TP0 : Utilisation de logiciels pour le calcul scientifique et le développement

Objectif

L'objectif de ce TP est de vous permettre de découvrir des outils/logiciels de :

- Maillage : `Gmsh`, un logiciel de maillage 2D et 3D,
- Remaillage : `MMG`, logiciel de remaillage 2D et 3D,
- Visualisation : `Paraview`, un logiciel de visualisation de résultats 2D et 3D.
- Partage de fichiers : `Git` un logiciel de gestion de version pour coder facilement à plusieurs.

Ces outils (installés et utilisables sur les machines de l'école) pourront être utiles pour votre TER.

Quelques références pour aller plus loin :

- `Gmsh` http://gmsh.info/doc/texinfo/gmsh.pdf
- `MMG` http://www.mmgtools.org
- `Paraview` http://www.paraview.org/paraview-guide/
- `Git` https://services.github.com/kit/downloads/github-git-cheat-sheet.pdf

Maillage et remaillage

Exercice 1 - Découverte de `Gmsh`

`Gmsh` permet de créer un maillage à partir d'une géométrie, c'est-à-dire de créer une discrétisation de votre domaine spatial de calcul.

1. Créer un fichier cube.geo et ajouter dans ce fichier :

```
// Paramètres du maillage
meshEp = 0.1;
meshTransFinite = 10;
numLayers = 10;
// Points du carré
Point(1) = {0.0, 0.0, 0.0, meshEp};
Point(2) = {1.0, 0.0, 0.0, meshEp};
Point(3) = {1.0, 1.0, 0.0, meshEp};
Point(4) = {0.0, 1.0, 0.0, meshEp};
// Lignes qui relient les points
Line(1) = {1, 2};
Line(2) = {2, 3};
Line(3) = {3, 4};
Line(4) = {4, 1};
// Ligne pour déterminer la surface du carré
Line Loop(5) = {1,2,3,4};
// Surface du carré
Plane Surface(1) = {5};
```

Ouvrir ce fichier avec `Gmsh`. Mailler la géométrie du carré en cliquant dans le menu sur *Mesh/2D*. Jouer avec le paramètre *meshEp* (pour prendre en compte une modification du fichier .geo : *Geometry/Reload*).

2. Afin de faire un maillage structuré avec `Gmsh`, ici une grille, ajouter les lignes suivantes dans le fichier cube.geo :

```
// Structuration du maillage
Transfinite Line {1,3}=meshTransFinite;
Transfinite Line {2,4}=meshTransFinite;
Transfinite Surface 1;
Recombine Surface 1;
```

Est-ce que le paramètre *meshEp* est encore influant ? Jouer avec le paramètre *meshTransFinite*.

3. Créer le cube en ajoutant les lignes suivantes :

```
// Géométrie 3D
Extrude {0,0,1}{Surface{1};Layers{numLayers};Recombine;}
```

Le mailler avec la commande du menu *Mesh/3D*. Que se passe t-il ? Jouer avec le paramètre *numLayers*. Pour avoir un maillage non structuré, retirer les lignes qui ont été ajoutées à la question 2 et retirer l'option *Recombine* dans la fonction *Extrude*.

4. Sauvegarder la version non structurée du maillage en allant dans *File/Save as* (ou *Export*) et choisir le format : *Medit - Inria Mesh* et nommer le fichier cube.mesh (l'extension est importante). Dans la dernière fenêtre *MESH options*, sélectionner *Save all*.

5. Ouvrir le maillage dans un éditeur de texte et en étudier la structure :

```
Vertices
x y z ref
.
Triangles
v1 v2 v3 ref
.
Tetra
v1 v2 v3 v4 ref
.
```

Les références sont très importantes, en particulier pour pouvoir imposer les conditions aux bords. Vérifier que c'est bien le cas sur ce maillage.

6. Ouvrir le maillage avec `Gmsh` et tester la liste d'outils suivante :
a) *Tools/Options/Mesh/Visibility* pour afficher les points, les lignes, les surfaces, les volumes dans leur ensemble,
b) *Tools/Visibility/Tree_browse*r (ne pas oublier de développer la liste) pour afficher les entités en fonction de leurs références,
c) *Tools/Clipping* pour faire une coupe dans le maillage (jouer avec la souris pour voir les plans de coupe) puis cliquer sur *Mesh* puis *Redraw*,
d) *Tools/Statistics* pour faire des statistiques sur le maillage. Cliquer sur *Update* pour obtenir

les qualités de votre maillage. Pour chaque élément 3D (ou 2D), trois critères sont calculés :

$$\eta \text{ (ou SICN)} = \frac{V^{\frac{2}{3}}}{\sum (l_a)^2}, \quad \text{Gamma} = \frac{r_{ci}}{r_{cc}} \text{ and Rho} = \frac{\min_T l_a}{\max_T l_a},$$

où V est le volume (ou l'aire) de l'élément 3D (ou 2D), l_a correspond à la longueur d'une arête, r_{ci} le rayon de la sphère (ou du cercle) inscrit et r_{cc} le rayon de la sphère (ou du cercle) circonscrit. Ils sont normalisés pour être dans l'intervalle $[0, 1]$.
Ensuite tracer et analyser la courbe *Gamma* qui est l'indicateur le plus pertinent pour l'exercice en cliquant sur $X - Y$. Plus la qualité du maillage est bonne, moins celui-ci contient d'éléments avec un faible *Gamma*. Ces éléments sont dits distordus et ils détériorent les résultats des méthodes numériques (éléments finis/volumes finis).

Exercice 2 - Aller plus loin avec `Gmsh`

Télécharger les géométries de cet exercice en suivant ce lien : lien.

1. **Premier cas** : la géométrie *piece.brep* (BREP = Boundary REPresentation) a été obtenue par exemple par CAO (Conception Assistée par Ordinateur) et il faut la mailler. Avec le fichier .brep seulement la surface pourra être maillée. Pour mailler le volume il doit être créé. Ouvrir le fichier *piece.geo* avec `Gmsh` et éditer le pour voir comment le volume a été créé. Ensuite mailler le en 2D et en 3D. Le maillage obtenu est très grossier.
Modifier les paramètres du maillage en allant dans *Tools/Options/Mesh/General* (ne pas oublier de cliquer sur *reload* pour repartir de la géométrie d'origine et refaire TOUTE la procédure). Diminuer le paramètre *Max element size* jusqu'à être satisfait du maillage 2D obtenu, en particulier au niveau de l'hémisphère central.
Mailler ensuite en 3D. Conserver le maillage obtenu avec *Max element size = 1* sous le nom *piece_10Tet.mesh*.
Tester aussi les autres options de maillage, en particulier les deux méthodes pour mailler : Delaunay et frontale. Regarder l'impact de ces changements sur la qualité (courbe *Gamma*).

2. **Deuxième cas** : le maillage 2D *engrenage2D.mesh* est disponible et l'objectif est d'obtenir un maillage 3D.
a) Le fichier *engrenage.geo* définit 2 volumes : un engrenage et un cube autour de l'engrenage. Créer le maillage complet.
b) Il est possible de sauvegarder seulement le maillage de l'engrenage, en le définissant comme une entité physique. Rajoutant les 2 lignes suivantes dans le fichier *.geo* :

```
Physical Surface(3) = {2};
Physical Volume (3) = {2};
```

puis cliquer sur *Reload* et ensuite lors de la sauvegarde dans la dernière fenêtre *MESH options* sélectionner *Physical Entities*. C'est un des intérêts des *Physical Entities* mais il y en a d'autres (voir la documentation de `Gmsh` pour plus d'informations).

Exercice 3 - Remailler avec MMG

Les maillages générés précédemment ont des éléments de mauvaise qualité. Nous allons utiliser le remailleur MMG pour modifier localement les maillages afin d'en améliorer la qualité. MMG n'a pas d'interface graphique. Afin de le lancer facilement en ligne de commande nous allons créer un alias. Ouvrir un terminal et taper la commande suivante afin d'ouvrir le fichier *.bash_alias* (attention le copier-coller du tilde fonctionne mal) :

```
gedit ~/.bash_alias
```

Ajouter les lignes suivantes dans le fichier pour créer un alias MMG (versions 2D et 3D) :

```
alias mmg2d=/opt/mmg/5.2.5/bin/mmg2d_O3
alias mmg3d=/opt/mmg/5.2.5/bin/mmg3d_O3
```

Afin de prendre en compte ces changements dans le terminal (le faire du coup dans toutes les fenêtres qui sont ouvertes) il faut taper dans celui-ci :

```
source ~/.bash_alias
```

Pour lancer la version 3D, il faut un fichier contenant un **maillage tétraédrique** (par exemple le *cube.mesh* non structuré). Bien lire les messages d'erreur que vous avez ! Pour découvrir toutes les options, lancer la commande suivante :

```
mmg3d -h
```

Rq : si l'option *-out MyNewMesh.mesh* n'est pas utilisée le fichier maillage obtenu sera *myMesh.**o**.mesh*.

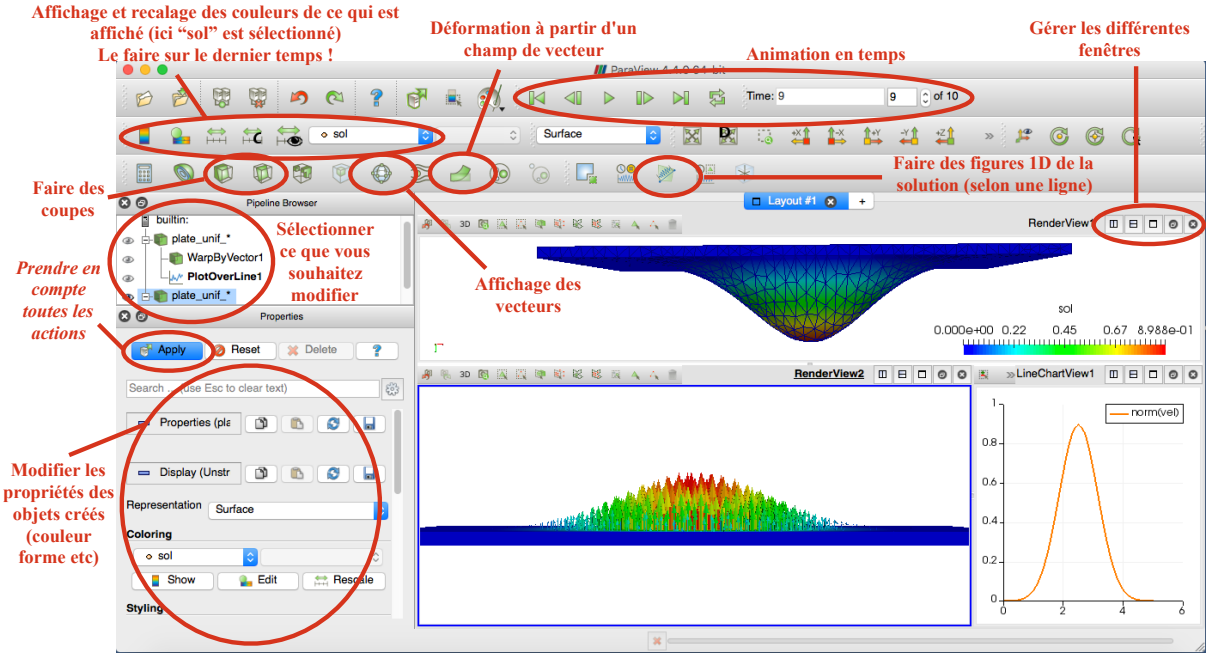
1. Quelques exemples de commande à tester sur le *cube.mesh* :

```
// Améliorer la qualité du maillage
mmg3d cube.mesh
// Améliorer la qualité du maillage en gardant le même nombre de points
mmg3d cube.mesh -noinsert
// Raffiner le maillage (tester avec différentes valeurs de hmax)
mmg3d cube.mesh -hmax 0.05
```

2. Améliorer la qualité du maillage *piece_10Tet.mesh* en gardant le même nombre de points. Comparer les courbes *Gamma* (avec Gmsh) entre le maillage d'origine et celui obtenu avec MMG. Conclure.

Exercice 4

1. Télécharger le dossier *Results* en cliquant sur le lien suivant : lien. Il contient 10 fichiers *vtk* : *plate_unif_i.vtk*, $i \in [0, 9]$ que nous supposons correspondre à la résolution d'une équation aux dérivées partielles. Il s'agit d'une séquence en temps. Pour l'ouvrir, ouvrir **d'abord** `Paraview`. Ensuite ouvrir la séquence de fichier *plate_unif_*$*$*.vtk* (File/Open). Cliquer sur *Apply*. La figure ci-dessous vous montre les commandes les plus utiles sur `Paraview`. Tester les en la reproduisant.



Remarque : Pour changer la palette de couleurs (comme sur la figure), il faut cliquer sur l'icône :

(en haut à gauche) et ensuite cliquer sur l'icône suivante dans le *Color Map Editor* :

2. Ensuite enregistrer avec *File/Save Animation*. Modifier les paramètres de sortie :

- le format pour sortir soit une série d'images, soit un film,
- *Frame Rate* pour contrôler la durée d'un film, etc ...

Le but de cette partie est de montrer sur un exemple comment utiliser ces outils. Télécharger le dossier *Code_ TP0* à l'adresse suivante : lien

Exercice 5

Créer un maillage (`Gmsh`)

1. Dans le dossier *Mesh* du dossier *Code_ TP0*, il y a un fichier *Lmesh-2D.geo* permettant d'obtenir une géométrie 2D non structurée correspondant à un maillage formant un angle. La mailler avec `Gmsh`. Sauvegarder le maillage sous le nom *Lmesh-2D.mesh* (le placer dans le dossier *Mesh*).

"Simulation numérique"

2. Comme nous n'avons pas encore implémenté de méthode numérique sur un maillage non structuré (il faudra attendre la fin du semestre !), nous allons utiliser la librairie d'éléments finis (méthode qui sera étudiée au cours de l'année) *Freefem++* (en C++).
Nous allons tout d'abord devoir reformater très légèrement le maillage pour qu'il soit compatible avec *Freefem++*. Pour cela nous allons utiliser le language C++ : il vous suffit juste dans un terminal ouvert au niveau du dossier *Mesh* de taper les 2 lignes suivantes (ne pas copier les commentaires) :

```
// Compilation du fichier main.cc
g++ -std=c++11 -o run main.cc
// Execution du fichier
./run Lmesh-2D.mesh
```

Un nouveau maillage *freefem-Lmesh-2D.mesh* est apparu dans le dossier *Mesh* (la colonne z sur les *vertices* a été supprimée pour obtenir un vrai maillage 2D compatible avec *Freefem++*).

3. À présent, nous allons résoudre l'équation aux dérivées partielles (dite équation de Poisson) suivante :

$$\begin{cases} -\Delta u &= f, \quad \Omega \\ u &= 0, \quad \partial\Omega \end{cases}$$

avec $f = 1$ en utilisant la librairie d'éléments finis *Freefem++*. Ouvrir le fichier *adaptLPoisson.edp* qui est dans le dossier *Code_ TP0*. Ce code calcule la solution de l'équation précédente sur le maillage donné à la ligne 10 (ligne que vous pourrez modifier pour considérer un autre maillage). À partir de la ligne 42, il y a le calcul d'une métrique à partir de la hessienne de la solution (c'est la hessienne qui intervient dans le calcul de l'erreur entre la solution approchée et la solution exacte dans la méthode des éléments finis). Pour lancer le code taper la ligne suivante dans le dossier *Code_ TP0* :

```
FreeFem++ adaptLPoisson.edp
```

Visualiser la solution obtenue avec `Paraview`

4. Un dossier solution *Results-freefem-Lmesh-2D* a été créé : vous pouvez ouvrir avec Paraview le fichier *u.vtk* qui vous permet d'afficher la solution (il faut afficher *scalaire1*).

Améliorer la solution en adaptant le maillage (MMG)

5. Un fichier *freefem-Lmesh-2D.sol* a été créé dans le dossier *Mesh* : il contient la métrique. La solution obtenue est non uniforme. Un maillage adapté, c'est-à-dire contenant des éléments de différentes tailles, avec :

- des éléments de taille plus petite là où sont les grandes valeurs de la hessienne,
- et de plus grandes tailles là où sont les petites valeurs de la hessienne,

améliorerait la convergence de la méthode numérique sans trop augmenter les temps de calcul (contrairement à l'utilisation d'un maillage uniforme raffiné).

Le logiciel `MMG` permet de créer un maillage non uniforme en donnant en entrée un fichier contenant la taille des arêtes voulues aux points du maillage ce qui est le cas du fichier *freefem-Lmesh-2D.sol*. Construire un nouveau maillage avec `MMG` en tapant dans le dossier *Mesh* :

```
mmg2d freefem-Lmesh-2D.mesh -sol freefem-Lmesh-2D.sol -out freefem-Lmesh-2D.mesh
```

Visualiser ce nouveau maillage. Remarque : `Gmsh` ne sait pas lire du vrai 2D (c'est à dire sans la coordonnée z). Comme il prend la référence des points pour la coordonnées z, il faut regarder le maillage du dessus pour une visualisation raisonnable. Que pensez-vous de ce nouveau maillage ? Vous pouvez calculer et afficher la solution sur ce nouveau maillage (modifier la ligne du 10 du fichier *adaptLPoisson.edp*). Est-ce nécessaire de refaire la procédure ?

Remarque finale : Comme précisé dans le titre, ce TP est seulement une introduction à ces différents logiciels. N'hésitez pas à lire les documentations pour en savoir plus. Pour finir, une introduction à un logiciel de gestion de versions est proposée.

Logiciel de gestion de version

Cette dernière section n'est pas un exercice. Il s'agit d'une présentation d'un logiciel de gestion de versions, en anglais *Versionning*. Nous vous conseillons fortement de l'utiliser pour le code de votre TER et aussi pour la rédaction de votre rapport.

Si vous avez déjà travaillé sur un projet informatique (ou sur un rapport) avec plusieurs personnes, vous avez certainement déjà rencontré un de ces problèmes :

- Quelqu'un a modifié un fichier qui fonctionnait bien et qui maintenant a des bugs,
- Quelqu'un a modifié un fichier dans la même période de temps que vous,
- Vous ne comprenez pas à quoi correspondent des nouvelles lignes ou des nouveaux fichiers,
- Vous avez fait une modification. Depuis plus rien ne fonctionne mais vous ne savez pas où est le problème.

L'utilisation d'un logiciel de gestion de version permet d'éviter tous ces problèmes : il permet de gérer les modifications de chaque membre du groupe. Un historique des changements et un retour à une ancienne version plus stable sont aussi proposés. Il existe plusieurs logiciels de gestion de versions : SVN, Mercurial ou Git. Dans ce tutoriel, nous parlerons de Git.
Ensuite il faut un hébergeur sur lequel déposer les sources. Nous vous proposons d'utiliser Bitbucket puisque l'hébergement d'un dépôt privé est gratuit jusqu'à 5 utilisateurs.
Vous pouvez même demander une licence universitaire (illimitée) avec votre adresse mail *Enseirb-Matmeca* en suivant ce lien (attention il faut écrire le mail en anglais) :
https://www.atlassian.com/software/views/bitbucket-academic-license.jsp.

1. **Installer Git sur votre ordinateur**
Regarder si Git est déjà installé. Sinon, suivre ce lien pour l'installer :
https://git-scm.com/book/en/v2/Getting-Started-Installing-Git.

2. **Créer un compte sur Bitbucket**
Créer un compte en cliquant sur : *Get Started For Free* sur la page : https://bitbucket.org. Valider votre inscription (l'arrivée du mail peut prendre du temps) puis se connecter à Bitbucket.

2 bis. **Enregistrer une clé id_rsa sur Bitbucket**
Sur les machines de l'école, il faut générer une clé id_rsa en tapant juste "ssh-keygen" dans un terminal (et appuyer sur Entrée 3 fois). Il faut ensuite déposer la clé publique : "id_rsa.pub" qui est soit dans .ssh soit .ssh/id_user sur l'interface administration du compte bitbucket (Paramètres/ClésSSH). Sur votre propre machine, il faut vérifier qu'une clé publique n'existe pas déjà avant d'en générer une. La procédure de génération de clé est différente sous mac et sous windows.

3. **Premier utilisateur : *Create a new repository*[1] (créer un dépôt)**
La première version de votre code se trouve dans un dossier qui s'appelle dans cet exemple *MonCode*. Cliquer sur *Create a repository* et choisir un nom au dépôt (dans cet exemple *DepotDeMonCode*). Ensuite choisir *I am starting from scratch* (Je commence de zéro). Bitbucket vous explique comment faire. Voici une explication détaillée :

```
cd MonCode // Aller dans le dossier où se trouve votre code
git init // Dire que ce dossier est un dépôt Git
// Préciser le dépôt considéré
git remote add origin https://MonLogin@bitbucket.org/MonLogin/DepotDeMonCode.git
```

1. repository (en) = dépôt (fr) : endroit où sont stockés les fichiers

```
// Ajouter les fichiers que vous souhaitez voir apparaitre sur le dépôt
git add file_1 file_2 file_3
// Ajouter les modifications à l'historique des modifications
// Mettre un commentaire qui décrit à quoi correspondent les changements
git commit -m "Mon premier commit" file_1 file_2 file_3
git push -u origin master // Envoyer vos fichiers sur le serveur
```

Cliquer sur l'icône *Bitbucket* tout en haut à gauche. Cliquer sur le dépôt *DepotDeMonCode* que vous venez de créer. Sur la colonne de gauche, cliquer sur *Source* et les fichiers qui ont été envoyés doivent apparaitre. Tester les autres icônes de la colonne de gauche.
Maintenant ajouter des collaborateurs en cliquant sur une fenêtre sur la droite proposant d'inviter des utilisateurs dans l'onglet *Overview*. Cliquer sur *Send Invitation* et ajouter vos collaborateurs. Décider du statut que vous leur donnez (lecture, écriture ou administrateur). Les collaborateurs doivent accepter l'invitation reçue par mail.

4. **Autres utilisateurs : *Checkout a repository* (récupérer un dépôt)**
Une fois l'invitation acceptée, les autres utilisateurs se connectent à leur compte Bitbucket. Le nouveau dépôt *DepotDeMonCode* est visible en cliquant sur l'icône *Bitbucket* tout en haut à gauche. Cliquer sur ce dépôt.
Pour récupérer le code, il faut utiliser la commande *git clone*. Pour récupérer la commande cliquer dans la colonne de gauche sur les trois petits points et cliquer sur cloner. Copier la commande. La procédure complète est détaillée ci dessous :

```
// Aller dans le dossier où vous souhaitez mettre le dossier dépôt
cd TousMesCodes
// Cloner ce dossier
git clone git@bitbucket.org:LoginPremierUtilisateur/DepotDeMonCode.git
// Vérifier que le dossier DepotDeMonCode et son contenu sont bien apparus
```

5. **Interaction entre les utilisateurs** À présent, tous les utilisateurs ont la même version du dépôt sur leur ordinateur. Une liste des commandes les plus importantes est présentée ci-dessous. D'autres commandes existent et il ne faut pas hésiter à lire la documentation. En particulier, ici nous ne considérons qu'une seule branche du projet mais il est possible d'en faire plusieurs, ce qui permet aux utilisateurs de développer des parties de code chacun de leur côté. Pour en savoir plus sur l'utilisation de plusieurs branches, suivre ce lien : http://nvie.com/posts/a-successful-git-branching-model/.

a) Avant de commencer à modifier le dépôt, ne pas oublier de se mettre à jour afin de prendre en compte les changements des autres utilisateurs :

```
git pull
```

b) Une fois que votre code est modifié, vous pouvez voir les changements que vous avez fait avec :

```
git status
```

Cette commande vous permet de voir les nouveaux fichiers que vous avez créés et aussi les fichiers que vous avez modifiés.

c) Pour ajouter les nouveaux fichiers *new_file_1* et *new_file_2* :

```
git add new_file_1 new_file_2
```

d) Pour regarder les modifications que vous avez fait dans un fichier *file* :

```
git diff file
```

e) Si jamais vous avez fait une erreur ou modifié très légèrement un fichier (ajout d'un espace, d'une ligne vide), vous pouvez remplacer les changements locaux par le fichier d'origine afin d'éviter les conflits avec :

```
git checkout file
```

Vérifier avec la commande *git status* que ce fichier n'apparait plus dans vos fichiers modifiés. Les autres fichiers sont conservés. Si vous souhaitez par contre abandonner tous vos changements locaux (attention, ceci n'est pas réversible) :

```
git fetch origin
git reset -hard origin/master
```

f) Pour *commiter* les nouveaux (après le *git add*) et anciens fichiers que vous avez modifié :

```
git commit -m "Message sur les changements" new_file_1 new_file_2 old_file_1
```

g) Pour envoyer la nouvelle version et la rendre accessible à tous :

```
git push origin master
```