Quelques fonctionnalités de `Freefem++` pour la création de maillages 3D

Application au filet de pêche

C. Ody - R. Lewandowski
Institut de recherche mathématique de Rennes (IRMaR)

September 15, 2009

Cadre de l'étude : La surpêche (Framework : Overfishing)

www.unep.org
Programme des Nations Unies pour l'environnement
P.O. Box 30552, Nairobi, Kenya
Tel: (254 2) 624105
Fax: (254 2) 624269
E-mail: dewainfo@unep.org
Web: www.unep.org
www.unep.net
PNUE

Bulletin d'Alerte Environnementale

La surpêche, principale menace pesant sur l'écologie maritime mondiale

En 2002, 72% des ressources halieutiques mondiales étaient exploitées plus rapidement qu'elles ne pouvaient se reproduire. La pêche industrielle a divers impacts sur les écosystèmes marins, le plus préoccupant restant la diminution rapide des populations de poisson. Un quart des prises totales (27 millions de tonnes pour 2003) ne sont pas celles visées, et sont le plus souvent perdues.

Lutte contre la surpêche (Limiting overfishing)

1. Pêche raisonnée et durable
2. Politique des quotas
3. Amélioration des techniques de pêche et de la sélectivité des filets

Amélioration de la sélectivité des filets (Improving net selectivity)

1. Campagnes en mer
2. Expériences sur modèles réduits

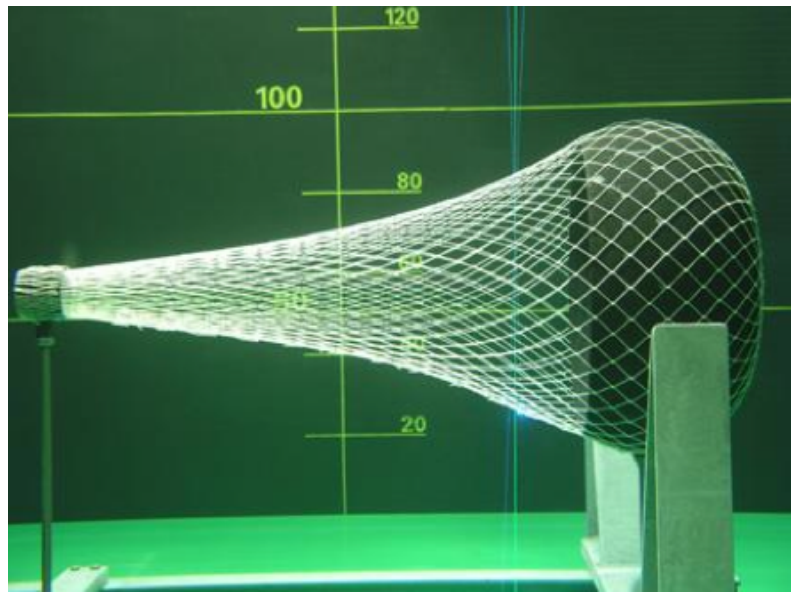


Figure: Maquette IFREMER.

3. Simulations numériques

Cahier des charges de la simulation numérique (Numerical simulation of flow through nets)

Prise en compte de

1. l'écoulement (turbulent) autour et au travers du filet
2. le mouvement et la déformation du filet
3. le déplacement et la prise des poissons

$\longrightarrow$ Interaction complexe entre ces trois aspects

Cahier des charges de la simulation numérique (Numerical simulation of flow through nets)

Prise en compte de

1. l'écoulement (turbulent) autour et au travers du filet
2. le mouvement et la déformation du filet
3. le déplacement et la prise des poissons

$\rightarrow$ Interaction complexe entre ces trois aspects

Première étape : simuler l'écoulement

1. prendre en compte le filet et les poissons
2. proposer une méthode qui permette la prise en compte future de l'interaction écoulement-filet
3. simuler l'écoulement en 3D

Méthode proposée (Thèse de G. Pichot)

Observations expérimentales réalisées à l'IFREMER:

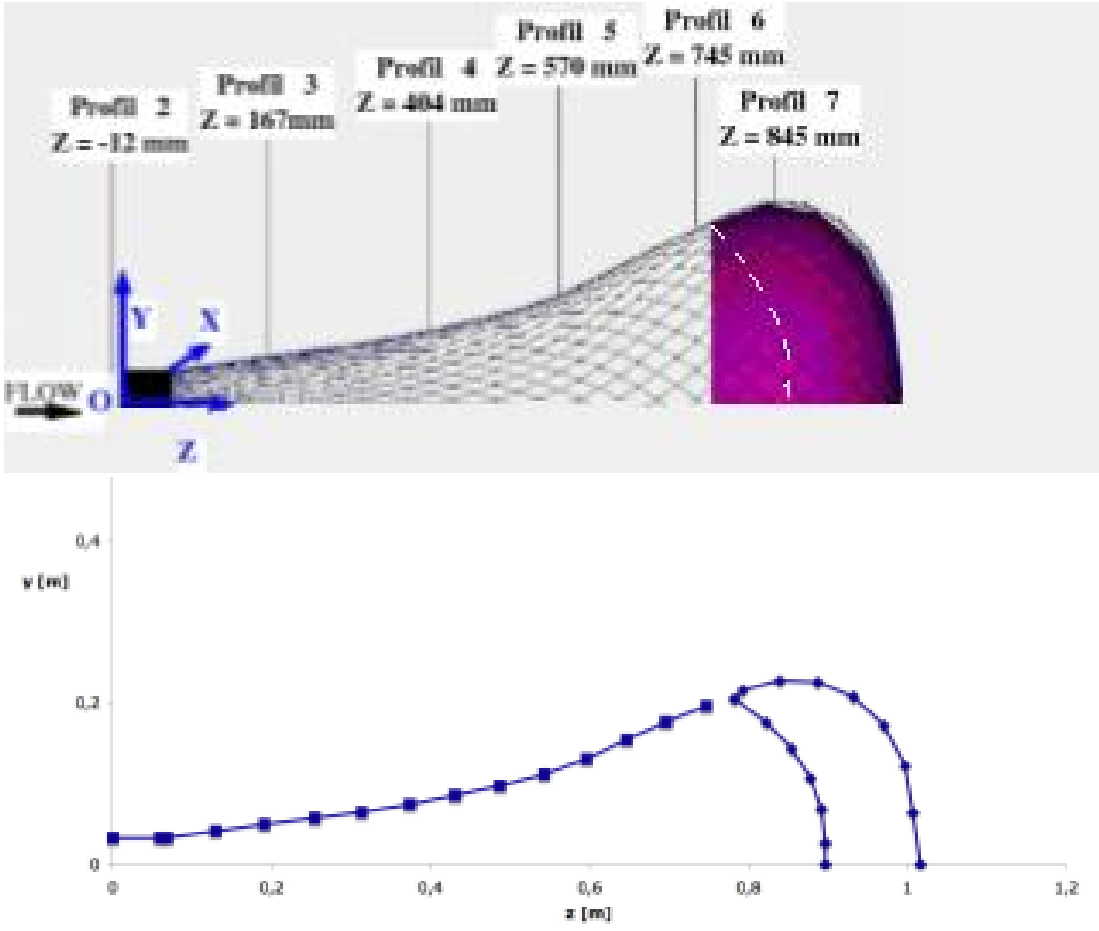


Figure: Filet et profil supérieur correspondant.

Méthode proposée (Thèse de G. Pichot)

Modélisation du filet et des poissons par une membrane poreuse

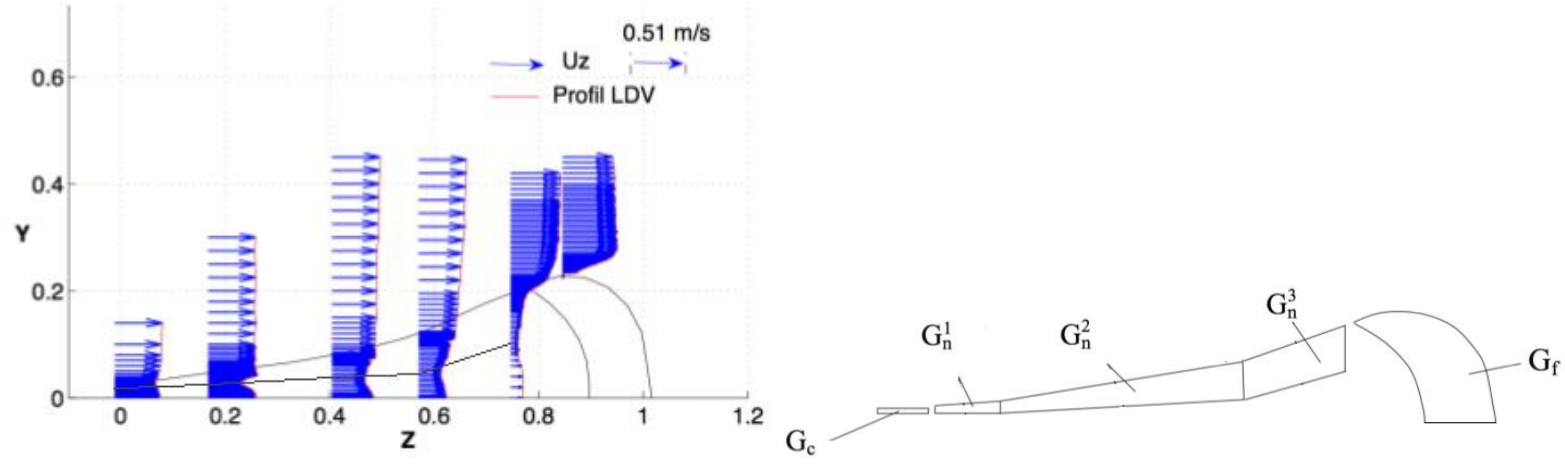


Figure: Le filet vu comme une membrane poreuse composée de 5 sous-domaines

$\longrightarrow$ Terme de pénalisation ajouté dans les équations de Navier-Stokes

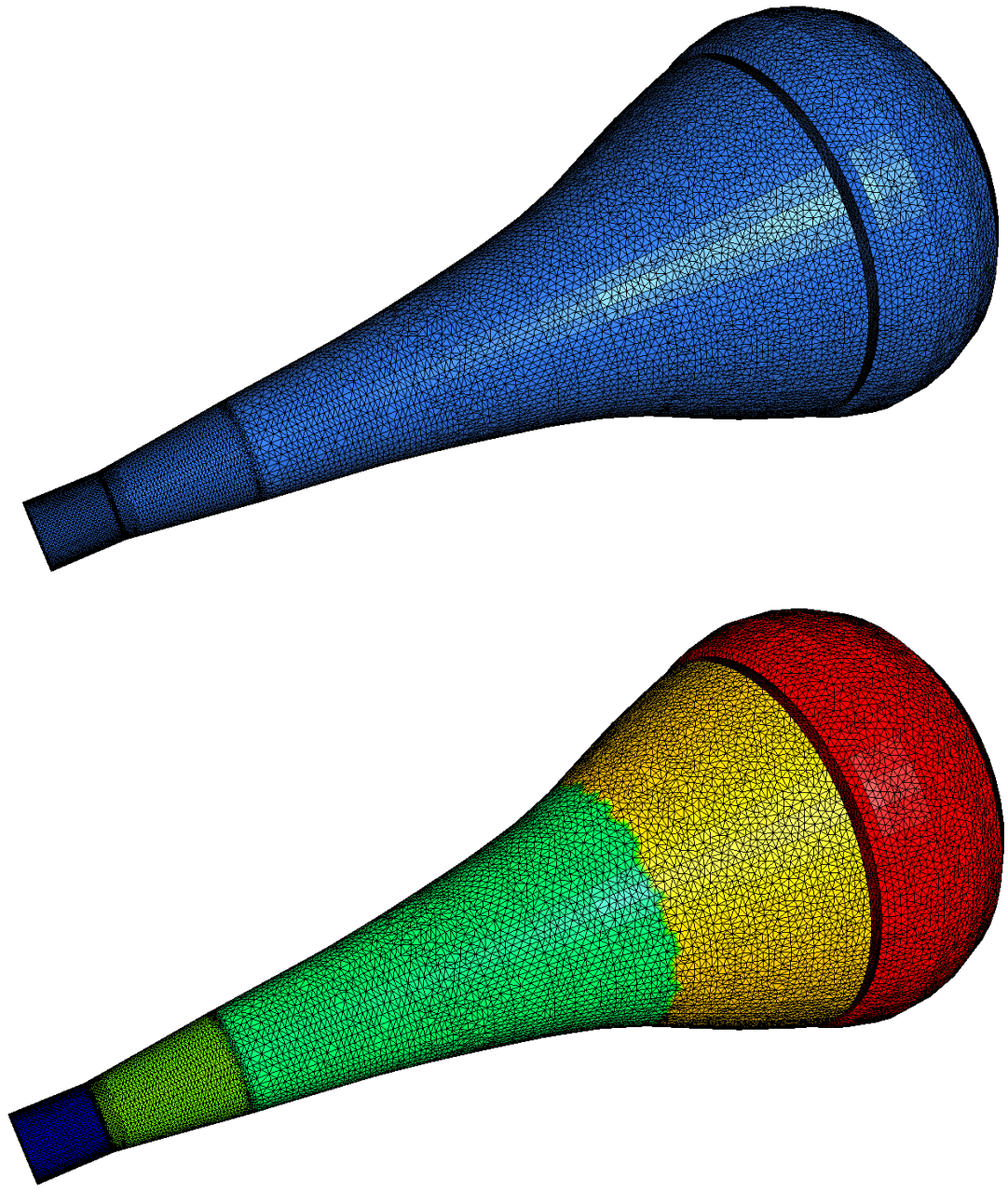
$\frac{\mathbf{u}}{K(\mathbf{x},t)}$ $\longrightarrow$ Introduction d'une perméabilité $K(\mathbf{x}, t)$ qui varie selon le sous-domaine

Début du travail en 3D à l'aide de FreeFEM++

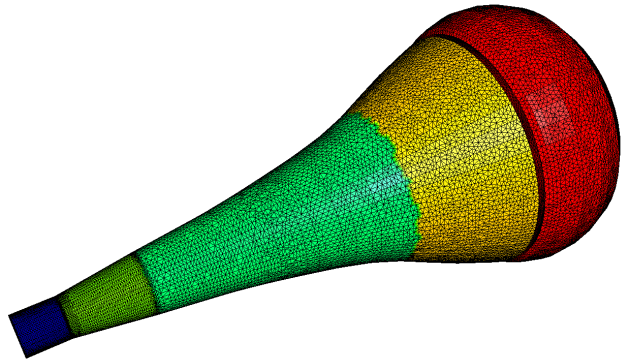
Structure du code:

```
// Main
//=====

// load "msh3", load "tetgen", load "medit"
include "includes_and_parameters.edp";

// Creation du filet 3D
include "net_mesh.edp";

// Creation d'un maillage non conforme
include "simple_3D_mesh.edp";

// Espaces fonctionnels et variables
include "fe_functions.edp";

// Projection de la permeabilite sur le maillage non conforme
include "permeability_projection_on_non_conformal_mesh.edp";

// Resout Navier_Stokes sur maillage non conforme
include "solve_Navier_Stokes.edp";
```

Création d'un filet avec différents sous-domaines



```
// Fonction pour la creation d'un maillage surfacique de revolution a partir d'un profil 2
func mesh3 RevolutionSurface(int numfile, int[int] & pos, int orient, int ntheta)
{
...
}

// Main
int orientation=1;
int ntheta=50;

// Surface de revolution 1
int[int] points1=[0,2,3,5];
mesh3 Surface1=RevolutionSurface(1,points1,orientation,ntheta);

// Surface de revolution 1
int[int] points2=[0,3,4,11];
mesh3 Surface2=RevolutionSurface(2,points2,orientation,ntheta);


....

// Addition des mailles surfaciques pour obtenir avant le maillage volumique
mesh3 FiletSurface=Surface1+Surface2+...+Surface5;
```



On part d'un fichier de points .txt qui correspondent au profil 2D de chaque sous-domaine.

```
7 // --> Nombre de lignes du fichier
596 53.16
770 110.98
770 203.35
747 195.5
696 176
647 154
596 130.5
```

Au début de la fonction, on importe ces points dans deux tableaux, un pour les abscisses, un pour les ordonnées:

```
func mesh3 RevolutionSurface(int numfile, int[int] & pos, int orient, int ntheta)
{
 // Ouverture du fichier "geometry4.txt"
 ifstream file("geometry"+numfile+".txt");

 // Lecture du nombre de lignes
 int nn=0;
 file >> nn;

 // Declaration des tableaux pour les abscisses et les ordonn\'ees
 real[int] Netx(nn),Nety(nn); // abs et ord des noeuds

 // Lecture du fichier
 for (int i=0;i<nn;i++)     file >> Netx[i] >> Nety[i] ;
```

Ensuite, on crée des fonctions (à l'intérieur de la fonction principale) pour réaliser une interpolation linéaire de la position des points situés entre deux points consécutifs du tableau

```
// Interpolation pour X
func real Xnet(real t)
{
 int i=min(int(t),Netx.n-2);
 real t1=t-i, t0=1-t1;
 return Netx[i]*t0 + Netx[i+1]*t1;
}

// Interpolation pour Y
func real Ynet(real t)
{
 int i=min(int(t),Netx.n-2);
 real t1=t-i, t0=1-t1;
 return Nety[i]*t0 + Nety[i+1]*t1;
}
```

On définit donc notamment les `border` à partir de ces fonctions

```
// Creation des frontieres du domaine
border botborder(t=0,points[1])  { x = Xnet(t); y = Ynet(t); }          // Bottom border
border rightborder(t=0,1)  { x = XnetR; y =  YnetR(t); }                // Right border
border topborder(t=points[2],points[3])  { x = Xnet(t); y = Ynet(t); } // Top border
border leftborder(t=0,1)  { x = XnetL; y = YnetL(t); }                  // Left Border

// Trace correspondant
int nh=20; int nv=20;
plot(botborder(nh)+rightborder(nv)+topborder(nh)+leftborder(nv),wait=1);
```

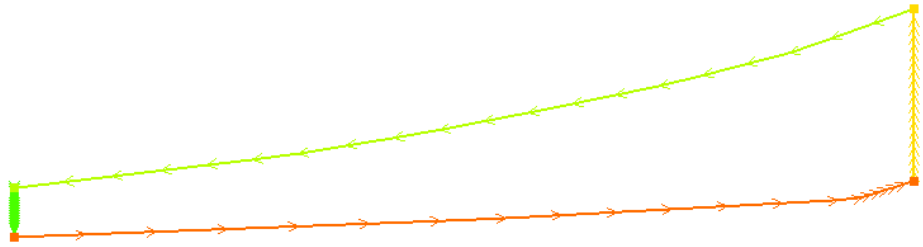


Figure: Tracé des frontières

On maille (notez l'option `fixeborder` pour garder les points du tableau):

```
mesh Th=buildmesh(botborder(nh)+rightborder(nv)+topborder(nh)+leftborder(nv),
                  fixeborder=1);
```

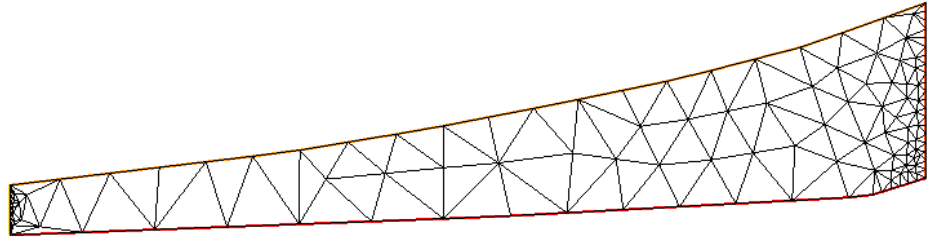


Figure: Maillage 2d

Maintenant, on va créer un maillage surfacique 3D à partir de ce maillage. On fera ensuite une transformation pour obtenir la surface de révolution. Les étapes sont donc les suivantes:

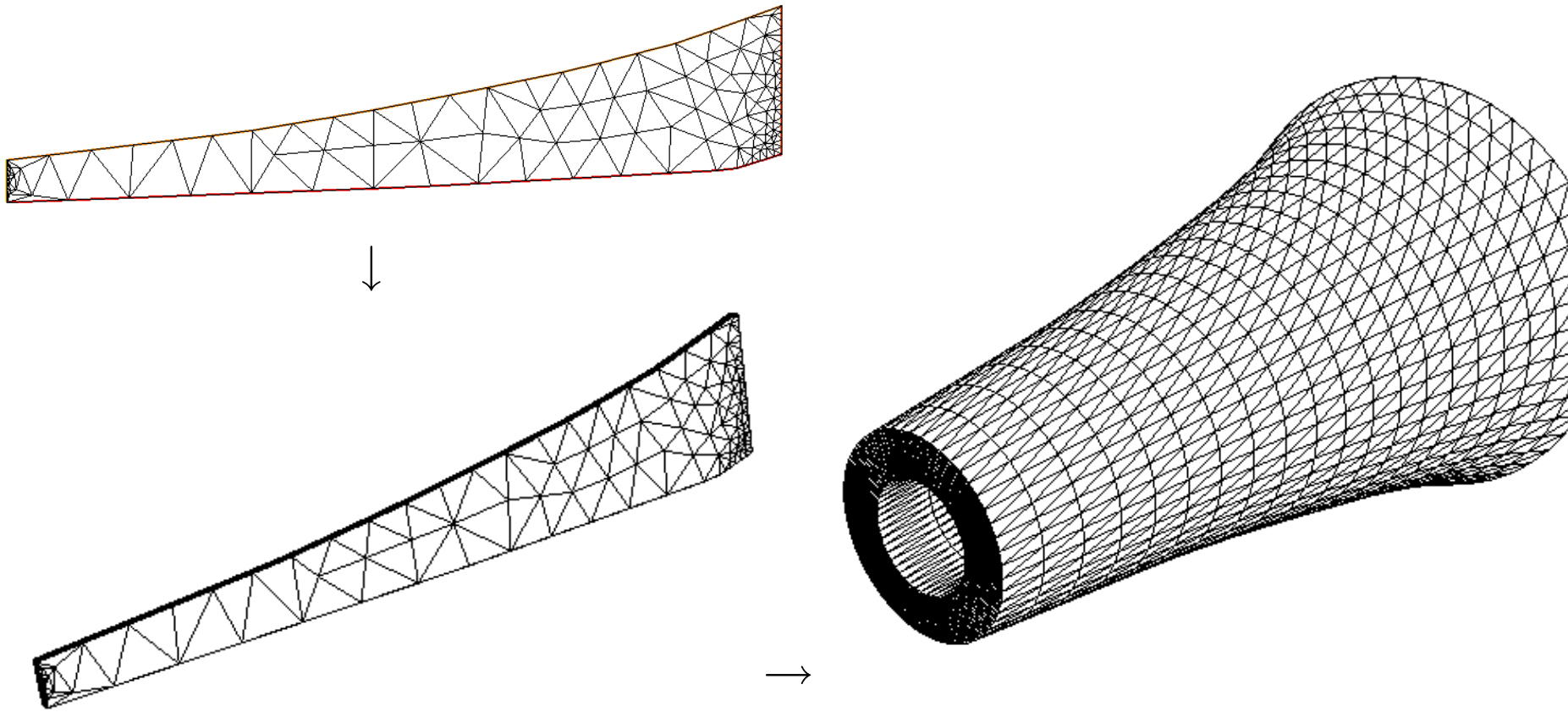


Figure: Du 2d au 3d

On obtient tout d'abord les faces avant et arrière en utilisant la fonction `movemesh23`

```
// Translation of Th to get front (at z=2pi) and rear (at z=0) faces of surface mesh
int[int] rrear=[0,1], rfront=[0,2];
mesh3 Threar = movemesh23(Th,transfo=[x,y,0.],refface=rrear,orientation=-orient);
mesh3 Thfront = movemesh23(Th,transfo=[x,y,2.*pi],refface=rfront,orientation=orient);
```

De façon similaire, on obtiendra les faces latérales, et celles du dessus et du dessous et on les additionne pour obtenir la surface intermédiaire

```
// Addition des surfaces obtenues
mesh3 SurfaceLongueur2Pi=Threar+Thfront+Thtop+Thbot+Thleft+Thright;
```

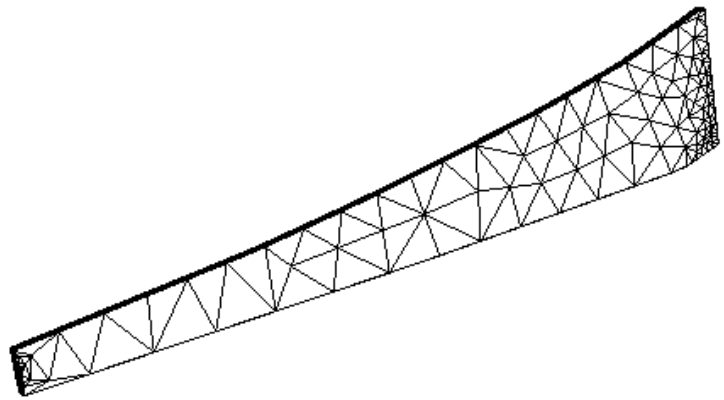


Figure: 3d mesh of length 2π

Pour obtenir finalement la surface de révolution, on utilise la fonction `movemesh3` avec l'option `facemerge` pour fusionner les faces qui vont venir se coller.

```
// Transformation de la surface pour faire la revolution
mesh3 SurfaceFinale=movemesh3(SurfaceLongueur2Pi,transfo=[x,y*cos(z),y*sin(z)],facemerge=
```

```
// Fonction pour la creation d'un maillage surfacique de revolution a partir d'un profil 2
func mesh3 RevolutionSurface(int numfile, int[int] & pos, int orient, int ntheta)
{
...

// Retourne le maillage surfacique 3d
return SurfaceFinale;
}

// Main
int orientation=1;
int ntheta=50;

// Surface de revolution 1
int[int] points1=[0,2,3,5];
mesh3 Surface1=RevolutionSurface(1,points1,orientation,ntheta);

....

// Addition des mailles surfaciques pour obtenir avant le maillage volumique
mesh3 FiletSurface=Surface1+Surface2+...+Surface5;
```

Maintenant, il reste à mailler la surface obtenue. On utilise `tetgen` selon

```
real[int] pointsinterieurs = [65,0,30,17,100,190,0,38,19,100,590,0.,54.,18,100,597.,0.,54.
mesh3 FiletVolume= tetg(FiletSurface,switch="paAQ",nbofregions=5,regionlist=pointsinterieu
```

On peut définir la perméabilité sur le filet

```
fespace Vh_conforme(FiletVolume,P0);
Vh_conforme k;
k_conforme=0*(region==17)+10*(region==18)+0*(region==19)+20*(region==20)+25*(region==21);
```

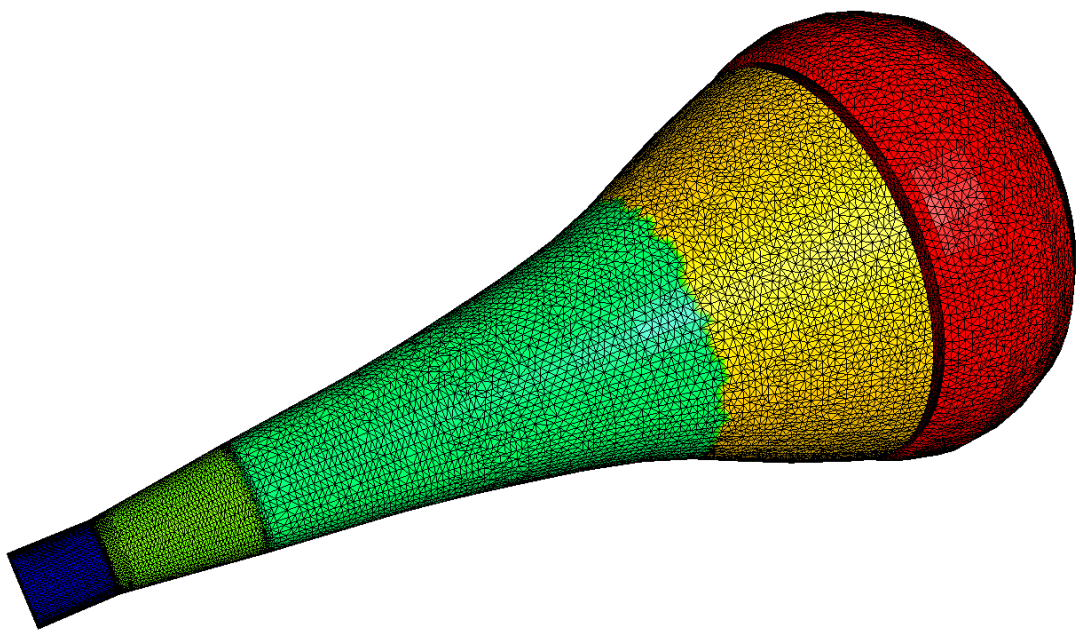
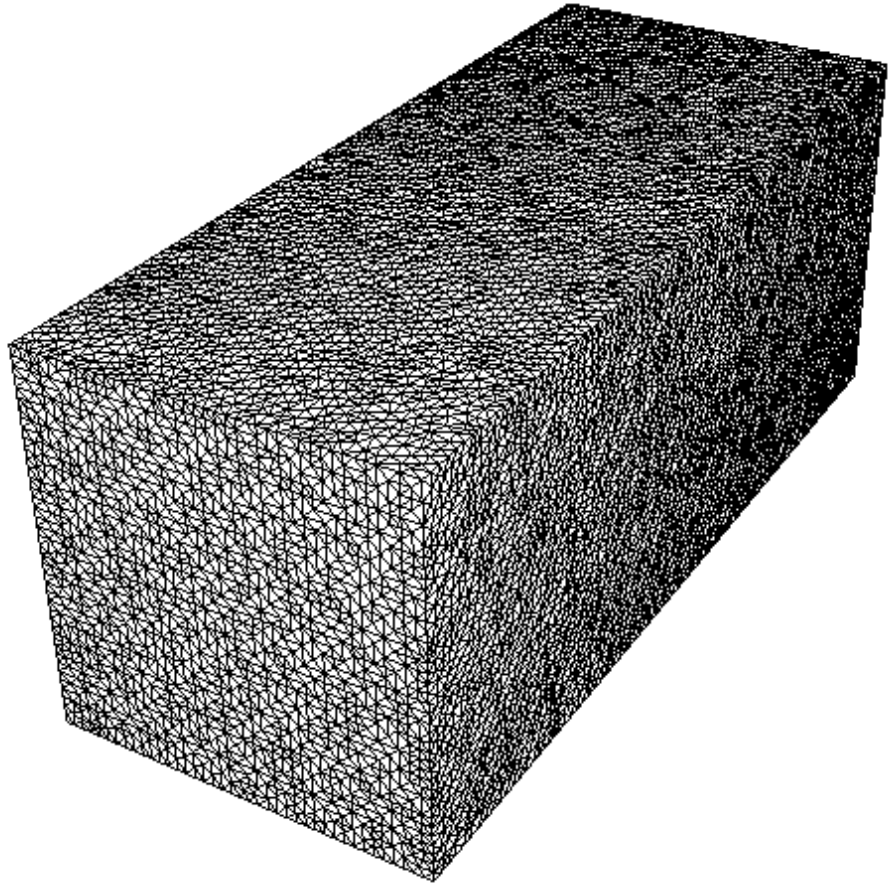


Figure: Le filet 3d composé de 5 sous-domaines

On va projeter sur un maillage non conforme. On crée ce maillage en prenant la fonction `SurfaceHex.edp` et `tetgen` de `mesh_surface.idp`. On pourrait également utiliser la fonction `buildlayer`.

```
mesh3 PaveSurfacique = SurfaceHex(N,B,L,1);
mesh3 PaveVolumique = tetg(PaveSurfacique,switch="pqaAQ",nbofregions=1,regionlist=doma
```



```
fespace Vh_non_conforme(PaveVolumique,P0);
Vh_non_conforme k_non_conforme;
k_non_conforme=k_conforme; // Interpolation/Projection between the 2 meshes
```