vise à présenter l'utilisation des commandes de transformation géométrique (translation, symétrie, homothétie, copie) afin de générer un domaine de type cylindre contenant 2 trous de type tuyères, une dans chaque sens.

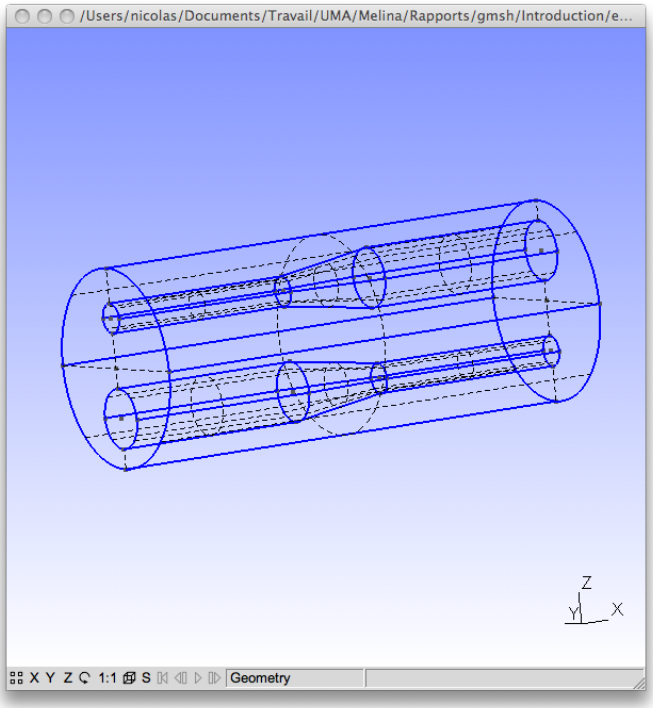

FIGURE 14 – Visualisation du domaine cylindrique comportant 2 tuyères.

On commence par générer un cercle dont on translate une copie afin de générer le cylindre principal. Par un procédé similaire, on définit la première tuyère, en utilisant astucieusement l'homothétie afin de définir le rétrécissement. La deuxième tuyère n'est autre que la symétrie plane de la première que l'on translate (on aurait pu effectuer 2 symétries planes).

On remarque au passage les contraintes liées à l'utilisation des commandes **Translate**, **Symmetry**, **Duplicata** et **Dilate** en terme de numérotation des nœuds générés. En effet, si tout se passe logiquement pour les lignes (segments et arcs de cercle), en ayant une numérotation continue, qui commence au premier entier disponible, il est en autrement pour les points. J'ai donc ajouté en commentaire les références des points générés. Pour les obtenir, il suffit de pointer chaque objet avec la souris dans la fenêtre graphique de GMSH.

```
lc=0.3;

// ---------- cylindre principal ----------
Point(1)={0,0,0,lc};
Point(2)={0,0,2,lc};
Point(3)={0,2,0,lc};
Point(4)={0,0,-2,lc};
Point(5)={0,-2,0,lc};

Circle(1)={2,1,3};
Circle(2)={3,1,4};
Circle(3)={4,1,5};
Circle(4)={5,1,2};

Line Loop(1)={1,2,3,4}; // cercle de base
Translate {10,0,0}{
  Duplicata {Line{1,2,3,4};}
}
// refs points 6,7,8,13,18
// refs circle 5,6,7,8

Line(9)={2,6};
Line(10)={3,8};
Line(11)={4,13};
Line(12)={5,18};

Line Loop(2)={5,6,7,8}; // cercle de base oppose
Line Loop(3)={1,10,-5,-9}; // face du cylindre
Line Loop(4)={2,11,-6,-10}; // idem
Line Loop(5)={3,12,-7,-11}; // idem
Line Loop(6)={4,9,-8,-12}; // idem

// ---------- premiere tuyere ----------
// premier troncon
Point(20)={0,0,-1,lc};
Point(21)={0,0,-0.4,lc};
Point(22)={0,0.6,-1,lc};
Point(23)={0,0,-1.6,lc};
Point(24)={0,-0.6,-1,lc};

Circle(20)={21,20,22};
Circle(21)={22,20,23};
Circle(22)={23,20,24};
```

```
Circle(23)={24,20,21};

Line Loop(20)={20,21,22,23};
Translate {4,0,0}{
  Duplicata {Line{20,21,22,23};}
}
// refs points 25,26,27,32,37
// refs circle 24,25,26,27

Line(28)={21,25};
Line(29)={22,27};
Line(30)={23,32};
Line(31)={24,37};

Line Loop(21)={20,29,-24,-28};
Line Loop(22)={21,30,-25,-29};
Line Loop(23)={22,31,-26,-30};
Line Loop(24)={23,28,-27,-31};

// deuxieme troncon
Dilate {{6,0,-1},0.5}{
  Translate {2,0,0}{
    Duplicata {Line{24,25,26,27};}
  }
}
// refs points 38,39,40,45,50
// refs circle 32,33,34,35

Line(36)={25,38};
Line(37)={27,40};
Line(38)={32,45};
Line(39)={37,50};

Line Loop(25)={24,37,-32,-36};
Line Loop(26)={25,38,-33,-37};
Line Loop(27)={26,39,-34,-38};
Line Loop(28)={27,36,-35,-39};

// troisieme troncon
Translate {4,0,0}{
  Duplicata {Line{32,33,34,35};}
}
// refs points 51,52,53,58,63
// refs circle 40,41,42,43
```

```
Line(44)={38,51};
Line(45)={40,53};
Line(46)={45,58};
Line(47)={50,63};

Line Loop(29)={40,41,42,43};
Line Loop(30)={32,45,-40,-44};
Line Loop(31)={33,46,-41,-45};
Line Loop(32)={34,47,-42,-46};
Line Loop(33)={35,44,-43,-47};

// ---------------- deuxieme tuyere ----------------
Symmetry {1,0,0,-5}{Translate {0,0,2}{Duplicata {Line
    {20,21,22,23,24,25,26,27,28,29,30,31,32,33,34,35,36,
     37,38,39,40,41,42,43,44,45,46,47};}
  }
}

// premier troncon
// refs points 64,65,66,71,76,84,85,86,91,96
// refs circle 48,49,50,51,52,53,54,55
// refs lignes 56,57,58,59
Line Loop(40)={48,49,50,51};
Line Loop(41)={48,-56,-52,57};
Line Loop(42)={49,-57,-53,58};
Line Loop(43)={50,-58,-54,59};
Line Loop(44)={51,-59,-55,56};

// deuxieme troncon
// refs points 120,121,122,127,132
// refs circle 60,61,62,63
// refs lignes 64,65,66,67
Line Loop(45)={52,-64,-60,65};
Line Loop(46)={53,-65,-61,66};
Line Loop(47)={54,-66,-62,67};
Line Loop(48)={55,-67,-63,64};

// troisieme troncon
// refs points 156,157,158,163,168
// refs circle 68,69,70,71
// refs lignes 72,73,74,75
Line Loop(49)={68,69,70,71};
Line Loop(50)={60,-72,-68,73};
```

```
Line Loop(51)={61,-73,-69,74};
Line Loop(52)={62,-74,-70,75};
Line Loop(53)={63,-75,-71,72};

// ---------- def des surfaces ----------
// bases du cylindre principal
Plane Surface(1)={1,20,49};
Plane Surface(2)={2,29,40};

// faces du cylindre principal
Ruled Surface(3)={3};
Ruled Surface(4)={4};
Ruled Surface(5)={5};
Ruled Surface(6)={6};

//faces de la premiere tuyere
Ruled Surface(21)={21};
Ruled Surface(22)={22};
Ruled Surface(23)={23};
Ruled Surface(24)={24};
Ruled Surface(25)={25};
Ruled Surface(26)={26};
Ruled Surface(27)={27};
Ruled Surface(28)={28};
Ruled Surface(30)={30};
Ruled Surface(31)={31};
Ruled Surface(32)={32};
Ruled Surface(33)={33};

//faces de la deuxieme tuyere
Ruled Surface(41)={41};
Ruled Surface(42)={42};
Ruled Surface(43)={43};
Ruled Surface(44)={44};
Ruled Surface(45)={45};
Ruled Surface(46)={46};
Ruled Surface(47)={47};
Ruled Surface(48)={48};
Ruled Surface(50)={50};
Ruled Surface(51)={51};
Ruled Surface(52)={52};
Ruled Surface(53)={53};

// def du volume
```

```
Surface Loop(1)={1,2,3,4,5,6}; // cylindre principal
Surface Loop(2)={21,22,23,24,25,26,27,28,30,31,32,33};
    // tuyere 1
Surface Loop(3)={41,42,43,44,45,46,47,48,50,51,52,53};
    // tuyere 2
Volume(1)={1,2,3};
```

Une façon de contourner ces contraintes est d'utiliser les fonctionnalités avancées du langage geo : On peut définir une macro qui dessine un cylindre à partir d'un cercle dont on donne soit les coordonnées du centre et du rayon, soit les numéros des 5 points et des 4 arcs de cercle qui le composent. Ce cylindre peut être droit ou non. On construit par ailleurs à la volée les listes de numéros géométriques nécessaires à la définition du volume, profitant également de la syntaxe complète des fonctions `Dilate` et `Translate` pour obtenir de manière automatique les numéros des objets créées. Il suffit pour cela de mettre en argument de la fonction `Duplicata` tous les objets dont on veut leur duplication, en particulier les points, qui faisaient l'objet de la critique de la version précédente. Dupliquer les surfaces évite par exemple de reconstruire les `Line Loop`. L'ordre dans lequel les éléments sont passés à `Duplicata` sera celui des éléments dupliqués et transformés dans le tableau résultat.

```
Function MyCylinder
  If (mode == _notbuilt)
    P1=newp; Point(P1)={xc,yc,zc,lc};
    P2=newp; Point(P2)={xc,yc,zc+r,lc};
    P3=newp; Point(P3)={xc,yc+r,zc,lc};
    P4=newp; Point(P4)={xc,yc,zc-r,lc};
    P5=newp; Point(P5)={xc,yc-r,zc,lc};

    C1=newc; Circle(C1)={P2,P1,P3};
    C2=newc; Circle(C2)={P3,P1,P4};
    C3=newc; Circle(C3)={P4,P1,P5};
    C4=newc; Circle(C4)={P5,P1,P2};

    points[p]=P1; points[p+1]=P2;
    points[p+2]=P3; points[p+3]=P4;
    points[p+4]=P5; p+=5;
    curves[c]=C1; curves[c+1]=C2;
    curves[c+2]=C3; curves[c+3]=C4; c+=4;

    loops[t]=newc;
    Line Loop(loops[t])={C1,C2,C3,C4}; t++;
  EndIf
```

```
If (mode == _built)
  P1=inpts[0];P2=inpts[1];P3=inpts[2];P4=inpts[3];P5
     =inpts[4];
  C1=incirc[0];C2=incirc[1];C3=incirc[2];C4=incirc
     [3];
EndIf

// 2eme cercle de base
If (alpha == 1)
  out[]=Translate{h,0,0}{Duplicata{Point{P1,P2,P3,P4
     ,P5};Line{C1,C2,C3,C4};}};
EndIf
If (alpha != 1)
  out[]=Dilate{ {xc+h,0,zc},alpha}{Translate {h
     ,0,0}{Duplicata{Point{P1,P2,P3,P4,P5};Line{C1,
     C2,C3,C4};}}};
EndIf

P6=out[0];P7=out[1];P8=out[2];P9=out[3];P10=out[4];
C5=out[5];C6=out[6];C7=out[7];C8=out[8];

loops[t]=newc;
Line Loop(loops[t])={C5,C6,C7,C8};t++;

// faces laterales
L1=newc;Line(L1)={P2,P7};
L2=newc;Line(L2)={P3,P8};
L3=newc;Line(L3)={P4,P9};
L4=newc;Line(L4)={P5,P10};

points[p]=P6;points[p+1]=P7;
points[p+2]=P8;points[p+3]=P9;
points[p+4]=P10;p+=5;
curves[c]=C5;curves[c+1]=C6;
curves[c+2]=C7;curves[c+3]=C8;c+=4;
curves[c]=L1;curves[c+1]=L2;
curves[c+2]=L3;curves[c+3]=L4;c+=4;

loops[t]=newc;
Line Loop(loops[t])={C1,L2,-C5,-L1};
surfs[s]=news;
Ruled Surface(surfs[s])={loops[t]};s++;t++;
loops[t]=newc;
Line Loop(loops[t])={C2,L3,-C6,-L2};
```

```
  surfs[s]=news;
  Ruled Surface(surfs[s])={loops[t]};s++;t++;
  loops[t]=newc;
  Line Loop(loops[t])={C3,L4,-C7,-L3};
  surfs[s]=news;
  Ruled Surface(surfs[s])={loops[t]};s++;t++;
  loops[t]=newc;
  Line Loop(loops[t])={C4,L1,-C8,-L4};
  surfs[s]=news;
  Ruled Surface(surfs[s])={loops[t]};s++;t++;

Return
```

Quelques mots sur la macro `MyCylinder` :

Arguments d'entrée Cette remarque sera d'autant plus pertinente pour celles et ceux qui sont familiers avec les langages de programmation et en particulier avec la notion de variable globale / locale : toutes les variables définies dans un script geo sont globales. Il n'y a pas de notion de variables d'entrées ni de sortie. Tout est transparent. Néanmoins, l'argument d'entrée principal de la macro `MyCylinder` est `mode`. Il permet de définir le premier cylindre de base à partir des coordonnées du centre du cercle et de son rayon, à savoir les arguments d'entrée `xc`, `yc`, `zc` et `r`, ou de le définir à partir de la donnée des 5 points et des 4 arcs de cercle, à savoir les arguments d'entrée `inpts` et `incirc`. Le dernier arguments d'entrée est `alpha` qui détermine le rayon du second cercle (αr).

Arguments de sortie Puisque toutes les variables définies dans ou hors la macro `MyCylinder` sont globales, toutes les variables peuvent être considérées comme des arguments de sortie. Certaines d'entre elles ne sont pas utilisées dans ce sens. Je vais donc parler des autres : `loops` qui stocke le numéro des `Line Loop` pour pouvoir construire les surfaces et le volume dans un deuxième temps, `points`, `curves` et `surfs`, qui stockent respectivement les numéros des points, des courbes et des surfaces créés, et les données du second cercle de base : les 5 points `P6`, `P7`, `P8`, `P9` et `P10`, et les 4 arcs de cercle `C5`, `C6`, `C7` et `C8`.

newp, newc, news Quelques mots sur ces macros fort utiles. Elles fournissent respectivement le premier numéro disponible pour définir respectivement un point, une ligne (Line, Circle, Line Loop, ...), ou une surface (Plane Surface, Ruled Surface, Surface Loop, ...). Il existe également la macro `newv` pour les volumes et une macro particulière `newreg` pour tout sauf les points.

Avec l'exploitation de cette macro, tout est automatique au niveau de la construction, excepté la construction des 2 surfaces de base du cylindre principal, puisqu'il faut la donnée des 3 contours circulaires, et que 2 d'entre eux ne sont pas construits. En comparaison de tout ce qui a pu être rendu automatisé, ce n'est quand même pas grand chose.

```
Include "cylindre_lib.geo";

_built=1.0;_notbuilt=0.0;
lc=0.3;
p=0;c=0;t=0;s=0;

// ---------- cylindre principal ----------

mode=_notbuilt;
xc=0.0;yc=0.0;zc=0.0;r=2.0;h=10;alpha=1.0;
Call MyCylinder;

For i In {0:s-1}
  msurfs[i]=surfs[i];
EndFor
mains=s;

// ---------- premiere tuyere ----------

// definition
mode=_notbuilt;
p=0;c=0;s=0;
xc=0.0;yc=0.0;zc=-1.0;r=0.3;h=4.0;alpha=1.0;
Call MyCylinder;

mode=_built;
inpts[]={P6,P7,P8,P9,P10};incirc[]={C5,C6,C7,C8};
xc+=h;h=2.0;alpha=2.0;
Call MyCylinder;

mode=_built;
inpts[]={P6,P7,P8,P9,P10};incirc[]={C5,C6,C7,C8};
xc+=h;h=4.0;alpha=1.0;
Call MyCylinder;

// ---------- deuxieme tuyere ----------
out[]=Symmetry {1,0,0,-5}{Translate {0,0,2}{Duplicata
    {Point{points[]};Line{curves[]};Surface{surfs
```

```
    []};}}};

For i In {0:p-1}
  points2[i]=out[i];
EndFor

For i In {0:c-1}
  curves2[i]=out[p+i];
EndFor

For i In {0:s-1}
  surfs2[i]=out[p+c+i];
EndFor

loops[t]=newc;
Line Loop(loops[t])={curves2[0],curves2[1],curves2[2],
    curves2[3]};t++;
loops[t]=newc;
Line Loop(loops[t])={curves2[c-8],curves2[c-7],curves2
    [c-6],curves2[c-5]};t++;

// ---------------------- volume ----------------------

// surface du cylindre principal
s=mains;
msurfs[s]=news;Plane Surface(msurfs[s])={loops[0],
    loops[6],loops[23]};s++;
msurfs[s]=news;Plane Surface(msurfs[s])={loops[1],
    loops[17],loops[22]};s++;

SL1=news;Surface Loop(SL1)={msurfs[]};
SL2=news;Surface Loop(SL2)={surfs[]};
SL3=news;Surface Loop(SL3)={surfs2[]};

Volume(1)={SL1,SL2,SL3};
```