ANNEXE C

Création des fichiers de données

Cette annexe décrit une procédure permettant la création des fichiers de données nécessaires aux codes 2D associés à l'ouvrage. Les différences étant minimales, on se concentre d'abord sur les problèmes d'élasticité.

C.1 Définition des données et structure des fichiers de données

La mise en forme des données est présentée en s'appuyant sur un exemple : analyse d'une plaque trouée.

C.1.1 Définition de l'exemple

On se propose d'analyser la structure de la figure C.1. Il s'agit d'une plaque épaisse trouée, constituée d'un matériau élastique linéaire fictif (module de Young $E = 1$, coefficient de Poisson $\nu = 0.3$), traitée dans le cadre des déformations planes. Le contour intérieur (trou) est soumis à une pression $p = 2$. Sur les bords de droite et de gauche, les déplacements horizontaux sont imposés (nuls à gauche, égaux à 0.5 à droite), les déplacements verticaux restant libres sur ces mêmes bords. La résultante des efforts extérieurs en direction verticale est nulle, donc compatible avec l'équilibre ; cependant on voit aisément que la solution en déplacement de la plaque est, en l'absence de liaisons additionnelles, définie à un déplacement rigidifiant vertical $\alpha \underline{e}_2$ près. Pour éviter la singularité associée dans la matrice de rigidité finale de la structure, on empêche ce déplacement en bloquant en direction verticale le sommet inférieur gauche. Graphiquement, dans la figure C.1, on a ajouté le « chariot » horizontal sur ce nœud. On sait *a priori* que l'effort exercé par le « chariot » sur la structure est nul et ne perturbe donc pas la solution.

C.1.2 Fichier de données

Les données nécessaires à l'analyse par éléments finis de cette structure sont mises en forme dans un fichier de données, que l'on appellera `example.inp`. On souligne que les conventions utilisées ici pour la préparation des données ne représentent absolument pas un choix universel, mais seulement une approche raisonnable, quoique certainement améliorable. Les unités des mesure adoptées n'apparaissent jamais ni dans les fichiers de données, ni dans le code, et c'est l'utilisateur qui doit choisir un système d'unités et fournir les données en entrée de manière cohérente ; les résultats en sorties sont alors fournis dans le même système.

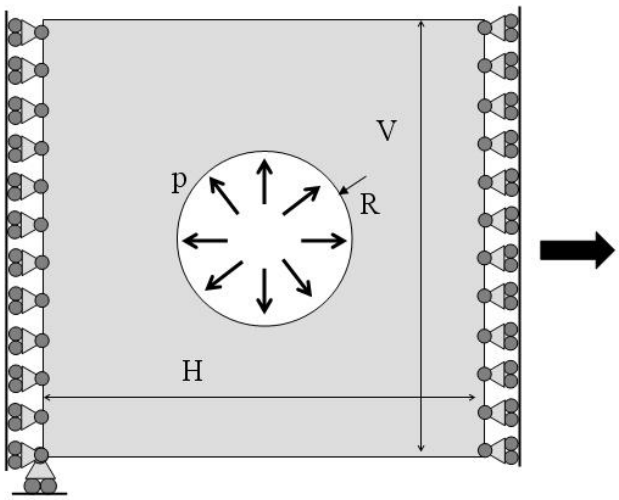

Figure C.1: *Plaque épaisse creuse en déformations planes : problème exemple*

Fichier `example.inp` :

```
example.msh

*ANALYSIS
STATIC,TYPE=PLANESTRAIN
**

*MATERIAL,TYPE=ISOTROPIC
YOUNG=1.
POISSON=.3
**

*SOLID
ELSET=4
**

*DBC
ELSET=1,DIR=1,VAL=.5
ELSET=2,DIR=1,VAL=0.
NODE=1,DIR=2,VAL=0.
**

*TBC
ELSET=3,DIR=0,VAL=-2.
**

*ENDFILE
```

Le fichier de données `example.inp` contient d'abord le nom (`example.msh`) du fichier (placé dans le même répertoire que `example.inp`) définissant le maillage de la structure analysée, dont le contenu sera détaillé en section C.1.3. Pour l'instant il suffit de savoir que ce fichier définit notamment quatre entités nécessaires pour formuler correctement l'analyse. Comme expliqué plus loin, les entités (`ELSET`) sont essentiellement des ensembles d'éléments de surface ou de ligne qui discrétisent certaines entités géométriques présentes dans la structure objet de l'analyse. Pour l'exemple considéré, on a :

- entité 1 (`ELSET=1`) : arête verticale de droite ;
- entité 2 (`ELSET=2`) : arête verticale de gauche ;
- entité 3 (`ELSET=3`) : cercle intérieur sur lequel s'exerce la pression ;
- entité 4 (`ELSET=4`) : surface de la plaque.

On y définit aussi le nœud 1 : sommet en bas à gauche sur lequel on a bloqué le déplacement vertical.

La section `*ANALYSIS` indique que la condition d'équilibre cherchée est statique et que le problème est en déformation planes (l'autre possibilité étant `TYPE=PLANESTRESS` pour des analyses en contraintes planes).

La section `*MATERIAL` donne les propriétés du matériau, isotrope étant la seule possibilité développée dans la version actuelle des codes.

La section `*SOLID` sert à définir le numéro de l'entité (ensemble d'elements) qui constitue la surface de la structure à analyser ; dans le cas présent, comme anticipé, c'est l'entité 4.

La section `*DBC` définit les conditions aux limites en déplacement. On impose le déplacement des nœuds qui se trouvent sur l'entité numéro 1, en direction 1 et valeur `VAL` (dans ce cas 0.5) ; on bloque les nœuds sur l'entité numéro 2, en direction 1 ; enfin, on bloque le déplacement vertical sur le nœud 1.

La section `*TBC` définit les conditions aux limites en efforts (distribution de vecteur contrainte, appelé aussi traction). Dans l'exemple traité, une traction est imposée sur la ligne physique numéro 3, selon la direction normale (`DIR=0`), avec une intensité -2 (le signe négatif de cette donnée indiquant que l'effort est imposé selon la direction opposée à celle de la normale). La normale $\underline{n}$ à la ligne est définie selon la convention suivante : si $\underline{e}_3$ est la direction sortante de la surface et $\underline{t}$ la tangente à la ligne dans la direction positive (comme défini dans le fichier maillage), le trièdre $(\underline{n}, \underline{t}, \underline{e}_3)$ est direct. Pour définir la direction des efforts extérieurs on a deux autres possibilités : `DIR=1` ou `DIR=2` pour désigner une traction exercée suivant la direction $\underline{e}_1$ ou $\underline{e}_2$, respectivement.

Dans l'état actuel du code, on a fait le choix de ne permettre que des tractions constantes sur chaque entité. Prendre en compte une traction variable demande soit de modifier le code, soit de la représenter constante par morceaux. S'il y a plusieurs `ELSET`s chargés, il faut les ressembler tous dans la même section `*TBC`, une ligne par `ELSET`.

Par convention, une portion du bord de la structure sur laquelle aucune condition n'est explicitement imposée est un bord libre (vecteur contrainte imposé nul).

C.1.3 Maillage

L'opération de maillage doit créer les entités géométriques nécessaires au fichier de données et à l'analyse. On utilise ici le code `GMSH v.1.60.1` comme générateur de maillage.

Ce code est disponible gratuitement (avec documentation complète), pour plusieurs systèmes d'exploitation, à la page internet `www.geuz.org/gmsh/`. La version utilisée par les codes MATLAB associés à cet ouvrage est également fournie dans la distribution (voir section B.3). Un petit guide qui se limite à montrer comment utiliser GMSH pour l'exemple considéré est proposé ici. L'utilisation d'un code extérieur se justifie par la grande liberté que l'on obtient dans la génération des problèmes et des maillages associés, mais nécessite un effort additionnel pour en comprendre les potentialités et les conventions et les adapter aux nécessités des codes d'initiation présentés dans cet ouvrage. Les fonctions de GMSH utilisées dans ce document représentent une partie *minimale* du potentiel de ce programme. D'autres exemples seront présentés dans la suite, et le lecteur intéressé approfondira ses connaissances grâce à la documentation fournie avec GMSH.

On montre ici comment écrire le fichier `example.geo` qui contient les informations géométriques et topologiques nécessaires a GMSH pour creer le maillage de la structure choisie. La succession d'opérations est classique. Pour définir un maillage il faut d'abord introduire la géométrie de la surface. La surface étant plane, elle est complètement définie par la courbe fermée orientée qui constitue son bord. Celle ci est, à son tour, la réunion de lignes orientées. Dans l'écriture du fichier `example.geo`, on suit exactement cette procédure, mais dans le « sens » opposé.

On commence par définir des paramètres utilisés pour simplifier l'écriture du fichier : d'abord les dimensions `H` et `V` de la plaque et le rayon `R` du trou circulaire ; ensuite les paramètres de finesse `lc1` et `lc2` qui servent à imposer la finesse du maillage :

```
// global dimensions

H=10;
V=10;
R=2;

// mesh parameters

lc1=3;
lc2=2;
```

On spécifie ensuite les points qui sont nécessaires pour définir les lignes. Chaque point a trois coordonnées et un paramètre qui sert à « guider » la finesse du maillage en ce point :

```
// point coordinates

Point(1) = {-H/2,-V/2,0,lc1};
Point(2) = { H/2,-V/2,0,lc1};
Point(3) = { H/2, V/2,0,lc1};
Point(4) = {-H/2, V/2,0,lc1};

Point(5) = { 0, 0,0,lc2};
Point(6) = { R, 0,0,lc2};
Point(7) = { 0, R,0,lc2};
Point(8) = {-R, 0,0,lc2};
Point(9) = { 0,-R,0,lc2};
```

On poursuit par la définition des lignes. Un segment de droit est défini par les deux extremités ; un arc de cercle (d'ouverture strictement inférieure à $180°$) est défini par la première extrémité, le centre, et la deuxième extrémité, dans cet ordre. Enfin, pour creer une surface, il faut definir les courbes fermées qui constituent son contour au moyen de la commande `Line Loop` :

```
// lines and line loops

Line(1) = {1,2};
Line(2) = {2,3};
Line(3) = {3,4};
Line(4) = {4,1};
Line Loop(5) = {1,2,3,4};

Circle(6) = {6,5,7};
Circle(7) = {7,5,8};
Circle(8) = {8,5,9};
Circle(9) = {9,5,6};
Line Loop(10) = {-6,-7,-8,-9};
```

Dans le cas présent le bord de la surface est constituée de deux contours disjoints. Par convention, tout contour est orienté de sorte qu'un parcours dans le sens positif laisse la surface sur la gauche. On a donc inversé (à l'aide du signe moins), l'orientation des lignes élémentaires pour créer le contour interne avec l'orientation conforme à cette convention (bien que cela ne soit pas strictement nécessaire dans `GMSH`).

Enfin on définit la surface par la donnée des contours extérieur et intérieur, dans cet ordre :

```
// surface

Plane Surface(1) = {5,10};
```

Les entités crées jusqu'a maintenant sont dites *élémentaires* : points, lignes et surfaces (et, en 3D, volumes définis par la donnée des surfaces constituant leur frontière). Les entités élémentaires de chaque type (point,ligne,surface) sont identifiées par simplicité avec un numéro croissant dans les ensembles du même type.

On peut aussi créer des entités *physiques* qui combinent les entités élémentaires déjà introduites. Cette possibilité s'avère souvent utile pour la création de maillages plus complexes, et on en verra un exemple en mécanique de la rupture. C'est pour cette raison qu'on demande à l'utilisateur de définir comme entités physiques toutes les entités dont on a besoin pour la constitution du fichier de données `example.inp`, à part les nœuds :

```
// physical entities

Physical Line(1) = {2};
Physical Line(2) = {4};
Physical Line(3) = {-6,-7,-8,-9};
Physical Surface(4) = {1};
```

Par convention, les entités physiques sont ici identifiées avec un numéro croissant (à partir de 1) indépendamment du type. Si l'on définit des entités physiques, elles sont les seules quantités transcrites dans le fichier sortie, toujours en-dehors des nœuds. Comme attendu

à partir des informations présentes dans le fichier `example.inp`, on a dû ici définir quatre entités physiques.

Pour générer le fichier de maillage une fois `example.geo` écrit, on lance GMSH. Deux fenêtres s'ouvrent : la fenêtre graphique et la fenêtre menu avec le *menu module* qui contient "Geometry" par défaut. On ouvre le fichier `example.geo` avec File → Open et la géometrie apparaît dans la fenêtre graphique.

Dans le *module menu*, choisir Mesh et cliquer sur 2D produit le maillage avec éléments triangulaires linéaires T3. Après, cliquer sur Save demande à GMSH de sauver les informations (coordonnées des nœuds et connectivité des éléments) dans le fichier `example.msh`.

Si on préfère un maillage constitué d'éléments triangulaires quadratiques T6, après avoir choisi le menu Mesh et cliqué sur 2D comme pour un maillage d'éléments T3, il faut cliquer sur Second order avant de sauver le maillage. De cette manière, GMSH modifie le maillage avec éléments T3 en ajoutant des nœuds au milieux des cotés et transforme donc les éléments en T6.

Si l'on demande a GMSH de générer un maillage de T3, sa sortie consiste en le fichier `example.msh` expliqué rapidement dans la suite. Bien que pénible, la création à la main d'un tel fichier, sans passer par GMSH, est possible et peut notamment être utile pour des petits maillages de test.

Il y a deux sections distinctes. La première, `$NOD`, contient d'abord le nombre N_N de nœuds nécessaires pour définir le maillage suivi par une série de N_N lignes.

```
$NOD
24
1  -5 -5  0
2   5 -5  0
3   5  5  0
4  -5  5  0
...
...
23 -2.805508988531008 -2.805253658366925  0
24 -2.805081497670314  2.805987759665386  0
25  2.805825744133971 -2.80510292539923   0
$ENDNOD
```

Chaque ligne contient le numéro du nœud et ses trois coordonnées. Dans les exemples envisagés les analyses sont planes, et la troisième coordonnée est donc toujours mise à zéro. Cette section est fermée par le mot clé `$ENDNOD`. On remarque que la numérotation des nœuds n'est pas nécessairement consécutive. Par exemple, le nœud 5 est ici absent. Ce nœud, qui représente dans GMSH le centre du cercle et qui est utilisé pour générer ce contour, n'appartient à aucun élément du maillage et n'a donc pas à figurer dans `example.msh`.

La deuxième section, `$ELM`, contient les éléments :

```
$ELM
42
1 1 1 2 2 2 12
2 1 1 2 2 12 13
3 1 1 2 2 13 3
4 1 2 4 2 4 16
```

```
5 1 2 4 2 16 17
6 1 2 4 2 17 1
7 1 3 6 2 18 6
8 1 3 6 2 7 18
...
...
13 1 3 9 2 21 9
14 1 3 9 2 6 21
15 2 4 1 3 7 14 15
...
...
41 2 4 1 3 3 22 13
42 2 4 1 3 2 25 11
$ENDELM
```

Pour cet exemple on a des éléments isoparametriques linéaires à 3 nœuds (T3) pour la surface et à 2 nœuds (B2) pour les lignes. En réalité l'introduction des éléments linéiques est un artifice permettant de faciliter l'imposition des conditions aux limites. Ces éléments ne sont pas « indépendants » des éléments de surface, au sens où chaque élément linéique est un coté d'un élément de surface.

GMSH associe un code à chaque type d'élément : par convention 1 désigne les B2 et 2 les T3.

La première ligne contient le nombre total d'éléments. Les lignes qui suivent $ELM définissent les éléments. Chaque ligne contient :

1. le numéro de l'élément
2. le code de l'élément
3. le numéro de l'ensemble physique auquel l'élément appartient ; ce sont les entités utilisées dans le fichier example.inp
4. un numéro utilisé par GMSH et qui peut être mis a zéro si l'on crée le maillage a la main.
5. le nombre de nœuds contenus dans l'élément, (par exemple 3 pour T3, 2 pour B2...)
6. la liste des numéros globaux des nœuds de l'élément (sa connectivité)

On souligne que les éléments de surface et de ligne sont définis sur le même ensemble de nœuds.

C.1.4 Lecture des données.

Les données écrites dans les fichiers présentés doivent être lues par MATLAB et transformées en données informatiques. La lecture est effectue par la fonction read_input, selon une manière commune à tous les codes décrits dans cet ouvrage. La compréhension de cette fonction n'étant pas centrale dans le développement du cours, elle est laissée comme exercice et on se limite ici à expliquer la forme sous laquelle les données sont disponibles après lecture, c'est à dire après exécution de l'instruction :

```
% reads input files
[analysis materials connec coor dof displ TD]=read_input(pname,fname);
```

La variable `analysis` est une structure (au sens de MATLAB), c'est à dire un objet avec plusieurs champs. Certains champs sont actifs seulement pour des types d'analyses specifiques. Pour l'élasticité linéaire statique les champs actifs sont :

- `analysis.NN`, nombre de nœuds ;
- `analysis.NE`, nombre d'éléments de surface ;
- `analysis.type`, type d'analyse (`PLANESTRAIN` pour l'exemple) ;
- `analysis.nTD`, nombre de segments supportant des tractions imposées ;
- `analysis.Etag`, chaîne de caractère désignant le type d'élément : (`'3'` pour l'élément T3, `'6'` pour l'élément T6) ;
- `analysis.neq` (nombre d'inconnues).

La variable `materials` est une structure contenant les informations sur le matériau :

- `materials.type` (`ISOTROPIC` pour l'exemple) ;
- `materials.young` (module de Young) ;
- `materials.poisson` (coefficient de Poisson) ;
- `materials.A` (matrice des modules d'élasticité).

Rappelons que ces informations sont relatives au problème d'élasticité linéaire considéré dans l'exemple. D'autres informations seront nécessaires plus loin.

La variable `connec(analysis.NE,3)` est la table de connectivité des éléments de surface T3 seulement. `connec(e, :)` donne donc la liste des trois sommets de l'element `e`, ordonnés en sens trigonométrique. A ce propos il faut souligner que le code effectue, simultanément à la lecture des données, une renumérotation consécutive des nœuds et des éléments de surface à partir de 1 ; il n'y a donc pas correspondance directe entre la numérotation utilisée par MATLAB et la numérotation de `GMSH`.

La variable `coor(analysis.NN,2)` contient les coordonnées des nœuds.

La variable `dof(analysis.NN,2)` contient les informations sur la numérotation des inconnues du problème en accord avec les considérations suivantes. Le champ de déplacement implicitement choisi avec les éléments T3 est affine par morceaux et continu. Il dépend seulement des valeurs nodales qui représentent donc les inconnues du problème, à part les nœuds sur les surfaces S_u. Par convention, si une composante de déplacement est imposée, on place un nombre negatif dans la position correspondante de `dof` (voir le chapitre 3 pour une explication plus exhaustive). Autrement on remplit `dof` avec le numéro associé au déplacement inconnu. Le coefficient `dof(nod,j)` se réfère à la composante `j` du déplacement du nœud `nod`, en numérotation MATLAB. Par exemple, sur un nœud `nod` de l'arête à droite `dof(nod, :)` donne un vecteur avec deux composantes, dont la première est négative et la deuxième est le numéro progressif positif associé au déplacement nodal vertical inconnu.

La variable `displ(analysis.NN,2)` est une matrice qui, en sortie de `read_input`, contient des zéros sauf pour les déplacements imposés qui sont fixés par l'utilisateur. Par exemple, toujours pour le même nœud de l'arête de droite, `displ(nod, :)` est après lecture des données une liste de deux composantes dont la première vaut 0.5 et la deuxième zéro.

Finalement, la variable `TD` contient les informations sur les efforts de surface imposés. C'est un vecteur de `analysis.nTD` structures, chacune constituée des champs `dir` (direction de la traction exercé sur l'élément suivant les conventions décrites dans la Section

précédente), `val` (intensité de la traction) et `nodes` (nœuds de l'élément 1D, ordonnes suivant l'orientation de l'élément, sur lequel la traction s'exerce).

C.1.5 Fichiers de données disponibles pour le code `linel_T`

Certains fichiers de données sont fournis avec le code dans le répertoire `input`. La géométrie et les conditions aux limites sont présentées dans la suite.

Plaque avec un trou circulaire : `hole.*`**.** Ce problème est un exemple classique de la théorie de l'élasticité, notamment pour étudier l'effet de concentration des efforts sur le bord du trou. Pour le cas limite d'un trou de rayon R dans une plaque infinie, la solution analytique est connue :

$$\begin{aligned}
\sigma_{rr} &= \frac{\sigma}{2}\left(1-\frac{R^2}{r^2}\right)+\frac{\sigma}{2}\left(1+3\frac{R^4}{r^4}-4\frac{R^2}{r^2}\right)\cos 2\theta \\
\sigma_{\vartheta\vartheta} &= \frac{\sigma}{2}\left(1+\frac{R^2}{r^2}\right)-\frac{\sigma}{2}\left(1+3\frac{R^4}{r^4}\right)\cos 2\theta \\
\sigma_{r\vartheta} &= -\frac{\sigma}{2}\left(1-3\frac{R^4}{r^4}+2\frac{R^2}{r^2}\right)\sin 2\theta
\end{aligned}$$

où σ représente la tension appliquée à l'infini dans la direction $\underline{e}_1$, et θ est la coordonnée polaire angulaire ($\theta = 0$ dans la direction $\underline{e}_1$). Par exemple, la contrainte σ_{xx} pour $\theta = \pi/2$ (sur le bord supérieur du trou) vaut 3σ.

L'exemple fourni concerne une plaque trouée de dimensions finies. Pour aider à vérifier la convergence vers la solution analytique, les données permettent, pour la même plaque

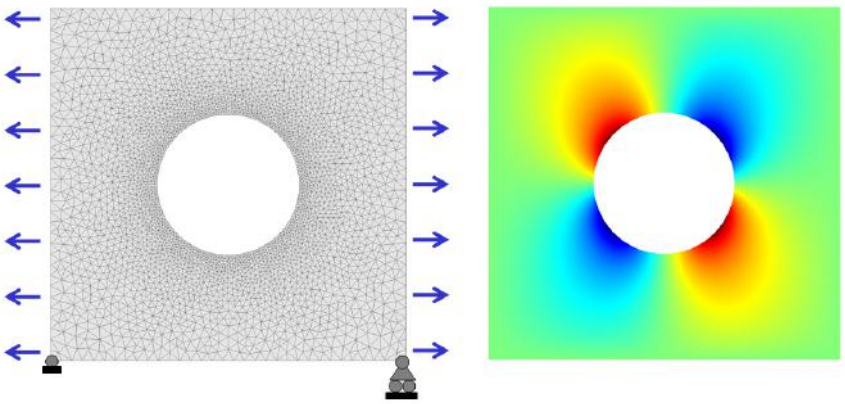

Figure C.2: *Plaque avec un trou circulaire : maillage et contrainte σ_{12}. Version couleur en p. 297.*

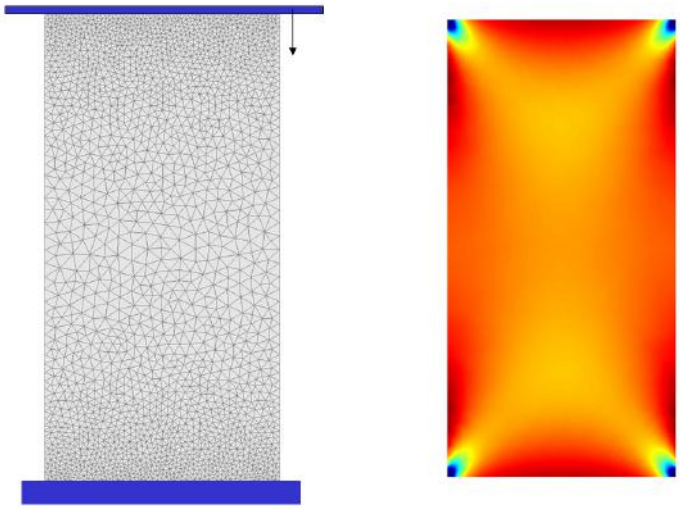

Figure C.3: *Plaque comprimée : maillage et contrainte σ_{22}. Version couleur en p. 297.*

rectangulaire, de choisir deux valeurs du rayon du trou et de jouer sur la finesse du maillage. La figure C.2 a été obtenue pour $\sigma = 1$.

Compression simple avec frottement : `compress.*`**.** Une éprouvette rectangulaire est coincée entre deux plaques rigides (supérieure et inférieure) avec frottement empêchant tout glissement le long de ces plaques. L'analyse est faite en déformations planes, avec $E = 1, \nu = 0,3$. La plaque inférieure est bloquée en déplacement, tandis que la plaque supérieure subit un déplacement vertical vers le bas imposé égal à -1 (Figure C.3).

Éprouvette en traction entaillée : `wedge.*`**.** Cette dernière série de fichiers de données permet l'analyse des effets de la présence d'une entaille dans une plaque rectangulaire (Figure C.4).

C.2 Création du maillage de la plaque fissurée

Les analyses de mécanique de la rupture du chapitre 4 se basent encore sur `linel_T#` et sur les fichiers de pré- et post-traitement. Le maillage se crée comme pour les exemples précédents, la seule différence étant qu'il faut introduire des nœuds doubles sur les lèvres de la fissure. Pour cela on définit deux surfaces différentes, pour les parties supérieure et inférieure, à l'aide de courbes fermées définies comme représenté en figure C.5. Sur la ligne séparant les deux surfaces, ces courbes sont constituées par les mêmes segments, sauf le long

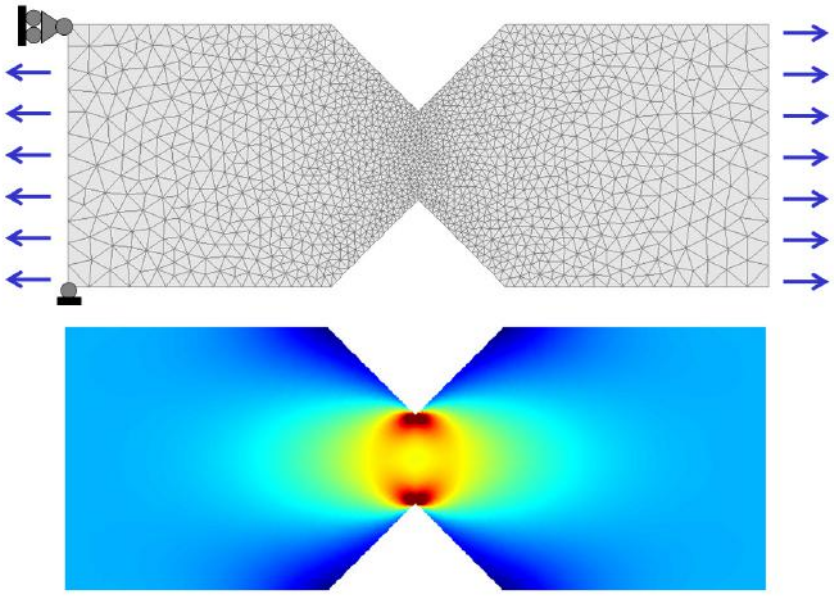

Figure C.4: *Éprouvette avec entaille : maillage et contrainte σ_{11}. Version couleur en p. 298.*

de la fissure, où elles utilisent les deux segments différents `Line(4)` et `Line(10)`. De cette manière, `GMSH` crée automatiquement des nœuds distincts sur les lèvres de la fissure.

Le fichier `fract.geo` contient un exemple de définition (géométrie et topologie) pour `GMSH` :

```
H=5;      //  semiwidth of plate
V=5;      //  semiheight of plate
a=1;      //  semilength of crack  (center crack is at x1=x2=0)

lc1=.5;
lc2=.2;
lc3=.01;  // element size at crack tip

Point(1) = {-H, -V,0.0,lc1};
Point(2) = { H, -V,0.0,lc1};
Point(3) = { H,0.0,0.0,lc2};
Point(4) = { H,  V,0.0,lc1};
Point(5) = {-H,  V,0.0,lc1};
Point(6) = {-H,0.0,0.0,lc2};
Point(7) = {-a,0.0,0.0,lc3};
Point(8) = { a,0.0,0.0,lc3};

Line(1)  = {1,2};
```

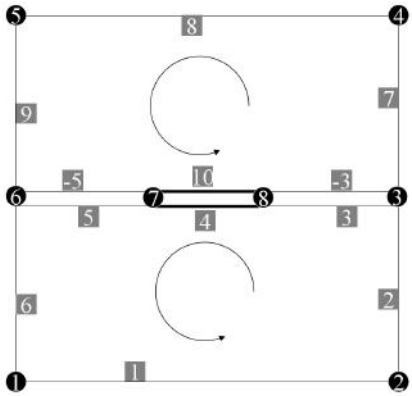

Figure C.5: *Plaque carrée fissurée : définition des courbes fermées pour les deux surfaces.*

```
Line(2)  = {2,3};
Line(3)  = {3,8};
Line(4)  = {8,7};
Line(5)  = {7,6};
Line(6)  = {6,1};
Line(7)  = {3,4};
Line(8)  = {4,5};
Line(9)  = {5,6};
Line(10) = {7,8};

Line Loop(11) = {1,2,3, 4, 5, 6}; // border of lower surface
Line Loop(12) = {7,8,9,-5,10,-3}; // border of upper surface

Plane Surface(1) = {11};          // lower surface
Plane Surface(2) = {12};          // upper surface

Physical Line(1) = {1};           // lower line for traction
Physical Line(2) = {8};           // upper line for traction
Physical Surface(3) = {1,2};
```

Le fichier géométrie `fract_ros.geo` pour le maillage de la figure 4.A2 utilise une procédure similaire, mais la structuration en rosette autour des points de la fissure entraîne des complications techniques mineures dont la lecture est laissée en exercice. Les résultats présentés dans le tableau ont été obtenus avec les paramètres de finesse suivants dans `fract_ros.geo` :

```
lc1=.2; rad1=.01; rad2=1.5*rad1; num=6; lc2=3.14*rad1/num; lc4=H/5;
```

C.3 Fichier de données pour *beam_ldisp_T3*

Comme expliqué dans l'annexe B le code `beam_ldisp_T3` a été développé spécifiquement pour le fichier input `buckling.*` qui suit exactement les memes conventions que les

fichiers entrée pour `linel_T` et `linel_T_fast`.

C.4 Fichier de données pour *plast_T*

Le code `plast_T` utilise en entrée des fichiers similaires à ceux de l'élasticité linéaire. Deux exemples sont proposés dans la distribution : `hole_plast`, qui reprend le problème de la plaque trouée , et `strip_plast`, qui correspond à l'exemple de la section 7.4.1. Les différences avec les analyses élastiques sont concentrées dans le fichier `*.inp` :

```
*ANALYSIS
STATIC,TYPE=PLAST
0.1  0.2  0.3  0.4  0.5  0.6  0.7  0.8  0.9  1.
**

*MATERIAL,TYPE=ISOTROPIC
YOUNG=1
POISSON=.3
HARDENING=0.
SIGMA0=1.
**
```

dans lequel sont définis les paramètres du comportement élastoplastique du matériau et l'histoire de chargement (suite des valeurs de $\lambda(t)$ aux instants discrets utilisés). Pour le jeu de données reproduit ci-dessus, la section `*ANALYSIS` définit 10 pas de chargement, les valeurs λ_i étant régulièrement espacées entre 0 et 1 . Ces valeurs sont lues par la fonction de lecture des données et sauvées dans les champs `materials.H` et `materials.sigma0` de `materials` (pour les paramètres du matériau) et dans le champ `analysis.history` de la structure `analysis` (pour l'histoire de λ).

C.5 Fichier de données pour *tire_contact_T3*

Comme pour le code `beam_ldisp_T3`, même dans ce cas l'utilisateur n'a pas de liberté dans le choix du fichier de données (qui est nécessairement `tire.inp`), mais il peut varier certains paramètres.

```
*ANALYSIS
TYPE=DYNAMIC
TIMEF=5.
DT=0.0005
PENALTY=50
GRAV=9.81
**

*INITIAL
VELOCITY,DIR=2,VAL=-1
**

*MATERIAL,TYPE=ISOTROPIC
YOUNG=100
POISSON=.2
```

```
RHO=.4
**
```

Les nouveaux champs de la structure `analysis` sont :

- `analysis.tf`, la durée totale de l'analyse (tirée de `TIMEF=5.`) ;
- `analysis.dt`, le Δt à utiliser (`DT=0.0005`) ;
- `analysis.penalty`, le coefficient de pénalisation pour le contact (`PENALTY=50`) ;
- `analysis.grav`, l'accélération de la pesanteur (`GRAV=9.81`) ;
- `analysis.invel`, une liste des deux composantes de la vitesse initiale supposée homogène (dans le cas présent vitesse verticale égale a -1).

La structure `materials` est enrichie par le champ `materials.rho` (masse volumique du matériau), nécessaire pour le calcul de la matrice de masse (`RHO=.4`).