

TP0 : Introduction à Matlab/Octave

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.

Cependant, Matlab n'est pas un programme gratuit et le nombre de licences à l'université est limité. Heureusement, il existe un logiciel équivalent à Matlab, qui est gratuit bien qu'un peu moins performant : Octave. Ce programme ressemble énormément à Matlab, et adopte le même langage de programmation que Matlab. Son avantage est que vous pouvez facilement l'installer chez vous pour l'utiliser, car il est gratuit.

L'objectif de ce TP est de vous familiariser avec les bases de Matlab et d'Octave : 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 devrez écrire dans Octave 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. Vous devez noter (soit sur une feuille, soit dans un fichier) toutes les lignes de commande qui vous paraissent importantes, ainsi que les réponses aux questions posées pour faciliter vos révisions futures. Aucun rapport ne sera à remettre, dans aucun des TPs.

Il est impératif, si vous ne terminez pas un TP pendant la séance de TP dédiée, de le terminer de votre côté avant la prochaine séance de TP.

1 Organiser ses fichiers et dossiers (1mn)

Pour rester organisé tout le long de cette matière, créez un répertoire **Matlab** propre à cette matière et, dedans, créez le dossier **TP0**. A chaque nouveau TP, vous créerez un nouveau répertoire (**TP1**, **TP2**, ...) dans le répertoire **Matlab**. Sachez que cette organisation est recommandée pour toutes les matières que vous suivrez.

2 Prise en main d'Octave

2.1 Démarrer Octave (1mn)

Octave est disponible, à l'école, sous Windows et Linux. Il est cependant préférable d'utiliser Linux.

2.1.1 Sous Linux

Sous Linux, pour démarrer l'interface graphique d'Octave, vous pouvez démarrer l'application **GNU Octave** dans le menu des applications, ou démarrer un terminal et écrire la commande

```
octave --gui
```

L'option "-gui" après la commande octave permet de signaler que l'on souhaite démarrer l'interface graphique du logiciel. Sans cette option, c'est Octave en mode ligne de commande qui démarrerait : tout s'écrit alors dans le terminal.

2.1.2 Sous Windows

Sous Windows, pour démarrer l'interface graphique d'Octave, cliquez sur le menu démarrer et cherchez le programme **Octave (GUI)** dans le dossier **GNU Octave**. Une fois lancé pour la première fois, une fenêtre vous invitant à créer un fichier de configuration apparaît : cliquez deux fois sur *Suivant*, puis une fois sur *Terminer*.

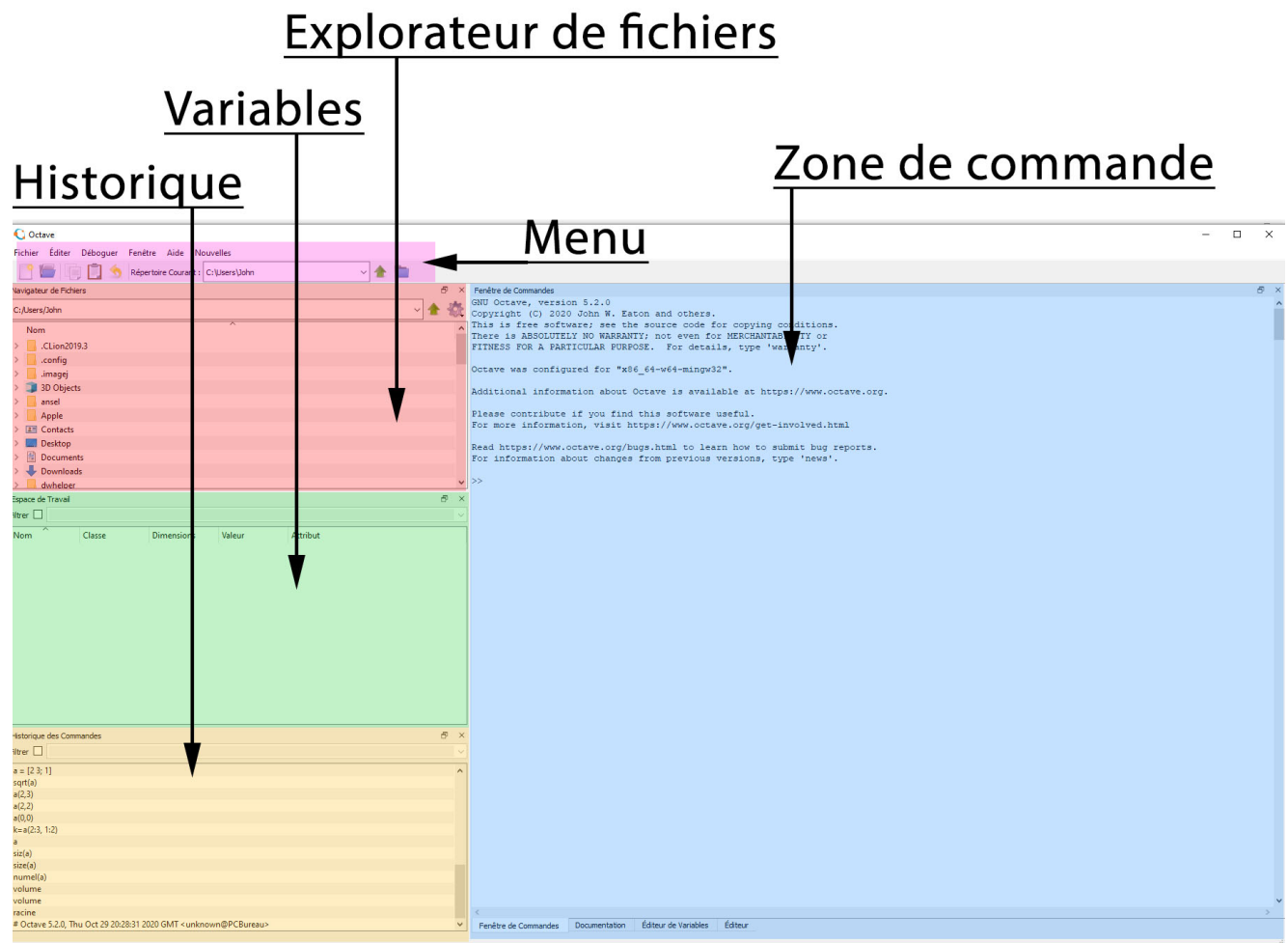


FIGURE 1 – Les différentes zones de l'interface d'Octave.

2.2 Présentation de l'interface (5mn)

L'interface d'Octave, similaire à Matlab, se compose de quatre zones ainsi que d'un menu situé en haut (voir Figure ??) :

- Le **menu** rassemble les principales fonctionnalités d'Octave, 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 Matlab/Octave.
- La **zone de commande** est la zone la plus importante : elle permet d'écrire des commandes à Octave, et de visualiser leurs résultats. Notez qu'en bas de cette zone, plusieurs onglets sont présents, permettant de lire la documentation des fonctions d'Octave, d'explorer le contenu d'une variable (nous y reviendrons plus tard) ou d'écrire des scripts (nous y reviendrons plus tard aussi).
- L'**historique** permet de visualiser l'intégralité des commandes Octave 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 d'Octave, et permet de visualiser leur type, leur taille et leur contenu.

Si, par accident, vous fermez une des zones, vous pourrez la ré-afficher en allant dans le menu *Fenêtre*, puis en sélectionnant l'une des options *Afficher xxx*. Si vous modifiez l'interface d'Octave et que vous n'en étiez pas satisfait, vous pourriez revenir à la présentation par défaut en allant dans le menu *Fenêtre* puis *Rétablir la disposition...*

2.3 Premières commandes en Octave (5mn)

Pour débiter, il est possible d'utiliser Octave 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 d'Octave, 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 Octave le résultat de $3 * 6$, et le résultat de $8/3$.

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

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

Questions

1. Calculez, avec Octave, 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.

2.4 Les variables dans Octave (5mn)

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
```

Tout comme en C, le symbole = (attribution de valeur) 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 Octave, 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 x qui sera égale à $\sqrt{\log(6)}$, ainsi qu'une variable y qui vaudra x plus un. Une fois la bonne commande écrite et validée, vérifiez que les deux variables ont bien été créées dans la zone des variables et qu'elles valent la valeur attendue.
2. En une seule ligne de code, multipliez la variable y par deux.

2.5 Les matrices dans Octave (25mn)

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, et Octave est tout aussi performant. Contrairement à ce que l'on peut penser aux premiers abords, beaucoup de solutions à différents problèmes concrets (conduite automatique, prédiction d'achat de consommateurs, ...) 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 : Octave 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 à la place de la zone de commande. Cliquez alors, en bas de la zone de commande, sur *Fenêtre de Commande* pour retrouver la zone de commande.

Pour créer une matrice dans Matlab (et Octave), il faut donc écrire des valeurs entre crochets '[' et ']'. On écrit les valeurs ligne par ligne, en séparant les valeurs d'une même ligne par un espace, et en séparant les différentes lignes par un point virgule.

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

```
vecteur_ligne = [2 5 1 1 9 7]
vecteur_colonne = [3; 2; 9; 5; 1]
```

Assurez-vous que les variables *vecteur_ligne* et *vecteur_colonne* ont bien été créées et possèdent la valeur attendue.

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

Commande Octave/Matlab	Effet
<code>a = rand(3,9)</code>	Place dans <code>a</code> une matrice de 3 lignes et 9 colonnes, remplie de nombres choisis aléatoirement entre 0 et 1.
<code>a = ones(4,2)</code>	Place dans <code>a</code> une matrice de 4 lignes et 2 colonnes, remplie de 1.
<code>a = zeros(3,5)</code>	Place dans <code>a</code> 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 éléments de la matrice a en faisant

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

Accès aux éléments des matrices Afin de lire un élément 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. Notez que, en Matlab, la première ligne et colonne ont pour indice 1, alors qu'en C, le premier élément d'un tableau a pour indice 0.

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, sans mettre aucun nombre 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.

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? Déduisez-en le rôle des points-virgules dans Octave (et

Matlab) : sont-ils obligatoires comme en C, sont-ils souhaitables ?

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. Regardez, à l'aide de l'onglet *Documentation*, l'aide de la fonction **input** permettant de demander des valeurs à l'utilisateur qui exécute le programme.

3.2 Debugger un script Octave pour comprendre ce qu'il fait (25mn)

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 autres 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 Octave - 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 quelque part sur la ligne 8, puis cliquez sur le bouton *Ajouter/Retirer un point d'arrêt* (voir ??) : un cercle rouge, appelé *breakpoint* (ou point d'arrêt en français), apparaît alors. Ce point d'arrêt forcera le script Octave à se mettre en pause lorsque la ligne 8 sera atteinte, nous permettant alors d'explorer l'état des variables du script ou bien de faire avancer le script ligne par ligne pour y corriger un éventuel bug.






Icone	Description
	Démarrer le mode déboguage
	Ajouter un point d'arrêt
	Avancer d'un pas dans le programme
	Continuer le programme (jusqu'à la fin ou jusqu'au prochain point d'arrêt)
	Quitter le mode déboguage

FIGURE 3 – Les principales commandes du mode déboguage

En haut de l'éditeur, cliquez sur le bouton *Enregistrer et exécuter*. Le script se démarre alors, et vous pouvez voir, dans la fenêtre de commande, qu'une valeur vous est demandée (à cause de la fonction **input** de la seconde ligne. Entrez une valeur positive, puis validez.

Vous pouvez voir que, sur l'éditeur de code, le programme s'est mis en pause à l'endroit indiqué par une flèche jaune (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 regardant dans la zone des variables, vous pourrez voir les valeurs des variables *r1* et *x*.

Si vous cliquez sur le bouton **Continue** (ne le faites pas maintenant), le programme continuera son exécution jusqu'à la fin ou jusqu'à rencontrer un autre breakpoint. Cliquez trois fois sur **Avancer d'un**

pas afin de faire avancer de trois lignes le programme. Vous verrez alors que la variable *r2* a fait son apparition dans la zone des variables, et qu'il est possible d'y voir sa valeur.

Cette méthode, le débogage, permet d'exécuter ligne par ligne un programme, 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 débogage, cliquez dans l'éditeur de code sur **Quitter le mode Déboguage**. Retirez le breakpoint que vous aviez placé en cliquant dessus.

Questions

1. Placez un point d'arrêt 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 *Avancer d'un pas* 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 des 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 ce premier affichage d'avoir lieu ?
2. Utilisez le mode débogage d'Octave 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. 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 14. Attention, les moyennes des élèves doivent être pondérées par les valeurs de pondération de chaque matière.

*Indice : Regardez la documentation de la fonction **sum**.*