ANNEXE B

Liste et description des codes

B.1 Informations générales sur l'utilisation des codes

Les codes MATLAB utilisés dans le cours ont une finalité exclusivement pédagogique. Il privilégient la simplicité, ne sont en général pas optimisés et sont très rudimentaires en comparaison avec les codes utilisés en milieu industriel.

Les fichiers principaux des codes sont rassemblés dans le répertoire `matlab`, tandis que les fonctions appelées de façon interne par ces codes sont partagées en sous-répertoires :

- `B2`, `T3`, `T6` contiennent les fonctions élémentaires pour les trois différents types d'éléments mis en œuvre ;
- `all-elements` contient les fonctions de calcul communes à tous les types d'éléments (par exemple le fonction de correction locale en plasticité) ;
- `utility` contient les fonctions utilitaires (lecture, post-traitement), qui ne sont pas d'intérêt primordial pour le cours et ne sont commenté que très rapidement.
- `input` contient les fichiers de données correspondant à certains exemples, ce qui permet de lancer les analyses correspondantes. Le lecteur peut facilement, à l'aide du mailleur `GMSH`, enrichir cette bibliothèque d'analyses possibles ;
- `temp` sert pour sauver les résultats de certaines analyses.

Pour lancer un code il suffit de taper son nom sur la ligne de commande MATLAB. Pour permettre à MATLAB de trouver les fonctions nécessaires aux différents codes il faut ajouter au chemin (path) les sous-répertoires de la liste la première fois qu'on les utilise.

Les codes `linel_T`, `linel_T_fast` et `plast_T` sont assez généraux et admettent un fichier de données quelconque préparé en accord avec les conventions de l'annexe C. Les autres codes, par contre, sont développés pour traiter certains exemples spécifiques. Les modifications nécessaires pour les adapter à d'autres situations voisines sont souvent immédiates, et sont conseillées comme exercice dans les chapitres.

Pour faciliter l'interaction avec l'utilisateur, des interfaces simples ont été ajoutées. Les instructions qui y font appel sont toujours mises en évidence et séparées du reste, par exemple :

```
%--------------------------------------------------------------------------
[figmess,t1,t2]=createfigmess();
set(t1,'string','Pre phase');
%--------------------------------------------------------------------------
```

En particulier, les codes 2D lancent, à la fin de chaque analyse, un post-traitement graphique. La compréhension de ces parties n'a évidemment pas d'importance primaire pour le cours et on la laisse comme exercice pour le lecteur motivé. On signale que la fonction `interface`

gère la fenêtre graphique avec les différentes options, tandis que `post` s'occupe effectivement du post-traitement des résultats de calcul. L'outil essentiel utilisé dans ces fonctions est la fonction `patch` de MATLAB, très puissante, qui permet par exemple de tracer le maillage, ou les cartes des déplacements et des contraintes.

B.2 Liste et description des codes

B.2.1 Code sphere_linel_B2S

Le code `sphere_linel_B2S` analyse une sphère creuse en élasticité linéaire assujettie à une pression imposée en face interne et un déplacement radial imposé en face externe. Il est décrit en détail au Chapitre 1. Le problème est ramené, en vertu de la symétrie sphérique, à un problème 1D que l'on résout à l'aide de l'élément linéaire à deux nœuds « sphérique » `B2S`. Le code effectue une comparaison avec la solution exacte de ce problème.

Tous les paramètres de l'analyse sont définis dans le fichier `sphere_linel_B2S`. Le code appelle les fonctions `stiff_linel_B2S` (évaluation de la matrice de rigidité élémentaire) et `stress_linel_B2S` (évaluation de la contrainte radiale au milieu de chaque élément).

B.2.2 Codes linel_T *et* linel_T_fast

Il s'agit de deux versions différentes du même code pour analyses planes en élasticité linéaire avec éléments triangulaires linéaires T3 où quadratiques T6. Le code et les éléments sont expliqués en détail dans les chapitres 2 et 3. Ils diffèrent par la procédure d'assemblage : le code `fast` privilège la vitesse et effectue l'assemblage des matrices en stockage Morse à trois vecteurs. Par exemple, en utilisant cette version, une analyse avec environ $2\,10^5$ éléments T3 prend moins de 5 minutes sur une machine Windows avec processeur Xeon 3.2GHz et 2Gb mémoire vive. La limite supérieure du nombre d'éléments dépends essentiellement de la plateforme utilisée (mémoire vive disponible, paramètres du système) et n'est pas imposée *a priori*. Le symbole # prenant les valeurs 3 ou 6 selon le type d'élément utilisé, les fonctions utilisées sont `read_input` (lecture des fichiers de données), `stiff_linel_T#` (création de la matrice de rigidité élémentaire), `nf_tractions_T#` (évaluation des forces nodales élémentaires dues aux efforts de surface appliquées), `stressG_linel_T#` (calcul des contraintes aux points de Gauss – pour T3 les contraintes sont donc constantes sur chaque élément) et `G2N_T#` (extrapolation des contraintes aux nœuds).

B.2.3 Codes pre_fract_T *et* post_fract_T

Les codes `pre_fract_T` et `post_fract_T` effectuent des pré- et post-traitements pour les applications en mécanique de la rupture linéaire du chapitre 4. Il ont été développés pour l'exemple spécifique de l'analyse d'une plaque rectangulaire fissurée, afin d'évaluer numériquement les facteurs d'intensité des contraintes K_{I}. Ils doivent être utilisés avec le code d'élasticité (la version rapide `linel_T_fast` est conseillée dans ce cas) et les fichiers de données `fract_ros.*`

Le post-traitement utilise le champ de déplacement estimé par `linel_T_fast` pour extrapoler K_I, soit de manière directe à partir du saut de déplacement sur la fissure, soit en utilisant une technique énergétique. Cette dernière option est développée complètement pour les seuls éléments triangulaires linéaires T3, pour lesquels la fonction élémentaire utilisée est `gtheta_T3`. Dans le cas des éléments quadratiques T6 on a la possibilité d'introduire des éléments spéciaux autour de la pointe de fissure de droite grâce au pre-processeur `pre_fract_T`, à lancer dans les codes d'élasticité avant d'effectuer la procédure d'assemblage à l'endroit indiqué dans `linel_T_fast`.

B.2.4 Code beam_ldisp_T3

Le code `beam_ldisp_T3` effectue l'analyse du flambage de la poutre doublement encastrée traitée en fin de chapitre 5. Il s'agit d'un problème en grands déplacements (donc non-linéaire) résolu à l'aide d'une procédure de Newton-Raphson. La mise en œuvre est limité aux éléments linéaires T3. La structure du code est générale, mais certains détails (comme l'évaluation de la force de compression) sont spécifiques à la structure considérée. D'autres configurations demandent des modifications minimales qui sont laissées comme exercice. Le nouvelles fonctions développées pour ce code sont : `stiff_Ldisp_T3` (évaluation de la matrice de rigidité d'un élément et la contribution de l'énergie élastique au second membre), `stressN_Ldisp_T3` (calcul des contraintes dans les éléments extrapolées aux nœuds) et `force_Ldisp_T3` (estimation de la contribution élémentaire à la force de compression).

B.2.5 Code strip_plast

Le code `strip_plast`, analysé au chapitre 6, sert comme exemple d'introduction au code `plast_T` présenté au chapitre suivant. Il traite le problème d'une barre rectangulaire en déformations planes avec déplacements normaux imposés aux extrémités droite et gauche, et autrement libre d'efforts de surface et de volume, pour laquelle l'état de déformation et de contrainte est homogène. L'algorithme de retour radial est mis en œuvre (dans la fonction `RR_VonMises`) et est combiné, de manière itérative, à une imposition simple de l'équilibre. Le code effectue une comparaison avec la solution exacte de ce problème.

B.2.6 Code plast_T

Le code `plast_T`, commenté en détail au chapitre 7, analyse des problèmes de plasticité en déformations planes en utilisant la loi de comportement de von Mises avec règle de normalité et ecrouissage isotrope linéaire. Il est basé sur la matrice tangente « cohérente ». Il accepte en entrée un fichier quelconque rédigé selon les conventions de l'annexe C, et fonctionne avec un maillage d'éléments linéaires T3 ou quadratiques T6.

Le code se base sur la structure de `linel_T` adaptée à la plasticité, et est donc restreint à des problèmes de taille modérée, l'assemblage étant non optimisé (comme pour le code élastique `linel_T`). La mise en œuvre de l'assemblage explicite et rapide (suivant le modèle de `linel_T_fast`) est laissée en exercice au lecteur. Le code `plast_T` utilise la fonction

`lcorr_T#` (évaluation de la contribution d'un élément à la matrice de rigidité tangente et au vecteur résidu), laquelle appelle la fonction `RR_VonMisesTM` (correction locale en un point de Gauss et construction de la matrice de rigidité tangente locale).

B.2.7 Code sphere_diff_B2S

La résolution numérique d'un problème de diffusion de la chaleur, analysée dans le chapitre 8, est appliquée à l'exemple d'une sphère initialement à température uniforme nulle et soumise sur sa surface à une température donnée constante T_0. Le code `sphere_diff_B2S` traite ce problème sous sa forme 1D obtenue par utilisation de la symétrie sphérique. Deux options d'intégration (explicite ou implicite) sont proposées, choisies par l'utilisateur au moyen du paramètre `implicit` dans le fichier de données. L'intégration explicite repose sur une matrice de masse condensée, tandis que l'intégration implicite emploie la matrice de masse non modifiée. Le trois fonctions principales utilisée sont `caplump_B2S` (matrice de masse élémentaire condensée), `cap_B2S` (matrice de masse élémentaire non condensée) et `cond_B2S` (matrice de conductivité élémentaire). Dans sa version explicite, le code permet l'évaluation *a priori* du pas de temps critique, que l'on compare avec l'expérimentation numérique. Enfin, le code effectue une comparaison avec la solution exacte de ce problème.

B.2.8 Code bar_dyn_B2

Le chapitre 9 utilise l'exemple classique de la propagation d'ondes de compression dans une barre cylindrique (d'axe Ox) au repos initial, encastrée à son extrémité gauche $x = 0$ et soumise à son extrémité droite $x = L$ à une force de traction constante P appliquée à partir de l'instant initial. Le problème se prête ici encore à une solution 1D, proposée dans `dyn_bar_B2` au moyen d'éléments de barre à deux nœuds. Le code met en œuvre l'algorithme de Newmark, l'utilisateur pouvant choisir librement les paramètres β, γ afin d'apprécier la stabilité de l'algorithme. Le code `dyn_bar_B2` appelle les fonctions `mass_B2` (calcul des matrices de masse élémentaires) et `blinel_stiff_B2` (calcul des matrices de rigidité élémentaires).

B.2.9 Code tire_contact_T3

Le code `tire_contact_T3` est associé à l'analyse, proposée dans la dernière partie du chapitre 9, d'un pneu qui tombe, par l'effet de sa pesanteur, sur une surface rigide horizontale. La dynamique (méthode des différences centrées) et le contact (méthode de pénalisation) sont traités de manière explicite, ce qui simplifie la mise en œuvre et facilite la convergence du problème non linéaire. Il s'agit, encore plus que pour les autres situations rencontrées, d'un code « jouet », proposé dans le seul but d'ouvrir la méthode des éléments finis vers des problèmes plus complexes (et réels) que ceux présentés dans le cours. La mise en œuvre est proposée pour cet exemple spécifique mais peut, avec des modifications limitées, être adaptée à d'autres applications.

B.3 Comment obtenir les codes

Les codes présentés dans cet ouvrage et récapitulés dans cette annexe sont librement téléchargeables et modifiables. Au moment de l'édition de cet ouvrage, ils sont proposés au téléchargement sur les sites Internet suivants :

- Editions de l'Ecole Polytechnique : `http://www.editions.polytechnique.fr/`. Chercher le catalogue, puis la fiche de présentation de l'ouvrage, à partir de laquelle les codes seront accessibles via un lien.
- Page personnelle de M. Bonnet, rubrique « enseignement » : `http://www.lms.polytechnique.fr/users/bonnet/enseignement.html`
- Page personnelle de A. Frangi : `http://www.stru.polimi.it/home/frangi/personale.html`

Il est, comme souvent dans ce domaine, difficile de garantir la pérennité de ces adresses Internet. Les lecteurs sont encouragés à entrer en contact avec les auteurs[1] s'ils rencontrent des difficultés pour obtenir ces codes.

Ces codes sont distribués en l'état. Les auteurs dégagent toute responsabilité quant à l'utilisation, quelle qu'elle soit, de résultats produits au moyen de ces codes, et ne s'engagent pas à fournir une assistance.

Enfin, rappelons que le mailleur `gmsh`, utilisé pour certains exemples et dont une version est fournie avec les codes MATLAB écrits pour cet ouvrage, est un logiciel libre développé indépendamment des auteurs. Le lecteur intéressé par des détails (ou par la dernière version du logiciel) est renvoyé au site de `gmsh` (actuellement `http://www.geuz.org/gmsh/`).

[1] `bonnet@lms.polytechnique.fr`, `attilio.frangi@polimi.it`