TP0 : Utilisation de logiciels pour le calcul scientifique

Le **calcul scientifique** est une discipline regroupant un ensemble de champs mathématiques et informatiques permettant la **simulation numérique** de phénomènes issus de la physique, de la chimie et de la biologie. Cette simulation numérique fournit un outil efficace de **prédiction**, de **compréhension**, d'**optimisation**, voire de **contrôle** du comportement de systèmes physiques, chimiques ou biologiques.

Voici les grandes étapes du calcul scientifique :

1. **Modélisation** du phénomène étudié avec mise en équations sous forme d'Équations Différentielles Ordinaires (EDO) ou d'Équations aux Dérivées Partielles (EDP),
2. **Analyse mathématique** de ces équations (existence et unicité d'une solution) permettant de vérifier que le problème est bien posé,
3. (Si pas de solution explicite), **Choix de l'approximation numérique** de la solution : choix du schéma en temps (Euler ; Runge Kutta 4 ...) et du schéma en espace (Différences finies ; Volumes finis ; Éléments finis ...),
4. **Implémentation de l'approximation** : le choix d'approximation numérique va imposer des contraintes : discrétisation de la géométrie 2D/3D (grilles ou **maillages**), capacité de calcul et entraînent d'autres choix : utilisation de librairies existantes de ***solver*** ou implémentation propre : choix du langage de programmation ; quelle généralité ...
5. **Validation de l'approximation** : validation de code (via des solutions connues ou explicites) ; convergence de la solution ...

Objectif

L'objectif de ce TP est de vous permettre de découvrir des outils/logiciels en libre accès très utiles pour le calcul scientifique de :

- Maillage : `Gmsh`, un logiciel de maillage 2D et 3D,
- *Solver* : `FreeFem++`, une librairie d'éléments finis,
- 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
- `FreeFem++` http://www3.freefem.org
- `Paraview` http://www.paraview.org/paraview-guide/
- `Git` https://services.github.com/kit/downloads/github-git-cheat-sheet.pdf

1. **Modélisation** - Nous allons considérer l'équation de la chaleur sur un ouvert borné Ω qui est une équation aux dérivées partielles parabolique, permettant de décrire le phénomène physique de conduction thermique : soit $u(t,x,y,z)$ la température au temps t à la position (x,y,z) :
$$\partial_t u - k\,\Delta u = f, \quad \Omega,$$
où k correspond au coefficient de conductivité thermique et f au terme source (par exemple une source de chaleur localisée). Pour que le problème soit bien posé, c'est à dire qu'une solution unique existe, l'équation doit être couplée à une condition initiale :
$$u(0,x,y,z) = u_0(x,y,z), \quad \Omega,$$
et des conditions aux bords qui peuvent être variées en fonction du contexte :
   - Si isolé on considère des conditions de type Neumann homogène (sur une partie $\Gamma \subset \partial\Omega$ ou sur tout le bord $\partial\Omega$) : $\nabla u \cdot n = 0$.
   - Si non isolé (typiquement une source) on considère des conditions de type Dirichlet (sur une partie $\Gamma \subset \partial\Omega$ ou sur tout le bord $\partial\Omega$) : $u = g$.
2. **Analyse mathématique** - Il existe une unique solution $u \in C^\infty(0,T) \times H^1(\Omega)$ à l'équation de la chaleur (avec condition initiale et conditions aux bords) sous certaines hypothèses. La preuve repose sur des éléments classiques d'analyse fonctionnelle : espace de Sobolev, Théorème de trace, Inégalité de Poincaré, principe du maximum ... La base de la preuve repose sur l'écriture sous *forme variationnelle* du problème. Pour toute fonction v appelée fonction test ayant les bonnes propriétés mathématiques, à partir de l'équation de la chaleur, on a :
$$\int_\Omega \partial_t u\, v - \int_\Omega k\,\Delta u \cdot v = \int_\Omega f v,$$
En utilisant le théorème de la divergence (IPP généralisée), on obtient la formulation variationnelle de l'équation de la chaleur :
$$\int_\Omega \partial_t u\, v + \int_\Omega k\,\nabla u \cdot \nabla v - \int_{\partial\Omega} \nabla u \cdot n\, v = \int_\Omega f v,$$
3. **Choix de la discrétisation** - Pour résoudre ce système, nous pouvons par exemple utiliser la méthode des éléments finis. Dans un cas très simple (éléments dit $\mathbf{P}_1$) en supposant une discrétisation du domaine Ω en triangles et tétraèdres (maillage) et en notant U le vecteur concaténant les valeurs de la solution aux noeuds du maillage, on peut discrétiser l'équation de la chaleur par :
$$\partial_t U + kHU = F,$$
où H est une matrice et F un vecteur à partir de la formulation variationnelle. Un schéma d'Euler implicite peut ensuite être utilisé permettant de ramener le problème à la résolution d'un système linéaire :
$$(Id + \Delta t\, k\, H)U^{n+1} = \Delta t\, F - U^n.$$
4. **Implémentation de l'approximation** - Dans ce TP pour résoudre la résolution numérique décrite précédemment c'est-à-dire : créér le maillage du domaine Ω puis calculer H, F et trouver la solution du système linéaire, nous allons utiliser un logiciel de maillage `Gmsh` et un logiciel (méthode des éléments finis) `FreeFem++` qui est en libre accès. **Tout est déjà implémenté !**

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 pour en observer 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).

Solver

Exercice 3 - Découvrir une librairie d'éléments finis

Dans cet exercice, nous allons résoudre l'équation de la chaleur sur la pièce industrielle de l'exercice précédent avec la librairie d'éléments finis `FreeFem++`. Télécharger le code et les fichiers décrivant la géométrie en suivant ce lien : lien.
On suppose que suite à une contrainte mécanique appliquée dans le demi arc de cercle de la pièce pendant 12 min (0.2 h), la pièce, initialement à 0 degrés chauffe. Nous souhaitons contrôler cette augmentation de la température. Plus précisément nous souhaitons répondre à la question : **au bout de combien de temps après l'arrêt de l'utilisation de la pièce, l'augmentation de la température moyenne est inférieure à 0.1 degrès ?**

1. Plusieurs fichiers *.geo* sont disponibles dans le dossier *GeometriesAndMeshes*. Ils discrétisent tous la pièce mais avec différents choix de pas de maillage. Les maillages (les fichiers .mesh) ont été créés avec `Gmsh`. Ouvrir les 3 géométries et les 3 maillages pour bien voir leurs différences.

2. Le fichier *heat.edp* qui se trouve dans le dossier *Heat* correspond au fichier permettant de résoudre l'équation de la chaleur avec `FreeFem++`. Ouvrir le fichier et le parcourir rapidement pour comprendre les grandes étapes. Choisir le maillage en adaptant cette ligne :

```
string namemesh = "piece_h_1";
```

et choisir le pas de temps sur cette ligne :

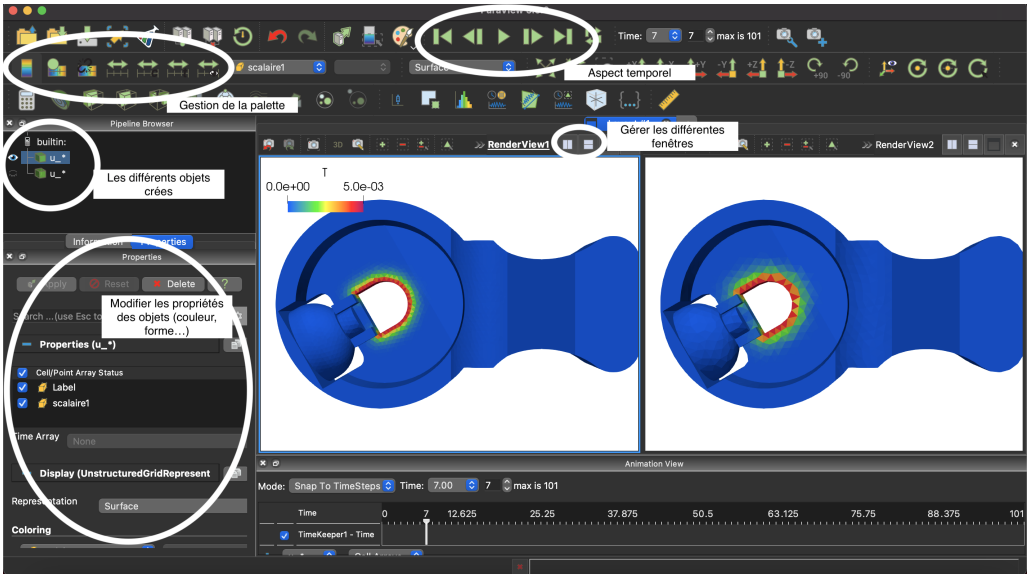
```
real dt=0.1;
```

et enfin lancer le code en ligne de commande :

```
FreeFem++ heat.edp
```

Les fichiers solution (correspondant à une séquence en temps) se trouvent dans le dossier *Results_ +nom du maillage*. Un fichier donnant la valeur moyenne de la température est aussi donné dans ce dossier.

3. On va pouvoir les ouvrir avec le logiciel `Paraview`. Ouvrir **d'abord** `Paraview`. Ensuite ouvrir la séquence de fichier *u_*.vtu* (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* :

4. 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 ...

5. Lancer les simulations associées aux 3 maillages et les comparer. Modifier le pas de temps. Faire une étude critique du choix de maillage. Il est aussi possible de comparer les moyennes de température avec gnuplot. Quel maillage et quel pas de temps vous semblent les plus adaptés **pour répondre à la question** ?

6. Il est aussi possible de construire des maillages en 2D ou en 3D avec `FreeFem++`. Dans le fichier *heatAdapt.edp*, le maillage d'une géométrie en forme de *L* est construit. Ouvrir le fichier et voir qu'à part la construction du maillage et le fait que la géométrie soit 2D, le fichier est très proche du fichier *heat.edp*. Lancer la simulation. Les fichiers créés par le code se trouvent dans le dossier *Results_no_adapt*. Ouvrir le maillage *Lshape.mesh* avec `Gmsh` et regarder les résultats sous `Paraview`.

7. Le temps de calcul est donné dans le terminal à la fin de la simulation. Ce n'est pas instantané. On peut remarquer que le maillage construit est relativement raffiné mais surtout comme quasiment tous les maillages vus dans ce TP, il n'est pas adapté localement. Dans le cas d'un problème elliptique comme l'équation de la chaleur ou les équations de Navier Stokes, on peut

montrer qu'on peut conserver une bonne approximation numérique en adaptant le maillage en ayant des mailles plus petites dans les zones où la Hessienne (dérivée seconde) de u est grande et vice et versa. La fonction *adaptmesh* de la librairie `FreeFem++` permet de faire cette adaptation en 2D (pour le cas 3D il est possible d'utiliser la librairie `MMG`). Vous pouvez tester cette fonction en modifiant la valeur du booléen *adapt* dans le fichier. Observer les 2 solutions en les comparant visuellement (avec des coupes 1D par exemple) et en temps de calcul. Conclure.

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 GitLab. Vous pouvez vous créer un compte.

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. **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 GitLab. 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
// 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
```

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

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
```