Notes du cours d'Équations aux Dérivées Partielles de l'ISIMA, deuxième année
http://www.isima.fr/leborgne

Introduction à Matlab PDE Toolbox

Gilles Leborgne

20 février 2006

Table des matières

1 Introduction

Pour une introduction générale a Matlab, voir le cours de Pierre Chainais.

Et les informations générales principales de ce poly viennent de la doc Matlab : *Partial Differential Equation Toolbox, For Use with Matlab*, The MathWorks, Inc., August 1995. Cette doc est fournie en format PDF : allez dans `Help Desk (HTML)`, et cliquez sur `Online Manuals (in PDF)`, choisir `Partial Differential Equations (PDE) Toolbox User's Guide`.

Pour l'aide html des fonctions du Toolbox PDE, allez dans `Help Desk (HTML)`, cliquez sur `Partial Differential Equations (PDE) Toolbox Ref.`, et cliquez sur le nom de la fonction souhaitée.

Voir également le site http ://www.mathworks.com.

Sept programmes de démonstration sont proposés. Pour les lancer, la fenêtre Matlab étant ouverte :

» `pdedemo1`

pour le premier...

2 Démarche pour résoudre un problème avec le GUI

Dans ce paragraphe, on ne présente que les bases. En particulier, on ne pourra traiter que des géométries simples, et pour des EDP linéaires répertoriées. Pour des géométries plus spécifiques (autres que faites de morceaux d'ellipses ou de segments de droite), on renvoie au paragraphe suivant.

Ouvrez matlab, puis au prompt » de matlab, tapez `pdetool` :

» `pdetool`

Cela ouvre un GUI (Graphical User Interface), celui donnant l'environnement graphique permettant de résoudre des problèmes d'EDP dans des géométries simples.

2.1 Géométrie : préprocesseur

Il s'agit de définir la géométrie, i.e. le domaine Ω, dans lequel on cherche une solution. C'est la partie préprocesseur, qui peut être faite à l'aide de nombreux logiciels (de CAO ou de "dessins") dont on peut extraire un certains nombre d'informations : coordonnées de points du maillage, référencement des triangles dessinés en fonction de ces points...

2.1.1 Option

Dans cette rubrique, définir

1. *Axis limits* : limites dans lequel le dessin de Ω tient (de fait, limites plus ϵ de part et d'autre pour voir les frontières). Par exemple pour `Y` choisir `[-1 pi/2]`.
2. *Grid* : check it. Permet de se repérer facilement. Noter que le pointeur de la souris a ses coordonnées indiquées en haut à droite de la fenêtre.
3. *Grid spacing* : permet de définir l'espacement de la grille. On peut utiliser les possibilités de Matlab. Par exemple, pour Y, on peut définir l'espacement par $-1 : 0.2 : 0, 0 : 0.5 : 2$. Peut être redéfini après chaque dessin (utile avec l'option *snap*).
4. *Snap* : check it, si vous voulez que les rectangles et ellipses soient centrés sur la grille. Dans ce cas les coordonnées de la souris sont données par le point le plus proche d'un noeud de la grille (dont les coordonnées sont indiquées en haut à droite de la fenêtre).

2.1.2 Draw

Cette rubrique correspond aux dessins stylisés de la barre d'outils, et permet de créer les modèles CSG (Constructive Solid Geometry) :

1. *Draw mode* : permettra de revenir au dessin de la géométrie pour modification éventuelle.
2. *Rectangle/Square* : permet de dessiner un rectangle ou carré (avec l'option snap) en partant d'un coin (de la grille avec l'option snap), à l'aide du premier clique de souris (on lache la souris quand on a terminé le rectangle).

3. *Rectangle/Square (Centered)* : Le premier clique de souris donne le centre du rectangle.
   Un rectangle peut être créé directement dans l'environnement Matlab
   »`pderect([x1 x2 y1 y2],'name')`
   ce qui ouvre la fenêtre Pdetool et dessine le rectangle de sommets $(x1, y1)$, $(x1, y2)$, $(x2, y1)$ et $(x2, y2)$ dénommé "name" (optionnel, par exemple : name=R1).
4. *Ellipse/Circle* : parle de lui-même.
5. *Ellipse/Circle (Centered)* : parle de lui-même.
   Un cercle peut être créé directement dans l'environnement Matlab
   »`pdecirc(x,y,R,'name')`
   ce qui ouvre la fenêtre Pdetool (ou ajoute à la fenêtre déjà ouverte) et dessine le cercle de centre (x,y) et de rayon R dénommé name (par exemple name=C2).
   Et pour une ellipse :
   »`pdeellip(x,y,rx,ry,angle,'name')`
   pour ellipse de centre (x,y) axes de longueur rx et ry, et d'inclinaison angle en radian (sens trigonométrique), nommée name (optionel, par exemple E1).
6. *Polygon* : Dessine un polygone fermé. Cliquer à chaque extrémité des lignes formant le polygone. Ces lignes ne doivent pas s'intersecter. Attention : si problème, on doit sortir de «draw - polygon» (par exemple en allant sur «Ellipse») pour revenir à «Polygon».
   Dans l'environnement Matlab :
   »`pdepoly([x1 ... xn],[y1 ... yn],'name')`

Chaque objet est repésenté par un nom (R1 pour le premier rectangle dessiné...). Pour voir les propriétés d'un objet dessiné, double-cliquez sur cet objet. L'ojet final est la réunion de tous les objets dessinés, et dans la barre "Set formula" cette réunion est repésentée par des + (et non des $\cup$), par exemple "R1+E1" (pour dire "R1$\cup$E1").

Si vous souhaitez représenter une exclusion, par exemple "R1-E1", remplacer le signe + dans la barre "Set formula" par le signe $-$. Et pour visualiser le résultat, allez dans la rubrique "Boundary", et selectionnez "Boundary Mode". Revenez ensuite à "Draw Mode".

Pour une intersection utiliser *, par exemple R1*E1, pour signifier "R1$\cap$E1".

2.1.3 Maillage

Il s'agit de choisir un maillage et son degré de raffinement. On commencera toujours par un maillage grossier, ce qui permettra de vérifier qu'on ne s'est pas trompé dans les coefficients ou d'avoir une première idée de la solution.

1. Allez dans `Mesh` puis `Initialize Mesh`. De manière équivalente, cliquez sur le triangle.
2. Éventuellement, choisir ensuite `Refine Mesh` (pas dans un premier temps à cause du temps de résolution que cela risque d'impliquer). De manière équivalente, cliquer sur le triangle divisé en 4.

Remarque 1 : Notez que Matlab ne connaît que les éléments finis P_1. Il n'y a donc pas le choix de l'élément fini. Il n'y a donc pas de paragraphe sur le choix des éléments finis : un tel paragraphe existe bien sûr dans les codes industriels (choix d'éléments finis P_2, Q_1...).

Remarque 2 : dans `Mesh`, il y a l'option `Jiggle Mesh` (i.e. 'secouer légèrement' en français). Si vous l'essayer, elle ne changera rien ici car elle est activée par défaut quand vous maillez. Pour le voir, allez dans `Mesh, Parameters`, ou la case de `jiggle mesh` est checkée (activée). Vous pouvez la déactiver, relancez le maillage, puis `Mesh, Display Triangle Quality`. Puis faite `Jiggle Mesh`, puis de nouveau `Mesh, Display Triangle Quality`. Et comparez.

2.2 Équation aux dérivées partielles

On commence la partie écriture du programme éléments finis proprement dit.

2.2.1 Type de l'équation aux dérivées partielles

Il s'agit de définir le type d'équations qu'on va résoudre (elliptique, parabolique ou hyperbolique, scalaire ou vectorielle) et d'entrer les coefficients de l'équation choisie.

Aller dans `Option`, puis `Application`. Choisir le type des équations que vous voulez résoudre.

2.2.2 Coefficients de l'équation aux dérivées partielles

Allez dans `PDE`, puis `PDE Specification`. Cela ouvre une fenêtre avec éventuellement en haut l'équation type que vous avez sélectionnée (sinon l'équation est donnée dans la doc). Y choisir le type d'équation que vous voulez résoudre (elliptique, parabolique, hyperbolique, calcul des valeurs et vecteurs propres).

Dans cette même fenêtre, entrez les coefficients de l'équation à la main.

Remarque : pour résoudre :

$$-\mathrm{div}(C.\mathrm{grad}u) + au = f$$

avec C matrice 2*2, il suffit d'écrire dans le cadre reservé pour `c` :

`1 2 3 4`

si la matrice C est la matrice $\begin{pmatrix} 1 & 2 \\ 3 & 4 \end{pmatrix}$ (mais `[1 2 ; 3 4]` ne fonctionne pas). Allez voir l'aide de la fonction `assempde.m`.

Et pour entrer les coefficients de l'équation dans l'environnement Matlab, il faut sortir du GUI `pdetool` et se servir du programme :

»`assempde(b,p,e,t,c,a,f)`.

2.2.3 Coefficients de conditions aux limites

Il s'agit d'imposer les conditions aux limites. Dire s'il s'agit de CL de Dirichlet, ou de Neumann, ou mixtes, et d'entrer les valeurs imposées pour ces CL.

1. Allez dans `Boundary Mode`.
2. Cliquez (une fois) sur la partie de la frontière sur laquelle vous voulez imposer des conditions aux limites données. La frontière est définie par morceaux, et pour choisir plusieurs morceaux, maintenez la touche "shift" enfoncée. Pour sélectionner toutes les frontières, après avoir choisi `Draw Mode`, allez dans `Edit` et `Select All`.
3. Allez dans `Boundary`, puis `Specify Boundary Conditions`. Ou bien double cliquez sur la frontière à l'endroit désiré (après avoir éventuellement relaché la touche shift qui a permis de sélectionner différentes parties de la frontière). La ligne du haut donne la condition aux limite choisie, suivant que vous cliquez sur "Neumann", "Dirichlet" ou "Mixed". Et choisir en conséquence les coefficients.

   Remarque : on peut entrer des valeurs sous forme de fonctions. Par exemple, pour `r` on peut entrer `sqrt(x.^2+y.^2)`.

2.2.4 Solveur

C'est la partie résolution de l'EDP définie avec ses conditions aux limites sur le maillage défini ci-dessus.

1. Allez dans `Solve`, puis `Solve PDE`. Ou de manière équivalente, cliquez sur le signe "=".

2.3 Représentation de la solution : postprocesseur

Connaissant la solution sur les sommets du maillage (on a des éléments finis de type P_1), on en déduit la solution sur tout Ω (par interpolation linéaire pour des éléments finis P_1). Il s'agit maintenant de représenter la solution calculée.

1. Allez dans `Plot`, puis `Plot Solution`. On a ainsi une représentation plane de la solution.
2. Allez dans `Plot`, puis `Parameters`. Ou de manière équivalente, cliquez sur le 'dessin' 3D. Choisir alors `Height (3-D Plot)`, puis cliquez sur `Plot`. On obtient une représentation 3D.
3. Cliquez sur le dessin, et sans relacher la souris, la déplacer : on peut voir le résultat sous différents angles.
4. Dans `Plot`, puis `Parameters`, désactivez `color` puis activez `mesh`, et regardez le résultat.
5. Dans le cas où on connaît une solution analytique (ce qui est agréable pour tester ou debugger un code), on peut comparer la solution numérique trouvée avec la solution analytique : dans `Plot Solution`, choisir dans la colonne `property` première ligne `user entry` à la place de `u`. Puis colonne `user entry`, entrez la formule `u-(1-x.^2 - y.^2)` si la solution analytique est $1 - x^2 - y^2$.

2.4 Exercices

Avec utilisation du GUI.

Exercice 1

1. Créez un cercle de centre 0 et rayon 1, puis un cercle de centre 0 et rayon 0,2.
2. Enlevez le petit cercle du grand (pour obtenir une "couronne").
3. Imposez les CL de Dirichlet homogènes sur le grand cercle et celles de Neumann sur le petit.
4. Maillez et raffinez une fois.
5. Choisissez de résoudre l'application "génériс scalar" (dans `Options, Application`), et entrez les coefficients correspondant à l'EDP $-\Delta u + 2u = 3$ (dans `PDE, PDE Specification`).
6. Résolvez, et visualisez en 3D la solution, puis tournez la figure pour vous mettre à l'Azimut 0° et à l'Elevation 70°.

Exercice 2

Même géométrie, mais après avoir choisi l'application "génériс scalar" :

1. Pour les coefficients de l'EDP, choisir de résoudre le problème aux valeurs propres ("eigenmode") correspondant à $-\Delta u + 2u = \lambda u$.
2. Résolvez, et visualisez en 3D la solution (la valeur de la première valeur propre est donnée et la fonction propre correspondante est dessinée).
3. Visualisez la deuxième fonction propre. Puis la troisième fonction propre et tournez cette dernière de 90°. Les 2ièmes et 3ièmes fonctions propres sont-elles visuellement différentes ?
4. Revenez à la deuxième fonction propre et indiquez les axes : sous Matlab :

   » `xlabel('x'),ylabel('y')`.

   Sauvez ce fichier dans `figvp2.fig`.
5. Revenez à la troisième fonction propre et indiquez les axes : sous Matlab :

   » `xlabel('x'),ylabel('y')`.

   Puis dans la figure allez dans "Edit" "Open" pour ouvrir la figure `figvp2.fig`. Comparez.

   La valeur propre trouvée est double, mais on sait a priori que cette valeur propre n'est pas dégénérée (le laplacien est tel que son "inverse" est borné autoadjoint compact positif) , et donc que 2 fonctions propres L^2-orthogonale (non nulles) lui correspondent. Et Matlab les a trouvées.
6. Que se passe-t-il quand on résout $-\Delta u + 10u = \lambda u$ à la place de $-\Delta u + 2u = \lambda u$? Pourquoi les fonctions propres sont-elles identiques, alors que les valeurs propres sont toutes décalées de +8 ?
7. Combien y-a-t-il de valeurs propres à trouver sur un tel maillage, et comment demander au GUI de les calculer puis de les représenter ?

   Réponses : pour un pb de Dirichlet, la dimension de l'espace des solutions est égal au nombre de fonctions de bases. Ici ce sont des fonctions P_1, les fonctions "chapeau", en nombre le nombre np de points intérieurs, et il y a donc np fonctions propres 2 à 2 orthogonales, et np valeurs propres (comptées autant de fois que leur multiplicité). Pour demander au GUI d'en calculer plus, allez dans `Solve, Parameters`, puis demander un intervalle de recherche plus grand. Exportez la solution, puis demander la taille de l qui contient les valeurs propres trouvées.

Remarque : Sur la Doc papier, le dessin proposé en couverture n'est autre que la représentation de la 1ère fonction propre du Laplacien correspondant à 2 géométries différentes. Il se trouve que ces deux géométries ont mêmes valeurs propres (mêmes fréquences) : cela illustre un problème qui n'a été résolu qu'en 1992, problème énoncé sous la forme "Peut-on entendre la forme d'un tambour ?". Cela voulait dire : Est-ce que deux tambours différents émettent des fréquences différentes ? De telle sorte qu'à l'aide d'un micro enregistrant le bruit émis par un tambour on peut reconstituer la forme de ce tambour ? La réponse est donc "non", et l'exemple donné par Matlab le montre. Les applications de ce problème sont nombreuses. Par exemple, quand on détecte un sous-marin, peut-on savoir de manière certaine de quel type (forme) de sous-marin il s'agit ?

Exercice 3

Même géométrie que dans l'exercice 1, mais après avoir choisi l'application "génériс scalar" :

1. Pour les coefficients de l'EDP, choisir de résoudre le problème hyperbolique correspondant à $4\frac{\partial^2 u}{\partial t^2} - \Delta u + 2u = 3$.
2. Résoudre. Comme indiquez, vous devriez voir le résultat au temps `t=10`.
3. Animez la solution (animation permettant de voir la solution à chaque pas de temps). Pour cela allez dans `Plot, Parameters`, et choisissez `Animation` en 3-D. Ça devrait être "médiocre".
4. Allez dans `Solve, Parameters`, et changer le pas de temps en 0:.2:10. Résolvez et Animez en 3-D.

Exercice 4

Même exercice que le 1, mais en imposant des conditions de Neumann sur tous les bords, et pour l'EDP $-\Delta u = 3$.

Normalement vous devriez obtenir n'importe quoi (de l'ordre de $u \sim 10^{15}$). Pourquoi ?

Réponse : on ne résout pas un problème elliptique. En effet, la forme bilinéaire
$a(u,v) = \int_\Omega \text{grad}u.\text{grad}v\, d\Omega$
n'est pas elliptique dans $H^1(\Omega)$. En particulier, si u est solution, alors $u+c$ est solution pour toute fonction constante $= c$. Il n'y a donc pas unicité de la solution, et le problème est en particulier mal posé.

3 Problèmes non traités par le GUI

Dans le paragraphe précédent, on a pu traiter des problèmes répertoriés dans le Pdetool. Si la géométrie qu'on veut traiter est différente, on ne peut plus utiliser le GUI de Pdetool.

3.1 Géométries

Celles qu'on ne peut pas créer à l'aide du GUI Pdetool. Il faut donc faire des fichiers ".m". Ces fichiers sont tous des fichiers "`function`" de la forme `pdegeom.m` :

`function [x,y]=pdegeom(bs,s)`

bs = numéros des `boundary segment` (ou border segment).

s = paramètre de coordonnée curviligne permettant de décrire la courbe $(x(s), y(s))$ pour $s \in [a,b]$ correspondant au segment numéro bs.

Remarque : on peut créer un exemple, et donc le modifier, à partir de Pdetool, "`Boundary`", "`Boundary Mode`" puis, "`Boundary`", "`Export Decomposition...`", ce qui crée une matrice `g` de décomposition du domaine (la matrice `g` est appelée `dl` dans l'aide "decsg.html" ou dans "pdegeom.html" (!)). puis on crée le fichier "`ficnom.m`" à l'aide de

» `wgeom(g,'ficnom')`

On visualise le résultat à l'aide de :

» `pdegplot('ficnom')`

Et on modifie cet exemple à volonté. Le nom générique "`ficnom.m`" a été appelé "`pdegeom.m`" par Matlab (ce dernier nom correspond donc à un fichier virtuel).

Exemple : création du domaine carré $[0,1] * [0,1]$ modifié : segment supérieur remplacé par la sinusoïde $y = .2*\sin(\pi x)$ (cela donnera la base d'un conduit périodique très utilisé numériquement).

On commence par créer le fichier "`carmodg.m`" : ce fichier contient toutes les informations suffisantes pour créer le maillage.

1. Sous Matlab, lancer :

   » `pderect([0 1 0 1])`

   (ou bien lancer `Pdetool` et dessiner le carré).
2. Puis on choisit "Boundary mode", puis dans "Boundary" on choisit "Export Decomposed Geometry...".
3. Puis sous Matlab lancer,

   » `wgeom(g,'carmodg')`

   On a ainsi créé le fichier `carmodg.m` dans le répertoire courant. Puis on ouvre le fichier "carmodg.m" (on peut visualiser : » `pdegplot('carmodg')`).

4. On regarde la matrice `d` donnée dans ce fichier, dont chaque colonne (composée de 4 lignes) correspond à un côté de la géométrie :

   1ère ligne = coordonnée curviligne de début de segment.

   2ème ligne = coordonnée curviligne de fin de segment.

   3ème ligne = label de la région à gauche (sens trigo).

   4ème ligne = label de la région à droite (sens trigo).

   On rappelle que le paramétrage curviligne d'une courbe est donné par : $s \in [a,b] \to (x(s),y(s))$. Et dans la 1ère ligne de la matrice `d` on trouve a et dans la 2ème ligne on trouve b. La coordonnée curviligne `s` ne devra être utilisée que pour des valeurs de `s` $\in [a,b]$.

5. Pour repérer le 1er côté, il suffit donc de lancer Matlab :

   » `[x,y]=carmodg(1,[a b])`

   qui donne les coordonnées cartésiennes des points de coordonnées paramétriques curvilignes `s=a` et `s=b` extrémités du premier segment (donné par la première colonne de la matrice `d`). Par exemple c'est le bas du carré.

   Explication du fonctionnement de `carmodg.m` : on lance »`[x,y]=carmodg(1,[0 1]))`. Dans le programme `carmodg.m`, on a utilisé $bs = 1$ et $s = [0,1]$. Le programme commence par transformé bs : cela devient la matrice ligne de longueur $length(s)$ ($= 2$ ici) dont tous les termes valent l'input bs ($= 1$ ici). Et l'expression

   ```
   ii=find(bs==1) ;
   ```

   donne `ii=1,2`, tous les coeff de la matrice ligne bs valant 1. Puis »`s(ii)` donne `s(1)=0` et `s(2)=1`, d'où les `x(ii)` et `y(ii)` pour $ii = 1,2$. On déduit les extrémités du premier segment. Puis par exemple `find(bs==2)` donne `ii=[]` et donc aucune expression n'est créée.

6. On veut modifier le haut du carré : on vérifie que c'est le segment 3 :

   » `[x,y]=carmodg(3,[0 1])`

   Comme paramètrage de la courbe on utilise naturellement s dans $[0,1]$. On remplace alors la partie correspondant au segment 3 par :

   ```
   ii=find(bs==3); % en general, ii=1 si bs=3 (est un scalaire)
   if length(ii), % vrai si ii~=[]
      x(ii)=1-s(ii); % ou parcourt de droite à gauche
      y(ii)=1+.2*sin(2*pi*x(ii)); % ordonnée correspondante.
   end
   ```

   On visualise le résultat avec

   » `pdegplot('carmodg')`

Remarque. Exemple dangereux : Attention à l'exemple de la cardioïde donné dans Matlab, voir `pdegeom.html` : il mélange beaucoup d'informations, due à un choix assez peu judicieux de la fonction cardioïde `cardg.m` donnée en exemple. La cardioïde choisie a pour équation polaire $r = 1 + 2\cos(\theta)$.

Voici son fonctionnement :

1. `nbs=cardg` (pas de 'argin') : donne le nombre de segments qu'on doit entrer pour compatibilité avec Pdetool (doit être ≥ 3). Ici on trouve $nbs = 4$. Résultat attendu (usuel).
2. `d=cardg(bs)` (un seul 'argin') : si bs est un entier, avec $1 \leq bs \leq nbs$ (sinon erreur), d est un vecteur colonne donnant les propriétés du segment bs. Et si bs est un vecteur (par exemple bs =`[1 3]`), alors d est une matrice dont chaque colonne décrit le segment $bs(i)$. Comportement attendu (usuel).
3. `[x,y]=cardg(bs,s)` (deux 'argin') : s est la coordonnée curviligne pour laquelle on veut connaître les valeurs cartésiennes de x et y correspondantes. Attention, pour un segment bs donné (segment 1 à 4) ne peut correspondre que s dans l'intervalle $[d(1,bs),d(2,bs)]$. Et ce sont ces valeurs qui sont utilisées par les programmes se servant de `cardg.m` (malheureusement, il n'y a pas de test d'exclusion, disant que si $s \notin [d(1,bs),d(2,bs)]$, alors le programme s'arrête (erreur)).

   Remarque : Vue la définition de `cardg.m` on peut très bien demander `[x,y]=cardg(1,pi+1)` bien que $\pi + 1 \notin$ [segment 1]. Et ici bs semble ne pas intervenir (on obtient le même résultat avec `[x,y]=cardg(2,pi+1)`). Si : il intervient, mais uniquement lorsqu'on se servira de programme utilisant `cardg.m`, comme par exemple `[p,e,t]=initmesh('cardg')`. D'ailleurs

noter que x et y sont définis indépendamment du segment choisi, car les paramètres s ont été choisis pour que ça marche.

D'autre part, pour la cardioïde d'équation $r = 1 + 2\cos(\theta)$ pour $\theta \in [0, 2\pi]$, il se trouve que θ est proportionnel au paramètre intrinsèque de la courbe, d'où l'utilisation de `pdearcl.m`. L'usage du paramètre intrinsèque est rarement pratique.

Donc, considérez l'exemple donné par Matlab avec discernement.

3.2 Maillage qui s'en déduit

Une façon de voir si on a mal décrit la géométrie (en particulier le sens de parcourt des différents côtés) est de mailler la géométrie :

```
» [p,e,t]=initmesh('carmodg');
» pdemesh(p,e,t)
» [p,e,t]=refinemesh('carmodg',p,e,t);
» pdemesh(p,e,t)
» p=jigglemesh(p,e,t)
```

La fonction `initmesh` permet d'obtenir un premier maillage décrit à l'aide de :

1. une matrice p de taille $2 * np$ où np est le nombre de points constituant le maillage : 1ère ligne = abscisse des points, 2ème ligne leur ordonnée,
2. une matrice e de taille $7 * ne$ où ne est le nombre de côtés des triangles qui sont sur la frontière $\partial\Omega$: voir la description dans `initmesh.htm`,
3. une matrice t de taille $4 * nt$ où nt est le nombre de triangles : voir la description dans `initmesh.htm`.

La fonction `pdemesh` trace le maillage.
La fonction `refinemesh` raffine le maillage.
Et la fonction `jigglemesh` 'secoue' le maillage de manière à avoir des triangles le plus 'équilatéral' possible (pour obtenir des maillages dits 'réguliers' et ainsi avoir a priori une erreur d'interpolation la plus faible possible).

3.3 Conditions aux limites

Il faut maintenant disposer d'un fichier "*.m" décrivant les conditions aux limites. Les conditions aux limites sont entrées sous deux formes :

$$hu = r \quad \text{(CL de Dirichlet)}, \tag{3.1}$$

$$\vec{n}.(c.\text{grad}u) + qu = g \quad \text{(CL mixte)}. \tag{3.2}$$

(Cette condition est une condition de Neumann quand $q = 0$, et correspond au problème $-\text{div}(c.\text{grad}u) + ... = ...$ où c est un scalaire ou une matrice $2 * 2$.)

Sous Matlab, lancez :

1. » `pderect([0 1 0 1])`
2. imposez les conditions aux limites ci-dessus.
3. Puis dans "Boundary", cliquez sur "Export Decomposed Geometry...".
4. Puis sous Matlab lancer,

   » `wbound(b,'carmodb')`

   On a ainsi créé le fichier `carmodb.m` dans le répertoire courant. Puis on ouvre le fichier "carmodb.m". Les argout de la fonction "carmodb.m" seront utilisés lors de l'assemblage du membre de droite de l'EDP, et sont décrit dans le fichier d'aide correspondant, à savoir "assemb.htm" (attention à la description de "assemb.htm" qui n'est pas toujours bonne). Pour nous, il s'agit de modifier correctement la matrice `bl` du fichier "carmodb.m" qu'on vient de créer.

Ici, ce fichier est conservé, car le fait que la limite supérieur soit un segment de droite ou une sinusoïde n'intervient pas.

Si on doit modifier le fichier "carmodb.m", voir Annexe 2.

3.4 Coefficients des équations

On ne peut a priori exporter du "pdetool" que les coefficients des EDP scalaires. Dans ce cas, allez dans `PDE, Export PDE coefficients...`.

On pourra se reporter aux descriptions proposées dans
- "assempde.htm" pour les équations elliptiques,
- "parabolic.htm" pour les équations paraboliques,
- "hyperbolic.htm" pour les équations hyperboliques,
- "pdeeig.htm" pour les problèmes spectraux (calcul des valeurs et vecteurs propres),
- "pdenonlin.htm" pour les équations nonlinéaires (à coefficients variables),
- "adaptmesh.htm" pour après un premier calcul, raffiner le maillage uniquement là où la solution varie beaucoup (a un fort gradient).

On décrit brièvement "assempde.m" pour son caractère générique, correspondant à la résolution de :

$$-\mathrm{div}(c\,\mathrm{grad}u) + au = f, \tag{3.3}$$

On regarde deux de ses formes possibles. (Pour les autres formes, voir "assempde.htm" : en particulier, les argin optionnels supplémentaires `u` et `time` servent aux itérations de Neuwton dans le cas non linéaire.)

• Tout d'abord :

```
u=assempde(b,p,e,t,c,a,f)
```

1. `b` est soit un fichier de CL (comme b='carmodb.m'), soit la matrice `b` dont le fichier est issu.
2. `p, e, t` sont décrits plus haut (points, edges and triangles de "initmesh").
3. `c, a, f` sont donnés principalement : soit sous forme de scalaires (constantes), soit sous forme de chaînes de caractères (soit, voir "assempde.htm").

Par exemple :

```
» u=assempde(b,p,e,t,c,a,f)
```

ou bien

```
» u=assempde(b,p,e,t,'1+10*x.^2',0,10) ;
```

Et représentation de la solution à l'aide de :

```
» pdesurf(p,t,u)
```

• Puis on regarde la forme :

```
[K,F]=assempde(b,p,e,t,c,a,f)
```

qui donne une représentation de la manière dont le problème est décomposé. On trouve alors la solution à l'aide de :

```
u=K\F
```

(qui résout le problème $Ku = F$).

Application : On veut imposer des conditions de Neumann partout sauf aux quatre extrémités du carré modifié :

On récupère les numéros des extrémités du carré, ici ce sont les points 1, 2, 3 et 4, et on applique la méthode simple :

```
» for i=1 :4, K(i,i)=10^10 ; F(i)=0 ; end
» u=K\F ;
» pdesurf(p,t,u)
```

ce qui donne des valeurs pour `u(1),...,u(4)` de l'ordre de 10^{-10}.

4 Annexe 1 : Détails sur le fonctionnement de Pdetool

Voir le fichier "decsg.htm", décrit en partie ici. Cette description est donnée à titre d'information. Elle ne servira qu'aux lecteurs désireux de modifier le fonctionnement du GUI Pdetool.

Ici, on prend l'exemple simple d'une géométrie constitué d'un cercle et d'un rectangle :

» `pdecirc(0,0,1)`, puis

» `pderect([0.1,1.1,-.1,.9])`.

Ce que crée Pdetool : Description matricielle. Pdetool crée :

1. Une matrice gd de description (called a Geometry Description matrix). Chaque colonne correspond à un objet. La 1ère ligne de chaque colonne indique la nature de l'objet : 1=cercle, 2=polygone, 3=rectangle, 4=ellipse. Pour notre exemple, il y a 2 colonnes et
$gd = \begin{pmatrix} 1 & 3 \\ x & 4 \\ y & xr1 \\ R & xr2 \\ 0 & xr2 \\ 0 & xr1 \\ 0 & yr1 \\ 0 & yr1 \\ 0 & yr2 \\ 0 & yr2 \end{pmatrix}$. Voir la doc.
2. Une matrice ligne sf où est écrit la "Set formula" (=barre d'adresse de Pdetool). Dans notre cas,
$sf = C1 + SQ1$ (matrice "string").
3. Une matrice ns (=Name Space matrix), chaque colonne donnant le code ascii des termes de sf, ici 1ère colonne pour C1 et 2ème colonne pour SQ1 :
$sf = \begin{pmatrix} 67 & 83 \\ 49 & 81 \\ 0 & 49 \end{pmatrix}$.
On rappelle que connaissant le code ascii, la lettre est retrouvée avec "setstr", par exemple,

```
» setstr(67)
ans=
C
```

Et la commande réciproque est la commande "abs", par exemple

```
» abs('C')
ans=
67
```

Un exemple de ces matrices peut être obtenu sous Pdetool en choisissant "Draw", "Draw Mode", puis "Draw", "Export Geometry Description".

Utilisation de gd, sf, ns : Decomposed geometry matrix dl.

À l'aide de la fonction "decsg" (Decompose Solid Geometry) : » `dl=decsg(gd,sf,ns)`. Chaque colonne de la matrice dl correspond à une ligne de la géométrie. Dans notre exemple, il y a 6 segments de droite dus au rectangle (avec intersections par le cercle), il y a 5 (et oui !) arcs de cercle (pour un cercle seul, il y a 4 arcs de cercle correspondant aux 4 quarts, ici le cercle a été partagé en 4+1 arcs, un de ces arcs correspondant à l'intersection avec le rectangle). Donc 11 colonnes en tout.

1ère ligne = type de ligne. 1=cercle, 2=line, 4=ellipse.
2ème ligne = xdébut
3ème ligne = xfin
4ème ligne = ydébut
5ème ligne = yfin
6ème ligne = label de région gauche (sens trigo), ici 1, 2, ou 3 pour l'intersection.
7ème ligne = label de région droit (sens trigo), ici 1, 2, ou 3 pour l'intersection.
8ème ligne = xr = abscisse centre du cercle (0 si ce n'est pas un cercle)
9ème ligne = yr = abscisse centre du cercle (0 si ce n'est pas un cercle)
10ème ligne = R = rayon du cercle (0 si ce n'est pas un cercle)

5 Annexe 2 : la matrice *bl*

Description de la matrice `bl` donnée dans le fichier des conditions aux limites créé par `wbound(b,'nomfic')`.

Poursuivons l'exemple de "carmodg.m", pour lequel on prendra les conditions de Dirichlet sur le côté sinusoïdal :

$$1.3u = s^2 \quad \text{(CL de Dirichlet avec : } \texttt{h=1.3} \text{ et } \texttt{r=s.^2)}, \tag{5.1}$$

(avec s paramètre de la courbe) et les conditions mixtes sur les trois côtés droits :

$$\vec{n}.(c.\text{grad}u) + \frac{1}{(1+x^2)}u = 1.32 \quad \text{(CL mixte avec : } \texttt{q=1./(1+x.^2)} \text{ et } \texttt{g=1.32)}. \tag{5.2}$$

Avec les variables `h, r, q, g` définies ci-dessus, on crée une fonction appelée sous la forme `[q,g,h,r]=carmodbd(p,e,u,time)`, et la fonction `carmodbd.m` créée décrit les valeurs `h, r, q, g` qu'on aurait éventuellement à modifier.

Il y a 4 colonnes, comme dans la matrice `g` décrite plus haut, chaque colonne correspondant à un segment du bord, et décrit les CL.

- 1ère ligne = dimension de l'inconnue u : ici on résout un problème scalaire, donc dim = 1. (Pour les systèmes, on aurait deux inconnues u et v et donc dim = 2, mais ici on décrit ce qui se passe pour le cas scalaire, et on renvoie à `assemb.htm` pour les autres cas).
- 2ème ligne = le nombre de conditions de Dirichlet : ici, 1 ou 0. (Dans le cas d'un système, se serait 2, 1 ou 0 suivant que l'on impose à la fois une condition de Dirichlet sur u et v ou non.)
- 3ème ligne = le nombre de caractères qui représente la quantité q, ici il y a 11 caractères sur les côtés 1, 2 et 4, voir (5.2). Par défaut, le résultat est 1 si on a les CL de Dirichlet.
- 4ème ligne = idem pour la quantité g, ici 4 sur les côtés 1, 2 et 4, voir (5.2). Par défaut, le résultat est 1 si on a les CL de Dirichlet.
- 5ème ligne = si on a des CL de Dirichlet, alors on donne le nombre de caractères pour h, ici 3, voir (5.1).
  Si on a des CL mixtes, alors dans les lignes qui suivent (ici les 11 lignes qui suivent) on écrit q sous forme ascii, puis les lignes qui suivent (ici les 4 lignes suivantes) on écrit g sous forme ascii.
- 6ème ligne = si on a des CL de Dirichlet, alors on donne le nombre de caractères pour r, ici 4, voir (5.1).
  Si on a des CL mixtes, voir description au point ci-dessus.
- 7ème et 8ème lignes = si on a les CL mixtes, c'est déja décrit, si on a les CL de Dirichlet, on a le 0 en ascii (à savoir 48).
- 9ème ligne et suivantes = si on a les CL mixtes, c'est déja décrit, si on a les CL de Dirichlet, les (ici 3) lignes qui viennent décrivent h en ascii, et les (ici 4) lignes suivantes décrivent r en ascii.