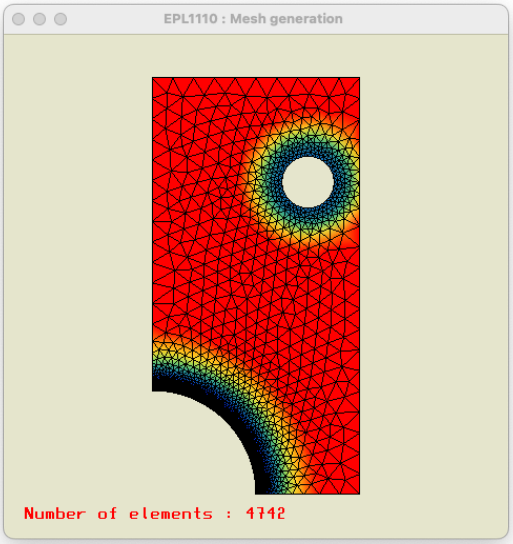
Finite elements for dummies : Créer un maillage !



L'objectif de ce devoir est d'utiliser le Software Development Kit de gmsh pour générer un maillage. Il s'agit bien de vous permettre de créer votre propre géométrie pour le projet.

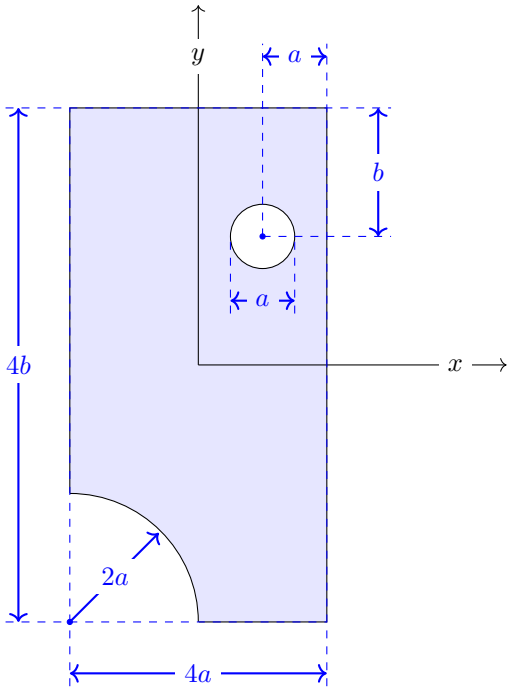
Tout d'abord, nous allons utiliser les concepts de Constructive Solide Geometry pour définir justement la géométrie du notre problème. L'implémentation informatique de cette construction géométrique est basée sur les outils open-source OpenCascade disponibles dans l'API de `gmsh`. On obtient la forme géométrique en retirant deux cercles d'un rectangle. So easy !

Ensuite, nous définissons une carte de taille pour raffiner le maillage le long des deux cercles. Il s'agit de définir $h(x, y)$ la taille souhaitée de l'élément qui englobe le point (x, y). A nouveau, nous allons utiliser l'API de `gmsh` pour générer une triangulation de Delaunay respectant au mieux cette carte de taille.

Définir la géométrie avec OpenCascade !

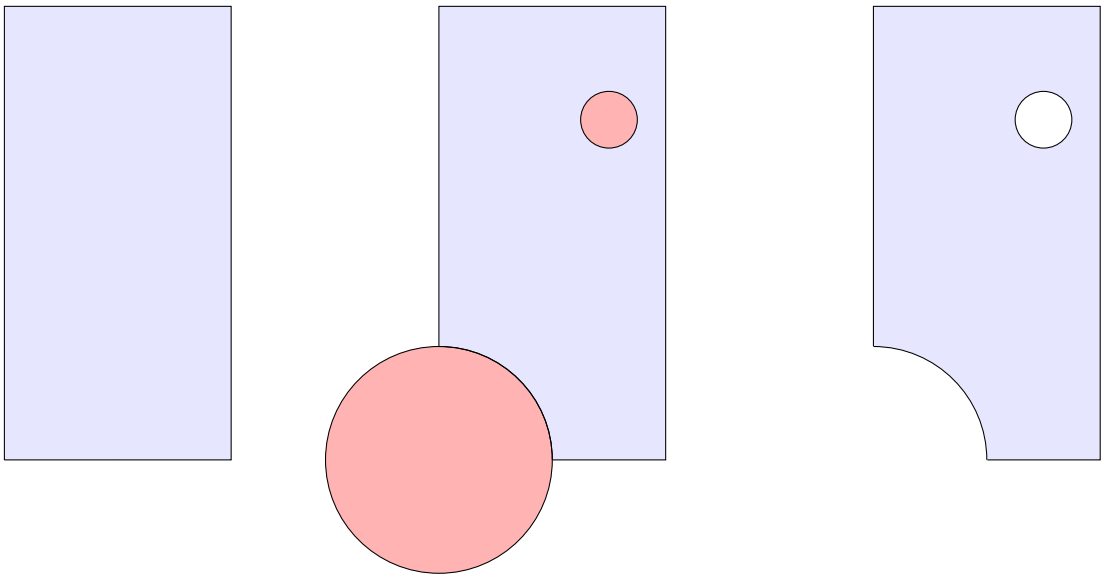
Considérons une plaque percée d'un trou (*hole*) et avec une encoche (*notch*) dans le coin inférieur gauche. Les deux axes sont placés au centre géométrique du rectangle.

```
double Lx = 1.0;
double Ly = 2.0;
theGeometry->LxPlate =  Lx;
theGeometry->LyPlate =  Ly;
theGeometry->xPlate  =  0.0;
theGeometry->yPlate  =  0.0;
theGeometry->xHole   =  Lx / 4.0;
theGeometry->yHole   =  Ly / 4.0;
theGeometry->rHole   =  Lx / 8.0;
theGeometry->xNotch  = -Lx / 2.0;
theGeometry->yNotch  = -Ly / 2.0;
theGeometry->rNotch  =  Lx / 2.0;
```



Pour obtenir la géométrie, on effectue simplement les opérations suivantes :

- Créer le rectangle avec la fonction `gmshModelOccAddRectangle`.
- Créer les deux disques avec la fonction `gmshModelOccAddDisk`.
- Retirer chaque disque du rectangle avec la fonction `gmshModelOccCut`.



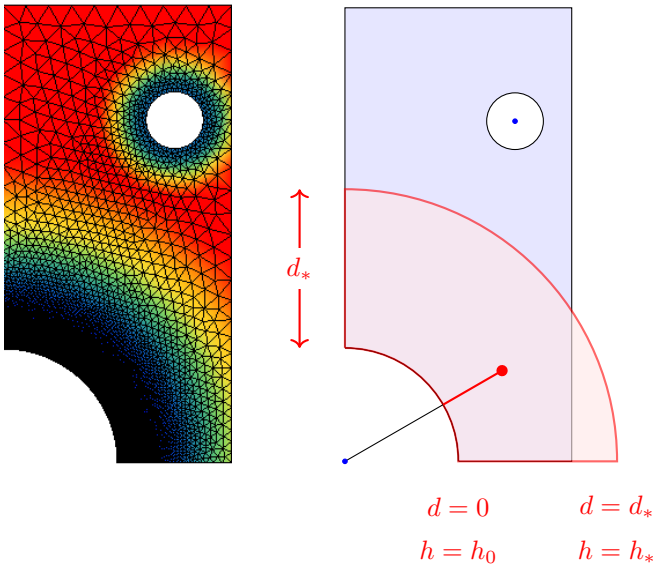
L'unique et la seule difficulté est de trouver les bons arguments à donner aux fonctions : c'est ici que le petit jeu de piste commence ! A vous de trouver les bons arguments en cherchant l'information dans le tutoriel de gmsh... Les assistants ont trouvé l'information assez rapidement et donc vous devriez être encore plus rapides qu'eux :-)

Générer une triangulation de Delaunay avec gmsh !

La seconde étape consiste à générer un maillage de triangles avec `gmesh`. Votre contribution ici se limite à écrire une toute petite fonction `geoSize` qui donnera la taille requise $h(x, y)$ en tout point du plan. Dans cette optique, on définit une taille de référence globale h_*.

Sur les deux cercles, on va prescrire une taille de référence différente. A titre d'exemple, on souhaite imposer une valeur h_0 avec une distance de transition d_* pour le cercle de l'encoche. Pour les points dont la distance au cercle $d(x, y)$ appartient à l'intervalle $[0, d_*]$, la taille de référence locale sera obtenue à partir d'une interpolation polynomiale du troisième degré en imposant les valeurs ainsi qu'une dérivée nulle aux deux extrémités. On obtiendra la valeur de h à partir d'un polynôme de degré trois $f(d)$ tel que :

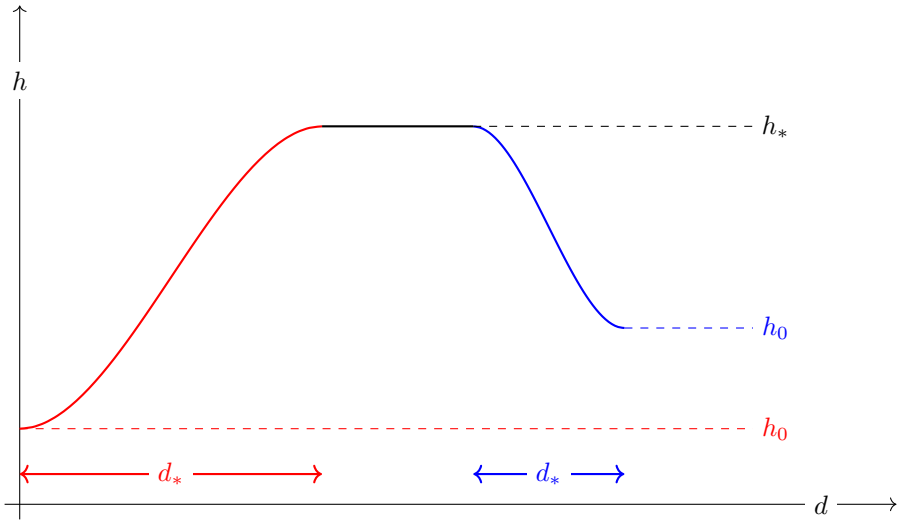
$$\begin{aligned} f(0) &= h_0 & \qquad f'(0) &= 0 \\ f(d_*) &= h_* & \qquad f'(d_*) &= 0 \end{aligned}$$



Les données pour construire la carte de taille se résument en cinq paramètres.

```
double h = theGeometry->h;
double h0 = theGeometry->hNotch;
double d0 = theGeometry->dNotch;
double h1 = theGeometry->hHole;
double d1 = theGeometry->dHole;
```

Il vous suffit donc d'écrire une interpolation d'Hermite unidimensionnelle et ensuite de superposer les deux fonctions de mélanges et de prendre la valeur minimale requise pour $h(x, y)$. A titre d'exemple, considérons l'évolution de la taille de référence sur la droite reliant les centres des deux cercles[1]. La taille de référence locale en fonction de la distance au premier cercle peut être esquissée sous la forme suivante.



[1] Attention, les deux zones à raffiner peuvent s'intersecter...
Est-ce que ce cas est correctement géré dans votre implémentation ?

Plus concrètement, il faut implémenter les fonctions suivantes :

1. A titre d'échauffement, il vous est simplement demandé de remplir les trous dans le code de la fonction suivante :

   ```
   void geoMeshGenerate();
   ```

   Cette fonction construit la géométrie avec `OpenCascade`.
   Ensuite, on fait appel à la libraire de `gmsh` pour générer le maillage de triangles.
   C'est juste un petit jeu de piste dans la documentation de l'API de `gmsh`.
   C'est vraiment élémentaire !

2. Et maintenant, un tout petit peu plus compliqué, mais vraiment un fifrelin plus compliqué :-)

   ```
   double geoSize(double x, double y);
   ```

   Il s'agit de calculer la taille requise en chaque point du domaine.
   Toutes les données sont stockées dans la structure `theGeometry` !

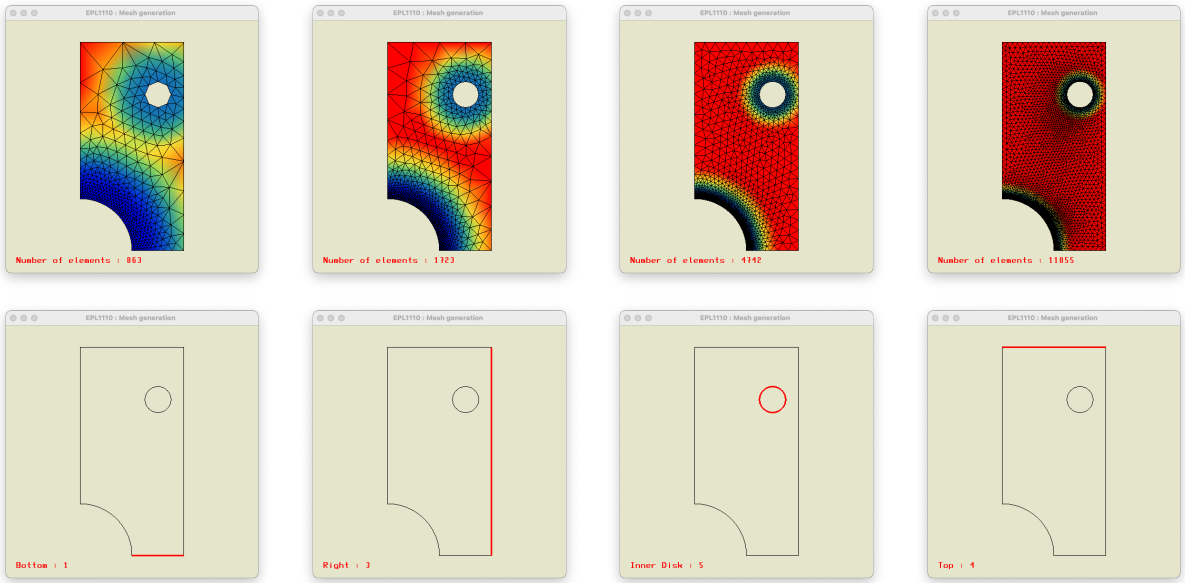
   ```
   double h  = theGeometry->h;
   double x0 = theGeometry->xNotch;
   double y0 = theGeometry->yNotch;
   double r0 = theGeometry->rNotch;
   double h0 = theGeometry->hNotch;
   double d0 = theGeometry->dNotch;


   double x1 = theGeometry->xHole;
   double y1 = theGeometry->yHole;
   double r1 = theGeometry->rHole;
   double h1 = theGeometry->hHole;
   double d1 = theGeometry->dHole;
   ```

3. Attention : le code du devoir a été épuré et l'interface a été légèrement modifié : si vous avez réalisé votre devoir avec la version provisoire, vous devriez être capable de récupérer la majorité de votre code et d'obtenir le résultat requis très rapidement : si, si !
   Ce devoir est vraiment simple et pas très compliqué !
   C'est vraiment impardonnable de ne pas y arriver, les gaminets !

4. Vos deux fonctions seront incluses dans un unique fichier `homework.c`, sans y adjoindre le programme de test fourni ! Ce fichier devra être soumis via le web et la correction sera effectuée automatiquement. Il est donc indispensable de respecter strictement la signature des fonctions. Votre code devra être strictement conforme au langage `C` et il est fortement conseillé de bien vérifier que la compilation s'exécute correctement sur le serveur.

Et tout cela, pourquoi ?



A l'issue de l'implémentation de votre devoir, il est possible d'utiliser votre programme comme un tout petit laboratoire numérique pour générer des petits maillages :

1. Qu'a-t-on obtenu à la fin du processus ?
   Un maillage de triangles !
   Mais également un maillage contenant tous les segments de la frontière du maillage.
   Les noeuds, les éléments triangulaires et les éléments frontières sont stockés à la fin de l'exécution du code : ils peuvent être également enregistrés dans un fichier en faisant appel à `geoMeshWrite`. Ce fichier servira de fichier de données pour le projet de cette année.

2. On obtient également des entités qui permettent de définir des conditions aux frontières !
   Dans l'exemple fourni, on a six morceaux de la frontière : les arcs de cercles et les quatres cotés du rectangle... Ces entités sont également récupérées et stockées sous la forme de liste d'éléments unidimensionnels... En d'autres mots, ce sont des sous-maillages ou des domaines qui sont définis par une liste d'éléments du maillage de la frontière.

3. A la fin de la génération du maillage, on obtient donc une série d'entités mais a priori, on ignore à quelle partie topologique correspond une entité qui n'a qu'un simple numéro (*tag* dans le jargon de `gmsh`) pour l'identifier. Il est donc nécessaire de pouvoir visualiser ces entités dans l'interface graphique. C'est pourquoi, l'équipe enseignante vous permet d'avoir la visualisation des domaines en tapant la lettre `D`. Au moyen de la lettre `N`, on fait défiler les diverses entités et on peut identifier celle qui nous intéresse. Finalement, on attribue un nom à une entité avec la fonction `geoSetDomainName`, comme nous l'avons fait dans le programme principal.

4. Il est aussi possible de générer des maillages de quadrilatères : cela peut être une bonne idée dans certains cas, mais vous allez observer que souvent notre mailleur favori n'arrive pas à mailler une géométrie rien qu'avec des quadrilatères... quelques triangles subsistent : ce qui est bien dommage. Adapter le code pour les quadrilatères est laissé aux bons soins des étudiants les plus astucieux...

5. C'est vraiment très très compliqué de générer des maillages de quadrilatères, le petit Jean-François avait obtenu un financement prestigieux pour y arriver, mais c'est pas encore très facile de les obtenir

sur la version actuelle du petit code de mon super cotitulaire.... Si vous souhaitez en apprendre davantage sur ce sujet, il y a un cours de géométrie numérique sur le cours sujet.

Vincent :-)