Documentation de Convmesh 1.0

Nicolas Kielbasiewicz*, Eric Lunéville†
U.M.A.-P.O.e.m.s., E.N.S.T.A. ParisTech

15 mars 2011

Table des matières

1 Introduction

1.1 Présentation

Convmesh est un utilitaire écrit en C++ permettant de convertir

*nicolas.kielbasiewicz@ensta.fr
†eric.luneville@ensta.fr

différents formats de maillages. L'idée fondatrice est la manipulation de maillages au format .mel, le format propriétaire de la librairie éléments finis MELINA.

1.2 Installation

1.2.1 Téléchargement

CONVMESH est disponible sur :

http://www.ensta.fr/~kielbasi/convmesh/

Vous disposez ainsi d'une archive compressée des sources, au format tbz (tar+bzip2).

1.2.2 Compilation et installation

L'extraction de l'archive crée un répertoire de la forme `convmesh-vXXX` (`XXX` correspond au numéro de version) contenant l'ensemble des sources. Vous pouvez placer ce répertoire à l'endroit où vous le voulez dans votre arborescence, il suffira de compléter la variable d'environnement PATH.

1. Aller dans le répertoire `convmesh-vXXX`
2. Exécuter la commande `make`
3. L'exécutable `convmesh` est créé dans le sous-répertoire `bin`.
4. Pour vérifier votre compilation, exécuter la séquence de tests à l'aide de la commande `make test`.
5. Si les tests ont réussi, votre installation est correcte. Il vous reste à ajouter le sous-répertoire `bin` dans votre PATH.

1.3 Fonctionnalités

1.3.1 Ce que CONVMESH fait et ne fait pas

CONVMESH est un convertisseur de formats de maillages s'exécutant en ligne de commande. Il peut prendre en entrée un maillage au format mel ou un maillage au format msh (version 2.X), et peut produire en sortie un maillage au format msh (version 2.0), mel, amdba ou vtk (version 2.0 en données non structurées). Chacun de ces formats est explicité dans la section 2.

Les formats .amdba et .vtk n'étant pas des formats de maillages éléments finis, mais seulement des formats de maillages géométriques, il ne sont pas proposés en format d'entrée de CONVMESH.

Par ailleurs, pour la sortie au format .mel, CONVMESH met à disposition un certain nombre d'outils de manipulation de maillage, permettant ainsi

de renommer des domaines, d'extraire des sous-domaines ou de fissurer un domaine. Ces deux dernières fonctionnalités sont explicitées en section 3. Pour se servir de ces outils, on utilise l'option `-i` qui invoque le mode interactif.

1.3.2 Ce que CONVMESH fera

- Le procédé de fissuration ne fonctionne a priori que sur un maillage 2D. À terme, il fonctionnera aussi en 3D.
- Des 31 types de mailles référencée par le format msh, seul le tétraèdre d'ordre 5 (56 nœuds) n'est pas géré. Il sera incorporé dès que sa génération sera possible proprement avec GMSH.
- Les transformations s'appliqueront également pour un maillage de sortie au format msh.
- Les formats de sortie ayant des versions (msh ou vtk) seront modifiés pour être les plus récents.

1.4 Utilisation

L'utilisation basique de CONVMESH est de spécifier le fichier d'entrée et le fichier de sortie :

```
convmesh filename1.ext1 filename2.ext2
```

Si le nom des fichiers d'entrée et de sortie sont les mêmes, on peut se contenter de spécifier uniquement l'extension de sortie (donc le format) :

```
convmesh filename1.ext1 ext2
```

Il existe un certain nombre d'options explicités dans l'aide que l'on affiche avec la commande `convmesh -h`. En particulier, l'option `-i` permet d'invoquer le mode interactif et donc d'accéder aux outils de manipulation de maillage.

2 Formats de maillage

2.1 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).

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). Il peut aussi valoir 2 (ou 2.0), voire même 1.x si vous avez une version très anciennes de votre mailleur. Je vous conseille dans ce cas de le mettre à jour, car CONVMESH ne gère que les versions postérieures à 2.0.

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

`data-size` précise le nombre de décimales significatives d'un flottant double précision (=sizeof(double)). Il vaut donc très souvent 8.

Le format 2.2 n'étant pas encore stabilisé, je vais décrire le format 2.1.

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
```

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'élément fini (géométrie de la maille et ordre d'interpolation).

`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 le numéro de référence de l'entité physique à laquelle appartient l'élément, le second est le numéro de 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 le numéro de référence de la partition du maillage contenant cet élément.

`node-number-list` désigne la liste ordonnée des références des nœuds de l'élément.

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.

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

La dimension du domaine, contenue dans la variable `physical-domain` n'est présente que depuis la version 2.1 du format msh. Attention si vous utilisez le format 2.0!!!

Un exemple

Voici, page suivante, la version en format msh d'un unique élément cubique d'ordre 1, où `domain_2` désigne le domaine volumique et `domain_1` le domaine définissant le bord de `domain_2`.

```
$MeshFormat
2 0 8
$EndMeshFormat
$PhysicalNames
2
1 domain_1
2 domain_2
$EndPhysicalNames
$Nodes
8
1 0 0 0
2 1 0 0
3 1 1 0
4 0 1 0
5 0 0 1
6 0 1 1
7 1 1 1
8 1 0 1
$EndNodes
$Elements
7
1 5 3 2 0 0 1 2 3 4 5 6 7 8
2 3 3 1 0 0 2 6 5 1
3 3 3 1 0 0 7 6 2 3
4 3 3 1 0 0 5 6 7 8
5 3 3 1 0 0 3 4 8 7
6 3 3 1 0 0 8 4 1 5
7 3 3 1 0 0 1 4 3 2
$EndElements
```

2.2 Le format mel

Le format mel est un format de maillage propre à la librairie de calcul éléments finis MELINA. C'est un format qui contient en particulier des informations de formatage de lecture des données, le nombre d'éléments d'un type donné et la définition des domaines géométriques à partir de leurs éléments. Dans la mesure où il y a différentes possibilités d'écriture des informations, on utilisera une syntaxe spécifique pour désigner les éléments optionnels ($[\ldots]$) ou à choisir ($<\ldots \wedge \ldots>$). Par ailleurs, une ligne de donnée dans ce format ne doit pas dépasser 80 caractères. Si besoin, on continuera sur la ligne suivante.

En-tête

```
* ligne de commentaire
TITRE  nbligne
...
[ FORMAT [DE] LECTURE
    [ [DES] COORDONNEES fmtStr1 ]
    [ [DE LA] NUMEROTATION [GLOBALE] fmtStr2 ]
    [ < SANS ^ AVEC > COMMENTAIRE ]
]
```

Les variables *fmStr1* et *fmtStr2* représentent des chaines de caractères de format type Fortran. Si on ne précise pas ce bloc d'information, elles ont pour valeur par défaut respectivement '6E12.4' et '18I4'. On peut spécifier aussi un format libre : '*'.

Description des élements

La première partie concerne la définition globale du maillage.

```
DESCRIPTION GLOBALE [DU] [MAILLAGE]
    [ [NOM] [DES] VARIABLES [D' '] ESPACE : string ]
    NOMBRE [D' '] ELEMENTS : nb-elem ]
```

La variable *string* peut être 'X' ou 'X' 'Y' ou 'X' 'Y' 'Z'.

La deuxième partie concerne la définition des éléments. Ces éléments sont organisés en blocs qui représentent le type d'élément. Dans un premier temps, on définit chacun des blocs, puis, en respectant l'ordre de définition des blocs, on va définir les éléments.

Chaque définition de blocs a pour syntaxe générale :

```
BLOC [DE]
  < typeCellStr DE LAGRANGE [ AUX POINTS DE GAUSS-
    LOBATTO ]
      < typeInterpStr ^ < D'ORDRE ^ DE DEGRE > order >
  ^ [TYPE] [GEOMETRIQUE] typeFECellStr
  > : nb-bloc-elem ELEMENTS
```

- *typeCellStr* désigne le type géométrique des éléments. Il peut être HEXAEDRES, PRISMES, TETRAEDRES, QUADRANGLES, TRIANGLES, ou SEGMENTS
- *typeInterpStr* désigne l'interpolation éléments finis. Il peut donc être de la forme Pn, $n \leq 6$ ou Qm, $m \leq 10$

– *order* désigne l'ordre d'interpolation.
– *typeFECellStr* est un tétragramme désignant le type des éléments finis. Par exemple, 'TR02' désigne un triangle d'ordre 2.

Chaque élément est défini sur 2 lignes, avec les coordonnées de chaque nœud de l'élément, puis leur numéro de référence globale. Chaque élément est donc défini par :

```
...
[ligne de commentaire]
x1 [y1 [z1]] x2 [y2 [z2]] x3 [y3 [z3]] ...
refnode1 refnode2 refnode3 ...
...
```

Si l'on a spécifié dans le format de lecture 'AVEC COMMENTAIRE', ces deux lignes peuvent être précédées d'une ligne de commentaire.

Description des domaines physiques

L'une des particularité du format mel est de contenir la définition des domaines physiques à partir de l'ensemble des éléments qu'ils contiennent. La syntaxe pour un élément est l'une des formes suivantes :

```
E[LEMENT[S]] elem-id [ / elem-id2 ]
E[LEMENT] elem-id < F[ACE] ^ A[RETE] ^ P[OINT] > edge-
    id
```

Les variables *elem-id* et *elem-id2* désigne le numéro de référence globale d'un élément. *edge-id* désigne quant à lui le numéro de référence locale d'un bord de l'élément.

Quand on précise *elem-id2*, cela signifie que tous les éléments dont le numéro de référence globale est entre *elem-id* et *elem-id2* inclus appartiennent au domaine.

Le mot clé F[ACE] est utilisé pour des éléments 3D, A[RETE] pour des éléments 2D et P[OINT] pour des éléments 1D.

Un maillage mel se termine par le mot clé FIN.

Un exemple

Voici la version en format mel d'un unique élément cubique d'ordre 1, où `domain_2` désigne le domaine volumique et `domain_1` le domaine définissant le bord de `domain_2`.

```
TITRE 1
(generated by convmesh from cubeQ1.msh)
FORMAT DE LECTURE DES COORDONNEES '*'
       DE LA NUMEROTATION GLOBALE '*'
       SANS COMMENTAIRE
DESCRIPTION GLOBALE DU MAILLAGE
NOM DES VARIABLES D''ESPACE : 'X' 'Y' 'Z'
NOMBRE D''ELEMENTS : 1
BLOC DE TYPE GEOMETRIQUE HE01 : 1 ELEMENTS
0 0 0 1 0 0 1 1 0 0 1 0 0 0 1 0 1 1 1 1 1 1 0 1
1 2 3 4 5 6 7 8
DOMAINE 'domain_1'
E 1 F 1 E 1 F 2 E 1 F 3 E 1 F 4 E 1 F 5 E 1 F 6
DOMAINE 'domain_2'
ELEMENT 1
FIN
```

2.3 Le format amdba

Définition

Le format amdba est un format de maillage géométrique, donc seuls les éléments d'ordre 1 (définis uniquement par leurs sommets) sont gérés. Par ailleurs, tous les éléments sont nécessairement de même nature géométrique. Le format ne contient donc pas de description des éléments sur les bords.

```
nb-nodes nb-elem elem-type
n1 x1 [y1 [z1]] phys-ref1
n2 x2 [y2 [z2]] phys-ref2
....
e1 i1 j1 k1 ... e-phys-ref1
e2 i2 j2 k2 ... e-phys-ref2
...
```

- *nb-nodes* désigne le nombre de nœuds du maillage.
- *nb-elem* désigne le nombre de mailles.
- *elem-type* désigne le type de maille géométrique. Il peut donc être SEGMENT, TRIANGLE, QUADRANGLE, TETRAEDRE, HEXAEDRE, PRISME, ou PYRAMIDE. Ce mot clé permet de savoir le nombre de nœuds de l'élément.
- *n1, n2, ...* désignent le numéro de référence globale du nœud.
- *e1, e2, ...* désignent le numéro de référence globale de l'élément.

– *i1, j1, k1*, ... désignent les numéros de références globales des nœuds de l'élément 1.
– *phys-ref1, phys-ref2*, ... désignent le numéro de référence physique du nœud.
– *e-phys-ref1, e-phys-ref2*, ... désignent le numéro de référence physique de l'élément.

Un exemple

Voici la version en format amdba d'un unique élément cubique d'ordre 1.

```
8  1  HEXAEDRE
1  0  0  0  1
2  1  0  0  1
3  1  1  0  1
4  0  1  0  1
5  0  0  1  1
6  0  1  1  1
7  1  1  1  1
8  1  0  1  1
1  1  2  3  4  5  6  7  8  2
```

2.4 Le format vtk

Tout comme le format amdba, le format vtk est un format de maillage géométrique, donc seuls les éléments d'ordre 1 (définis uniquement par leurs sommets) sont gérés. On ne décrit ici que le format 2.0 tel que CONVMESH le génère.

L'en-tête

On génère un maillage texte sur une grille non structurée. On précise ces informations de la manière suivante :

```
# vtk DataFile Version 2.0
* ligne de titre
ASCII
DATASET UNSTRUCTURED_GRID
```

Définition des nœuds

```
POINTS nb-nodes datatype
x1 y1 z1
x2 y2 z2
...
```

- *nb-nodes* désigne le nombre de nœuds du maillage.
- *datatype* précise le type des coordonnées de chaque nœud. Il peut valoir bit, unsigned_char, char, unsigned_short, short, unsigned_int, int, unsigned_long, long, float, ou double.

Définition des éléments

Tout comme le format amdba, le format vtk ne définit pas les éléments sur les bords.

```
CELLS nb-cells size
nb-nodes n1 n2 ....
...
```

- *nb-cells* désigne le nombre de mailles.
- *size* désigne le nombre total d'informations à lire.
- *nb-nodes* désigne le nombre de nœuds de l'élément, à savoir le nombre d'information qu'il reste à lire sur la ligne.

Définition du type des éléments

Comme des éléments géométriques différents peuvent être définis par le même nombre de nœuds (prenez un quadrangle et un tétraèdre par exemple : 4 nœuds), le dernier bloc d'information spécifie le type de chaque élément. Par exemple, 3 désigne un segment, 5 désigne un triangle et 14 désigne une pyramide à base quadrangulaire.

```
CELL_TYPES nb-cells
t1
t2
...
```

Un exemple

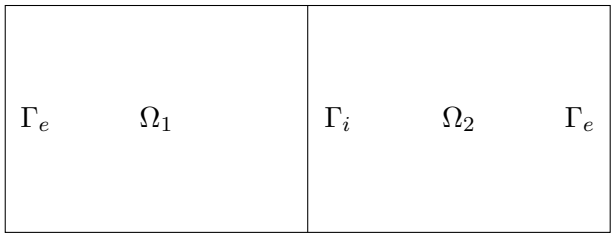
Voici la version en format vtk d'un unique élément cubique d'ordre 1.

```
# vtk DataFile Version 2.0
 (generated by convmesh from cubeQ1.msh)
ASCII
DATASET UNSTRUCTURED_GRID
POINTS 8 double
0 0 0
1 0 0
1 1 0
0 1 0
0 0 1
0 1 1
1 1 1
1 0 1

CELLS 1 9
8 0 1 2 3 4 5 6 7

CELL_TYPES 1
12
```

3 Transformations

3.1 Préciser le parent d'un domaine : Extraction

Considérons un domaine géométrique structuré de la manière suivante :

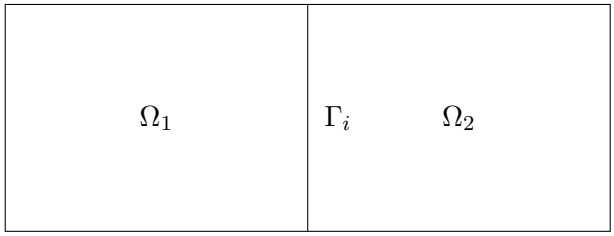


Les bords Γ_i et Γ_e sont partagés entre les deux domaines Ω_1 et Ω_2, le premier en tant qu'interface, le second en tant que bord extérieur englobant.

La technique d'extraction vise à pouvoir définir dans un maillage l'un des domaines $\Gamma_{e1} = \Gamma_e \cap \Omega_1$, $\Gamma_{e2} = \Gamma_e \cap \Omega_2$, $\Gamma_{i1} = \Gamma_i \cap \Omega_1$ et $\Gamma_{i2} = \Gamma_i \cap \Omega_2$.

3.2 Disjoindre un sous-domaine

Plaçons-nous dans le contexte suivant :

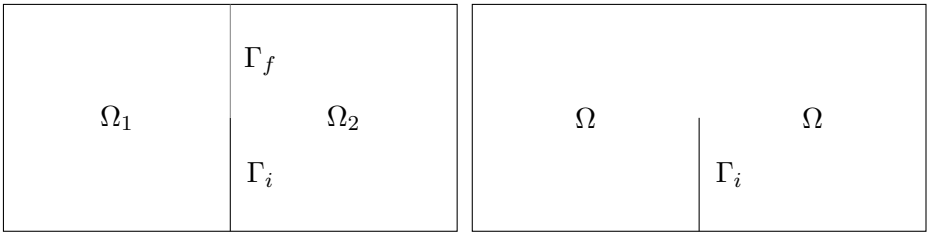


La technique de disjonction consiste à dupliquer le bord Γ_i et à les répartir tous les deux entre les domaines Ω_1 et Ω_2. Les nœuds de ces 2 exemplaires de Γ_i se superposent, mais n'ont pas la même référence globale.

La technique de disjonction consiste donc à détecter les nœuds sur le domaine à disjoindre, à les dupliquer, et à les distribuer de part et d'autre dans les mailles concernées.

3.3 Fissurer un sous-domaine

Plaçons-nous dans l'un des deux cas suivants :



La technique de fissuration consiste, sur le même principe que la technique de disjonction, à détecter les nœuds sur le domaine à fissurer, à les dupliquer, et à les distribuer de part et d'autre dans les mailles concernées.

Dans le premier cas, où la fissure est une partie de l'interface entre deux domaines physiques, la répartition se fait de la même manière qua pour la technique de disjonction. Dans le second cas, la fissure est à l'intérieur d'un même domaine. La répartition des nœuds de part et d'autre de la fissure, va donc se faire différemment, par un principe d'adjacence.

Il est à noter que le nœud en pointe de la fissure est lui aussi dupliqué.

4 F.A.Q.

Sont recensées ici la plupart des questions qui m'ont été posées suite à l'utilisation de Convmesh :

CONVMESH **est introuvable** Vérifiez que le chemin qui le contient fait bien partie de votre PATH. Si c'est le cas.

CONVMESH **ne s'execute pas** Vérifiez s'il n'a pas besoin d'être recompilé ou si vous n'avez pas une version trop ancienne. Contactez votre administrateur dans ce cas. Affichez l'aide de CONVMESH. Si l'aide s'affiche, c'est que CONVMESH est correctement installé.

CONVMESH **s'execute en boucle** Exécutez CONVMESH avec l'option -v, cela vous donnera de l'information sur la source de la boucle : lecture du format d'entrée, transformation, écriture du format de sortie

CONVMESH **n'arrive pas à lire un maillage au format msh** La première chose à regarder concerne le numéro du format msh (2.0, 2.1, 2.2, ...). Il est possible que les données écrites dans l'un des blocs ne soit pas écrites dans le bon format. Voir la sous-section 2.1