Utilisation d'un mailleur

La méthode des éléments finis nécessite l'utilisation d'un maillage du domaine de calcul considéré. Les mailleurs (outils logiciels permettant de créer des maillages) sont des programmes complexes. À titre d'exemple, on peut citer les deux mailleurs GMSH et Triangle qui sont gratuits et permettent de mailler des domaines en dimension 2 (ou 3 pour GMSH).
Dans ce TP nous utiliserons Triangle et vous serez libres de travailler avec Matlab ou avec Python. Pour Matlab, nous fournissons une interface et pour Python il faut importer la librairie correspondante (`import triangle`). Le but du travail ci-dessous est de prendre en main ce logiciel depuis Matlab ou Python.

Les principales caractéristiques de Triangle sont :

- Il génère des maillages 2D, composés de triangles, de type Delaunay.
- Il travaille à partir d'une description de la géométrie du domaine (description des frontières et des trous éventuels).
- L'utilisateur peut spécifier une taille maximale des triangles (d'où un contrôle sur le nombre de triangles).
- L'utilisateur peut spécifier une densité non homogène de triangles (raffinements locaux du maillage).

Ce mailleur est disponible gratuitement sur internet à l'adresse (http ://www-2.cs.cmu.edu/quake/triangle.html).

On fournit dans le cadre des travaux pratiques une interface permettant de l'utiliser depuis Matlab. Cette interface apporte deux commandes : `triangle`, qui maille un domaine géométrique et `refine`, qui raffine (ou déraffine) un maillage existant. Pour Python il existe une librairie `triangle` dont l'utilisation est similaire. La commande `triangle.triangulate` permet de mailler un domaine géométrique et l'option `'r'` raffine un maillage existant.

1 Définition de la géométrie

Le mailleur Triangle travaille à partir de la définition de la géométrie (frontière extérieure et bord des trous éventuels). La géométrie sera définie à l'aide d'une *structure* Matlab ou d'un *dictionnaire* Python.

C'est une variable qui regroupe des composantes de type différent (scalaires, tableaux de tailles différentes, chaînes de caractères, autres structures/dictionnaires, etc). Par exemple, pour construire une structure `S` (en Matlab) ou un dictionnaire `D` (en Python) contenant une chaîne de caractères nom et un tableau valeur, on pourra écrire :

```
S.nom = 'un nom';
S.valeur = [ 1 2 3];
```

```
D = {'nom': 'un nom',
     'valeur': np.array([1, 2, 3])
}
```

Exemple d'utilisation de `S` et `D` :

```
>> S
S =
nom: 'un nom'
valeur: [1 2 3]

>> S.valeur(1) + S.valeur(2)
ans =
3
```

```
>>> D
{'nom': 'un nom', 'valeur': array([1, 2, 3])}


>>> D['valeur'][0]+D['valeur'][1]
3
```

La géométrie sera décrite par une structure ou un dictionnaire contenant deux tableaux :

- Un tableau de points.
- Un tableau de segments représentant le bord du domaine à mailler et reliant les points précédents.

Exemple : On considère le domaine $\Omega = [0, 3] \times [0, 1.5]$. Sa frontière est décrite par les 4 points $p1, p2, p3, p4$ et les 4 segments qui les relient. On construit donc la structure `g` ou le dictionnaire `A` contenant les deux tableaux :

- De points

$$\texttt{g.points} = \texttt{A}[\texttt{'vertices'}] = \begin{pmatrix} x1 & y1 \\ x2 & y2 \\ x3 & y3 \\ x4 & y4 \end{pmatrix} = \begin{pmatrix} 0.0 & 0.0 \\ 3.0 & 0.0 \\ 3.0 & 1.5 \\ 0.0 & 1.5 \end{pmatrix}.$$

- De segments. Ceux-ci contiennent les numéros des points de début et de fin de chaque segment de la frontière :

$$\texttt{g.segments} = \texttt{A}[\texttt{'segments'}] = \begin{pmatrix} p1 & p2 \\ p2 & p3 \\ p3 & p4 \\ p4 & p1 \end{pmatrix} = \begin{pmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 4 \\ 4 & 1 \end{pmatrix}.$$

2 Premier maillage

Question 1. Définir la structure (resp. le dictionnaire) ci-dessus dans l'espace de travail Matlab (resp. Python). Générer un premier maillage à partir de cette géométrie, en tapant sur la ligne de commande (si `g` est le nom de la structure ou `A` le nom du dictionnaire ci-dessus) :

```
>>m = triangle(g);
```

```
>>>m = triangle.triangulate(A)
```

La fonction triangle (resp. triangle.triangulate) retourne une structure (resp. un dictionnaire) en résultat qui contient, entre autres, les composantes suivantes :

- `vertices` : tableau des coordonnées des nœuds du maillage,
- `triangles` : pour chaque triangle du maillage, les numéros de ses trois sommets.

Afficher et interpréter le contenu des tableaux produits en résultat. Combien de triangles y a-t-il dans le maillage ? Combien de sommets ?

3 Raffinement homogène : second maillage

Pour obtenir des maillages plus réalistes, Triangle permet de définir une surface maximale pour chaque triangle :

```
m2 = triangle(g, aireMax);  |  m2= triangle.triangulate(A,'Da'+str(aireMax))
```

où `aireMax` est un nombre décimal donnant l'aire maximale des triangles et `D` impose que le maillage obtenu soit une triangulation de Delaunay.

Question 2. Générer un maillage contenant à peu près 50 triangles (fixer un ordre de grandeur de l'aire maximale à imposer et tester plusieurs valeurs).

4 Affichage du maillage

Il serait très pratique de pouvoir afficher les maillages obtenus.

Question 3. Écrire une fonction `plotMesh` qui affiche un maillage.
Syntaxe : `plotMesh(mesh,options)`

On pourra utiliser la commande `trimesh` de Matlab ou `triplot` de Python (inclus dans la librairie `matplotlib.pyplot`). `options` est un paramètre optionnel qui doit permettre l'affichage optionnel de données supplémentaires comme suit :

- Un appel
  `»plotMesh(mesh)`
  affiche simplement le maillage.
- Un appel
  `»plotMesh(mesh,'v')`
  affiche le maillage avec les numéros des sommets à côté de chaque sommet.
- Un appel
  `»plotMesh(mesh,'t')`
  affiche le maillage avec les numéros des triangles au milieu de chacun d'eux.
- Un appel
  `»plotMesh(mesh,'vt')`
  affiche le maillage avec les numéros des sommets et des triangles.

Dans l'exemple précédent, la variable `option` sera vide si l'utilisateur ne fournit pas d'option. (On pourra également utiliser les commandes `nargin, varargin` de Matlab ou `len, *args` de Python pour connaître le nombre et les arguments fournis par l'utilisateur). Rechercher des caractères dans une chaîne d'option peut se faire grâce à la commande `findstr` de Matlab ou `.find` de Python. Enfin, on recalera les axes de la figure de façon à ce que les échelles soient les mêmes grâce à la commande `axis equal;` de Matlab ou `plt.gca().set_aspect('equal')` de Python.

5 Raffinement non homogène

Triangle permet de générer des maillages raffinés localement (on dit aussi avec raffinement local). Ce type de maillage est utilisé souvent de la façon suivante :

- On résout numériquement le problème en utilisant un maillage homogène.
- La précision de la solution est évaluée dans les différentes parties du maillage (on obtient un nombre positif sur les différents triangles du maillage). L'ensemble des valeurs positives sur les triangles forme une métrique sur le maillage.
- Un nouveau maillage est généré dont le degré de raffinement est fonction de la métrique ci-dessus.
- Le processus peut être répété et poursuivi jusqu'à convergence. Le critère de convergence est en général du type "erreur globale minimale" ou "erreur locale de même valeur sur tous les éléments".

Question 4. En repartant de la géométrie précédente, générer un premier maillage avec un raffinement uniforme défini par une aire maximale de 0.2. Visualiser le maillage généré.

Soit n, le nombre de triangles du maillage ci-dessus. Créer un vecteur colonne `area` de taille n, composé des valeurs :

$$\texttt{area}(k) = 0.2\ (x_{\text{Centre de gravité du triangle k}})^2, \quad \forall k = 1, \cdots, n.$$

Les composantes de `area` donnent pour chaque triangle, l'aire locale maximale à respecter dans le triangle correspondant. Cet exemple permet de spécifier un raffinement de maillage plus important pour x petit.
Générer un nouveau maillage à partir du précédent et du vecteur `area` par la commande :

```
m3 = refine(m2, area);
```

```
m2['triangle_max_area']=area
m3=triangle.triangulate(m2,'Dra')
```

Visualiser le maillage produit avec `plotMesh`.

6 Contraintes internes au maillage

Quelquefois le maillage doit représenter une structure. Par exemple, on peut vouloir calculer l'évolution d'une bulle d'air dans un fluide, ou bien un solide composé de deux matériaux différents. Enfin, le domaine peut tout simplement comporter des trous. Dans ces cas, il est nécessaire que le maillage reflète cette décomposition spatiale en plusieurs sous domaines.

Dans ce cas les tableaux `p` et `s` contiendront l'ensemble des points et segments sur le bord extérieur du domaine ainsi que sur l'ensemble des courbes intérieures que doit respecter le maillage.

Question 5. Écrire la fonction qui construit la géométrie composée du carré $[-1, 1]^2$ à l'intérieur duquel il y a un cercle de rayon $1/2$ et de centre O. Le nombre de segments intérieurs pris sur le cercle sera passé en arguments.
Syntaxe : `geomDisc(n)`

Ensuite, générer le maillage en calculant la contrainte sur la taille des triangles (raffinement homogène) de façon à obtenir un "beau" maillage. Vérifier que la maillage suit le frontière intérieure (on pourra tracer cette frontière en rouge).

7 Labels

Dans le cas précédent, il est indispensable de savoir dans quel sous domaine sont les triangles générés. Pour ce faire, on spécifie des *labels* de domaine dans la géométrie. Si `g` est cette géométrie, et nR est le nombre de régions que l'on souhaite spécifier, on doit donner en plus

- Un tableau `g.regions` de taille $nR \times 2$ contenant, pour chaque région désirée, les coordonnées d'un point interne à la région.
- Un tableau `g.lab_regions` de taille $nR \times 1$ (un vecteur colonne) contenant pour chaque région, un label (un nombre).

A la construction du maillage, le mailleur associera automatiquement à chaque triangle le label correspondant à sa région. Cette information est contenue dans le tableau `lab_triangles` du maillage qui est construit (si `m` est le maillage, le tableau en question est `m.lab_triangles`).
Cependant, cette fonctionnalité est implémentée dans l'interface Matlab, mais elle ne l'est pas dans la librairie Python.

De la même façon, le tableau `lab_vertices` (resp. `vertex_markers`) contiendra des labels pour les sommets du maillages, si l'on a pris soin de remplir les tableaux correspondant de la géométrie `g.lab_points` (resp. `A['vertex_markers']`) et `g.lab_segments` (resp. `A['segment_markers']`) pour chaque point et segment fourni dans la géométrie.

Question 6. Modifier le programme `plotMesh` de façon à faire apparaître les labels des triangles et/ou des sommets, suivants que la chaîne d'options contient les lettres 'T' ou 'V' respectivement. Visualiser les labels de triangles générés dans le cas du maillage précédent.

Question 7 (Bonus). Il peut être intéressant de dessiner le maillage non pas en "fil de fer" mais plutôt de colorier les triangles en fonction des labels correspondants. On pourra pour cela modifier la fonction plotMesh de sorte que si la chaîne d'option contient le caractère 'C', on utilise non pas la fonction `trimesh` mais plutôt `trisurf`. Attention, la fonction traçant des figures tridimensionnelles, il faudra lui fournir un argument d'élévation (un tableau de 0) et un tableau de couleurs (les labels). Il faudra ensuite veiller à changer la vue de la figure pour la voir "du dessus" grâce à la fonction `view`.

8 Domaines à trou

On peut aussi définir un domaine contenant des trous. Par exemple, dans le cas précédent du disque à l'intérieur du carré, le disque central peut être évidé. Pour cela, il suffit d'ajouter une composante `holes` dans la structure (resp. le dictionnaire) géométrie qui est passée en argument de `triangle` (resp. de `triangle.triangulate`). Si `g` est une structure (resp. `A` est un dictionnaire) géométrie, rajouter une ligne du type :
`g.holes = A['holes'] = [ x1, y1; ...; xn, yn ]`
pour créer `n` trous dans la géométrie.

Ici, `(xi, yi)` est la position d'un point dans le $i-$ème trou.

Question 8. Après avoir défini le trou dans le domaine précédent, exécuter la commande

```
m4 = triangle(g, aireMax); | m4 = triangle.triangulate(A, 'Dpa'+str(aireMax))
```

où l'on choisira le paramètre `aireMax` en fonction du nombre de segments que l'on a pris sur le cercle intérieur.

Tracer le maillage obtenu. En Python on pourra également s'intéresser à la signification de l'option `'p'` ajoutée ici et observer le maillage lorsque cette option n'est pas prise en compte.

9 Arêtes

En Matlab, le mailleur fournit également un tableau (`edge`) de toutes les arêtes présentes dans le maillage sous la forme d'un tableau des numéros des deux sommets qu'elles joignent. Les arêtes ont également un tableau de labels (`m.lab_edges`).
En python, on fournit une fonction `meshEdges(m)` qui, pour un maillage `m` donné, renvoie, entre autres, un tableau `edge` similaire à celui fournit par l'interface Matlab et un tableau de labels `edge_markers`.

Modifier la fonction `plotMesh` de façon à écrire les numéros des arêtes sur chaque arête (option 'e') ou leur label (option 'E'). Remarquer les labels des arêtes internes ou ceux d'arêtes du bord du domaine ou d'un sous-domaine.

10 Divers

Les programmes écrits en Matlab ou Python ont la particularité de pouvoir contenir une aide utilisateur. Cette aide figure dans le premier commentaire (juste après la définition de la fonction). On consultera avec intérêt le programme `triangle.m` et ce qui se passe lorsque l'on tape `help triangle` (resp. `help(triangle.triangulate)`).
Il pourra être très utile par la suite d'écrire une aide pour la commande `plotMesh` (et les autres) que vous avez écrite.
Essayer de mailler plusieurs domaines (en mélangeant les trous, régions, labels de bords, ...).