TP0 : Introduction à Matlab

Matlab est un logiciel permettant d'écrire des programmes. Il sera utilisé tout au long de votre formation, soit pour exécuter des programmes déjà écrits par d'autres, ou bien pour réaliser des prototypes de programmes. Quelle que soit l'utilisation que vous en aurez, il est important de savoir manipuler ce logiciel afin qu'il soit une aide à votre futur travail plutôt qu'un obstacle. L'objectif de ce TP est de vous familiariser avec les bases de Matlab : présentation de l'interface, écriture de lignes de code très simples, et création de vos premiers programmes.

Lors de ce TP, que vous devez lire attentivement, vous devez écrire dans Matlab toutes les lignes de code qui vous sont données afin de les tester, réaliser toutes les opérations demandées ou suggérées, et répondre à chaque question qui est posée.

1 Activer et démarrer Matlab sur les ordinateurs de l'école

Matlab existe sous Linux, Windows et Mac ; son interface graphique ainsi que les commandes que l'on peut utiliser sont les mêmes quel que soit le système d'exploitation sur lequel vous travaillez. Nous travaillerons, pendant ce TP et les prochains, exclusivement sous Linux, afin de vous familiariser avec ce système d'exploitation amplement utilisé en Informatique de Base.

Avant de pouvoir utiliser Matlab sous votre environnement Linux, vous devez "l'activer". Pour ce faire, suivez les étapes suivantes :

1. Une fois connecté à votre session Linux, cliquez sur "Activités" en haut à gauche de l'écran
2. Dans le champ "Rechercher", tapez le mot Terminal
3. Cliquez sur l'icône Terminal qui apparaît en dessous du champs de recherche
4. Dans la fenêtre qui vient de s'ouvrir, tapez "gedit .bashrc" puis validez en appuyant sur la touche Entrée.
5. Un éditeur de texte s'ouvre, et affiche le contenu du fichier *.bashrc*. A l'aide de l'éditeur de texte, recherchez le mot Matlab dans le fichier.
6. Une fois le mot trouvé, juste en dessous, vous verrez écrit

   ```
   # source /export/home/users/COMMUN/.matlabrc
   ```

   Effacez le dièse et l'espace pour avoir

   ```
   source /export/home/users/COMMUN/.matlabrc
   ```

7. Sauvegardez le fichier en cliquant sur "Enregistrer" en haut à droite de l'éditeur de texte, et fermez l'éditeur de texte.
8. Fermez le terminal.

Désormais, pour utiliser Matlab, procédez ainsi :

1. Une fois connecté à votre session Linux, cliquez sur "Activités" en haut à gauche de l'écran
2. Dans le champ "Rechercher", tapez le mot Terminal
3. Cliquez sur l'icône Terminal qui apparaît en dessous du champs de recherche
4. Dans la fenêtre qui vient de s'ouvrir, tapez

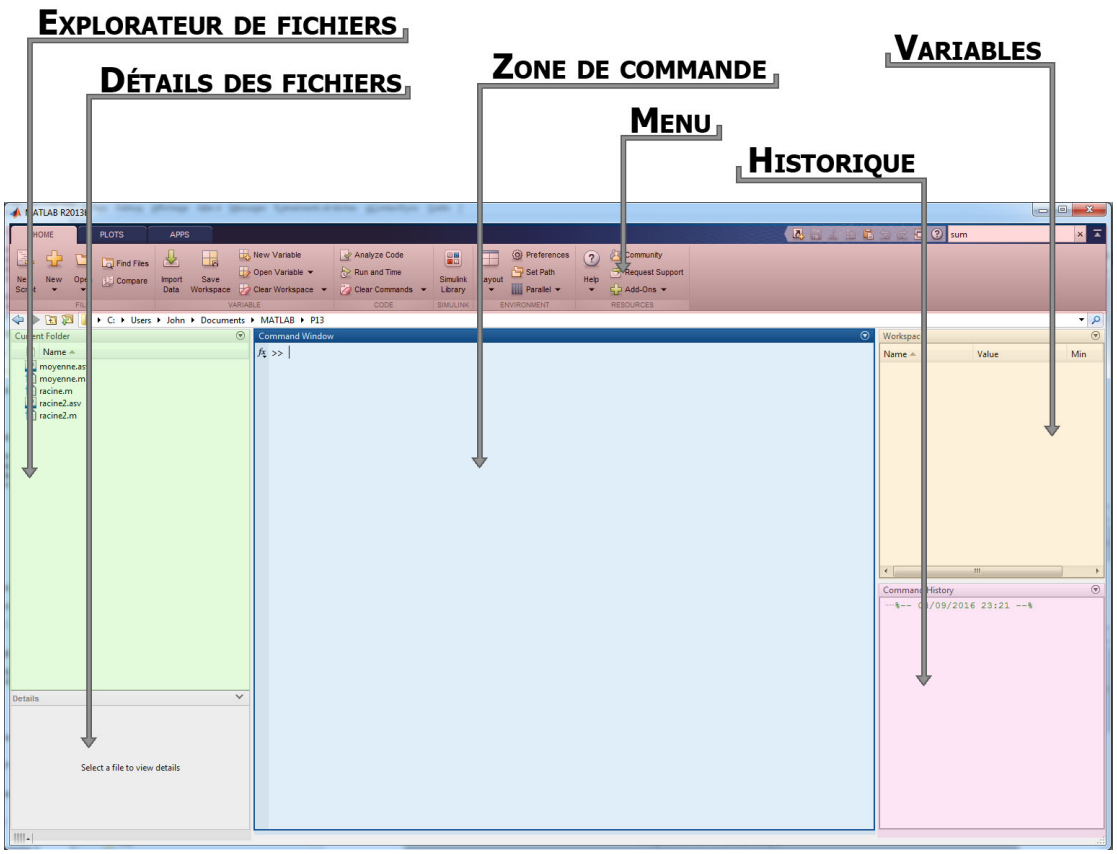
   ```
   matlab
   ```

   et Matlab démarre.

2 Prise en main de Matlab

2.1 Présentation de l'interface

L'interface de Matlab se compose de cinq zones et d'un menu situé en haut :



- Le **menu** rassemble les principales fonctionnalités de Matlab, comme enregistrer le fichier, ouvrir un fichier, voir les préférences, etc.
- L'**explorateur de fichiers** permet de se déplacer sur le disque dur pour éditer ou exécuter des programmes écrits en Matlab.
- La **zone de commande** est la zone la plus importante : elle permet d'écrire des commandes à Matlab, et de visualiser leurs résultats.
- L'**historique** permet de visualiser l'intégralité des commandes Matlab écrites depuis un certain temps, afin d'y retrouver des éléments que l'on aurait oubliés.
- La zone des **variables** rassemble toutes les variables actuellement en mémoire de Matlab, et permet de visualiser leur type, leur taille et leur contenu.
- Le **détail des fichiers** affiche des détails sur le script sélectionné. C'est la zone la moins importante.

Si par accident vous fermiez une des zones, ou modifiez l'interface de Matlab et que vous n'en étiez pas satisfait, vous pouvez revenir à la présentation par défaut. Pour ce faire, dans le menu en haut, cliquez sur le bouton **Layout**, puis sur **Default**.

2.2 Premières commandes en Matlab

Pour débuter, il est possible d'utiliser Matlab comme une calculatrice, afin de lui faire réaliser des opérations diverses ou même tracer des figures (nous verrons la partie sur les figures plus tard dans le cours). Ecrivez, dans la zone de commande de Matlab, l'opération suivante :

```
3+5
```

puis validez avec Entrée. Le résultat de l'opération s'affiche alors dans la zone de commande.

Questions

Affichez avec Matlab le résultat de $3 * 6$, et le résultat de $8/3$.

Voici une liste de quelques formules mathématiques ainsi que les commandes Matlab permettant de les réaliser.

Formule mathématiques	Commande Matlab
$\sqrt{6}$	sqrt(6)
3^9	3^9
$sin(3 * \Pi/2)$	sin(3*pi/2)
$ln(5)$	log(5)
e	exp(1)
$e^{3.5}$	exp(3.5)

Questions

1. Calculez, avec Matlab, la tangente de l'angle $4\Pi/7$.
2. Calculez la valeur de 7 à la puissance 2.8.
3. Que se passe-t-il lorsque vous appuyez, dans la zone de commande, sur la flèche du haut de votre clavier ? Et la flèche du bas ?

Notez bien que toutes les commandes que vous avez entrées sont présentes dans la zone d'historique. Il suffit de double cliquer sur l'une des commandes pour l'exécuter.

2.3 Les variables dans Matlab

Il est possible de stocker des valeurs (comme le résultat d'un calcul) dans une variable afin de pouvoir la réutiliser dans un autre calcul. Si l'on souhaite créer une variable appelée t qui vaut 2, on fera :

```
t=2
```

Le symbole = permet de recopier la valeur placée à droite dans la variable placée à gauche du symbole. Dans l'exemple précédent, on recopie donc la valeur 2 dans la variable t. **Contrairement au langage C, il n'est pas nécessaire de déclarer une variable avant de lui donner une valeur - il suffit de donner une valeur à une nouvelle variable pour que celle-ci soit automatiquement créée**. Notez que, dans la zone des variables, une nouvelle ligne s'est ajoutée : elle s'appelle t et possède comme valeur 2. La zone des variables permet de retrouver toutes les variables précédemment créées, ainsi que leur valeur. Lorsque l'on ferme Matlab, toutes les variables créées sont effacées.

Si l'on souhaite calculer la racine du logarithme de 6, on peut faire

```
x = log(6)
sqrt(x)
```

On peut aussi procéder sans aucune variable et faire

```
sqrt(log(6))
```

Questions

1. Créez une variable y qui vaudra x plus un. Une fois la bonne commande écrite et validée, vérifiez que la variable y a bien été créée dans la zone des variables et qu'elle vaut la valeur attendue.
2. Notez bien la valeur de x, puis entrez cette commande :

   ```
   x = x+1
   ```

   Que vaut maintenant la variable x ? Que s'est il passé en écrivant le morceau de code que vous avez écrit ?
3. En une seule ligne de code, multipliez la variable y par deux.

4. En une seule ligne de code, multipliez la variable y par deux sans utiliser le symbole *.

2.4 Les matrices dans Matlab

Créer des matrices Contrairement à ce que son nom pourrait laisser penser, Matlab ne signifie pas Mathematic Laboratory, mais Matrix Laboratory. Matlab est un très bon système pour faire des programmes qui font des calculs sur des matrices ou des vecteurs. Contrairement à ce que l'on peut penser aux premiers abords, beaucoup de programmes (comme les jeux vidéos, nous y reviendrons plus tard) reposent sur du calcul matriciel.

Pour stocker une matrice dans une variable, on fera :

```
m = [3 4 2.8; 4 3 2; 7 9 5; 4 1.5 3]
```

Notez dans la zone des variables qu'une nouvelle variable est apparue : Matlab ne montre pas sa valeur, mais ses dimensions, c'est à dire 4 ligne et 3 colonnes. Double cliquez sur le nom de la variable : le contenu de chaque case de la matrice apparaît alors. Fermez ensuite la zone qui vient d'apparaître afin de récupérer la mise en forme par défaut des fenêtres.

Il est possible, par la même méthode, de créer un vecteur ligne ou un vecteur colonne :

```
vec_l = [2 5 1 1 9 7]
vec_c = [3; 2; 9; 5; 1]
```

Enfin, il existe trois commandes importantes à connaître dans Matlab afin de générer des matrices :

Commande Matlab	Effet
a = rand(3,9)	Place dans a une matrice de 3 lignes et 9 colonnes, remplie de nombres aléatoires entre 0 et 1.
a = ones(4,2)	Place dans a une matrice de 4 lignes et 2 colonnes, remplie de 1.
a = zeros(3,5)	Place dans a une matrice de 3 lignes et 5 colonnes, remplie de 0.

Opération sur les matrices Les opérations usuelles sont possibles entre deux matrices de tailles compatibles. Par exemple, on peut additionner deux matrices de même taille :

```
a = rand(3,9)
b = rand(3,9)
c = a + b
```

Questions

1. Pourquoi le code ci-dessous ne fonctionne-t-il pas ?

```
a = [2 4 2; 8 6 6; 1 4 4; 5 7 3]
b = [3 2 2; 9 2 1; 2 0 8]
c = a+b
```

2. Pourquoi le code ci-dessous ne fonctionne-t-il pas ?

```
a = [2 4 2; 8 6 ; 1 ; 5 7 3]
```

3. Que fait ce code (observez bien les tailles de la matrice d) ?

```
d = a'
```

4. Que fait ce code ?

```
a = [3 2 1; 4 2 2]
b = [2 3 3]
c = [a; b]
```

5. Que fait ce code ?

```
a = [1 2 1; 3 3 2]
b = 3*a
```

Il est possible d'appeler certaines fonctions sur des matrices afin qu'elle s'exécutent sur chaque élément de la matrice. Par exemple, on peut calculer la racine de chacun des élément de la matrice a en faisant

```
c = sqrt(a)
```

Accès aux éléments des matrices Afin de lire une valeur spécifique d'une matrice, on écrit le nom de la matrice suivi, entre parenthèses, du numéro de ligne et du numéro de colonne de l'élément que l'on souhaite. Par exemple, pour lire l'élément à la seconde ligne et troisième colonne de la matrice a, on fera :

```
a = [2 4 2; 8 5 6; 1 4 4; 5 7 3]
j = a(2,3)
```

On récupère alors dans j le chiffre 6, qui est bien l'élément à la seconde ligne et troisième colonne de la matrice a. On peut aussi extraire de la matrice une sous matrice : pour ce faire, on utilise les deux points. Par exemple, pour extraire de la matrice a les ligne 2 à 3 et les colonnes 1 à 2, on fera :

```
k = a(2:3, 1:2)
```

Vous pouvez observer que k est bien une matrice de deux lignes et deux colonnes. Il est aussi possible de récupérer l'intégralité des lignes ou des colonnes en utilisant les deux points, dans mettre de nombres autour :

```
m = a(2:3, :)
```

La matrice m consiste en les lignes 2 et 3 et l'intégralité des colonnes de la matrice a. Pour récupérer la taille d'une matrice, on utilise le mot clef **size**. Par exemple, on peut faire

```
l = size(a)
```

La variable l est un vecteur ligne de deux éléments : le premier élément est le nombre de lignes de la matrice a, et le second élément est le nombre de colonnes de la matrice a. Pour connaître le nombre d'éléments total de la matrice, on utilise le mot clef **numel** :

```
n = numel(a)
```

Questions

1. Générez une matrice a de taille 4x5 avec des nombres aléatoires, et récupérez, dans une variable b, les ligne 2 à 3 et les colonnes 3 à 5 de la matrice a.
2. Récupérez, dans une variable c, le nombre de colonnes de la matrice b.
3. Sans utiliser le mot clef **numel**, récupérez dans une variable d le nombre d'éléments de la matrice b.

3 Les scripts en Matlab

3.1 Ecrire des scripts

Il est possible, dans Matlab, d'écrire les commandes dans un fichier texte (appelé script) afin de les rappeler plus tard. Pour écrire un script dans Matlab, cliquez sur le bouton **New script** en haut à gauche de la fenêtre Matlab. Dans la nouvelle fenêtre qui s'ouvre, écrivez les lignes de codes suivantes :

```
R=3;
A = 4*pi*R^2;
V = 4/3*pi*R^3;
```

Cliquez ensuite sur la petite flèche en-dessous du bouton **Save**, et cliquez sur le bouton **Save As**. Sauvegardez votre travail dans *Documents/Matlab/TP0/volume.m*. Pour exécuter votre programme, utilisez les boutons juste au-dessus de la zone explorateur de fichiers afin de vous déplacer dans le dossier *Documents/Matlab/TP0*. Ensuite, dans la fenêtre de commande, tapez

```
volume
```

Matlab exécute alors l'intégralité du fichier volume.m. Dans la zone des variables, vous verrez trois nouvelles variables apparaître : R, A et V. Ces trois variables ont bien la valeur que vous aviez imposé dans votre script. Pour rouvrir le fichier dans l'éditeur de code Matlab, il suffit de double cliquez dessus dans l'explorateur de fichiers.

Questions

1. Que se passe-t-il lors de l'exécution si vous retirez, dans le fichier *volume.m*, tous les points-virgules à la fin de chaque ligne ?
2. Ecrivez un script hypoténuse qui demande les deux côtés x et y d'un triangle rectangle à l'utilisateur, et affiche la valeur de l'hypoténuse.
   Indice : pour demander une valeur à l'utilisateur, et la stocker dans la variable x, on utilisera le mot clef ***input***

   ```
   x = input('Entrez une valeur svp : ');
   ```

   Le texte que l'on met avec la commande ***input*** *est libre de choix.*

3.2 Debugger un script Matlab pour comprendre ce qu'il fait

Pour mieux comprendre un programme, il est possible de l'exécuter ligne par ligne afin d'observer ce qu'il fait à chaque étape. Exécutez le script *racine* qui était inclus dans les fichiers joints à cet énoncé. Le programme vous demande une valeur supérieure ou égale à 0, puis affiche deux autre valeurs. En réalité, le programme calcule les deux solutions de l'équation

$$r^2 = x$$

où x est la valeur entrée par l'utilisateur.

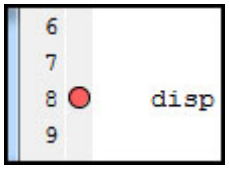
Questions

Faisons un peu de maths... Pouvez vous trouver, dans le cas général, les deux solutions (pour r) de l'équation précédente (avec $x \geq 0$) ?

Pour comprendre ce que fait ce programme, ouvrez le fichier *racine.m*. Lisez tout d'abord le code du fichier : les lignes en vert, qui commencent par le symbole pourcentage, sont en réalité des commentaires qui ne seront pas lus ni exécutés par Matlab - elles servent uniquement à aider une personne ouvrant le fichier à comprendre ce que fait le programme. On va chercher à comprendre ce que fait la ligne 8 :

```
disp(r1)
```

Cliquez juste à droite du chiffre 8 correspondant au numéro de ligne : un cercle rouge, appelé *breakpoint* (ou point d'arrêt en français), apparaît alors.



En haut de l'éditeur, cliquez sur le triangle vert **Run**. Vous revenez alors à la fenêtre de commande Matlab où le programme racine a commencé à s'exécuter. Entrez une valeur positive, puis validez.

Vous revenez alors à l'éditeur de texte, et le programme s'est mis en pause à l'endroit indiqué par une flèche verte (qui est précisément l'endroit où vous aviez placé le breakpoint) : le breakpoint permet de mettre en pause un programme à une ligne spécifique de son code. En survolant avec la souris la variable r1, une petite fenêtre affiche sa valeur. Vous pouvez aussi aller voir dans la zone des variables différents éléments afin de les examiner plus en détail.

Si vous cliquez en haut sur **Continue** (ne le faîtes pas maintenant), le programme continuera son exécution jusqu'à la fin ou jusqu'à rencontrer un autre breakpoint. Cliquez sur **Step** afin de faire avancer

d'une ligne le programme qui exécute alors la ligne 8. En revenant sur la fenêtre de commande, vous voyez que la valeur de r1 s'est affichée... Le mot clef **disp** permet donc d'afficher la valeur d'une variable.

Cette méthode, le debugging, permet d'explorer un programme récupéré d'une autre personne, et de comprendre comment il fonctionne. Cela permet aussi, et surtout, de trouver pourquoi un programme ne fonctionne pas et en éliminer les bugs.

Pour arrêter le debugging, cliquez dans l'éditeur de code sur **Quit debugging**. Retirez le breakpoint que vous aviez placé en cliquant de nouveau dessus.

Questions

1. Placez un breakpoint avant le mot clef **if** (à la ligne 11), et exécutez votre programme en donnant d'abord une valeur strictement positive pour x, puis une valeur nulle. Que fait le mot clef **if** ? Est-ce utile dans ce programme ?
2. Ouvrez le fichier *racine2.m*, et utilisez la même technique du debugging afin de comprendre ce que fait le programme, en cliquant sur le bouton **Step** pour avancer dans le programme ligne par ligne. Que se passe-t-il si l'utilisateur entre une valeur négative ? A quoi sert la variable **negatif** ? A quoi sert le **if** de la ligne 14 ?

3.3 S'exercer à faire un programme en s'inspirant d'un autre programme

Pour conclure ce TP, vous devrez lire un programme qui calcule la moyenne d'une classe et le nombre de notes supérieures à 10. Vous utiliserez ce que vous aurez appris de ce fichier afin de faire un programme qui calcule la moyenne de toutes les classes et affiche le nom de élèves qui valident leur année...

Questions

1. Ouvrez le fichier *moyenne.m*, lisez le code et exécutez le script. Parmi les affichages, le premier n'est pas bon (on affiche la variable **moy**, sans aucune phrase) : comment empêcher cela ?
2. Utilisez le mode debugging de Matlab afin de comprendre ce que fait le mot clef **sum** à la ligne 6.
3. Que fait la ligne 7 du programme ?
4. Que font les lignes 14 et 15 du programme ?
5. Que font les lignes 22 et 23 du programme ?
6. En regardant la zone des variables de Matlab, la variable **nom** est-elle définie comme une matrice ? Pourquoi (qu'est-ce qui change, dans le script, dans la façon de la définir) ?
7. C'est maintenant à vous de faire votre propre programme... Ouvrez le fichier moyenne2.m : on souhaite afficher les noms des élèves qui ont une moyenne générale supérieure à 12, ainsi que le nom des matières avec une moyenne générale supérieure ou égale à 9. Attention, les moyennes des élèves doivent être pondérées par les valeurs de pondération de chaque matière.
   *Indice : Dans la fenêtre principale de Matlab, tapez **sum** dans le champ de recherche en haut à droite, afin de voir les "options" du mot clef **sum**.*