Cours *N°1*

Initiation à MATLAB

Objectifs :

Ce premier TP a pour but de vous familiariser avec l'environnement MATLAB.

1- Présentation du logiciel :

Le nom MATLAB provient de MATrix LABoratory. L'objectif initial était de fournir un accès simplifie aux bibliothèques de fonctions des projets LINPACK et EISPACK (dédiées au calcul matriciel et à l'algèbre linéaire).

MATLAB est devenu un langage de référence pour l'analyse et la résolution de problème scientifique. Il intègre à la fois des solutions de calcul, de visualisation et un environnement de développement.

MATLAB a de nombreux avantages par rapport aux langages de programmation traditionnels (tel que le C/C++). Il permet le développement interactif de part l'utilisation d'un langage interprété. La structure de données de base est le tableau ne nécessitant pas de dimensionnement. Il fournit de nombreuses fonctions préprogrammées regroupées en boites à outils (toolbox) pour de nombreux domaines (par ex : simpowersystems, simscape, signal processing, statistics, control theory, optimization, ...).

De plus, MATLAB dispose d'une excellente documentation.

2- Démarrer Matlab :

Lorsque vous lancez Matlab pour la première fois, l'écran ressemble à celui de la Figure 1:

Le 'bureau' MATLAB est une fenêtre contenant d'autres sous-fenêtres. Les principaux outils disponibles depuis ce bureau sont :

- COMMAND WINDOW: invite de commande permettant de taper des instructions, d'appeler des scripts, d'exécuter des fonctions Matlab.

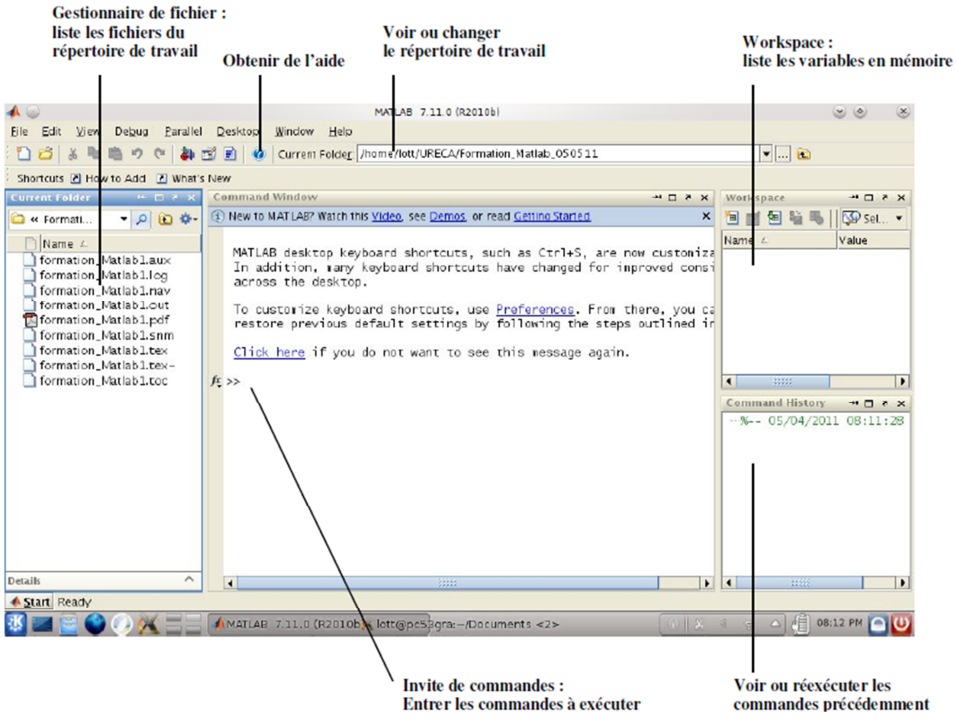
- COMMAND HISTORY : historique des commandes lancées depuis l'invite de commande.

- WORKSPACE : il liste les variables en mémoire, il permet également de parcourir graphiquement le contenu des variables.

- CURRENT DIRECTORY : un navigateur de fichier intégré a MATLAB pour visualiser le répertoire de travail courant et y effectuer les opérations classiques tel que renommer ou supprimer un fichier.

- le HELP BROWSER : un navigateur permettant de parcourir l'aide de MATLAB. L'aide est un outil précieux pour trouver les fonctions et apprendre leur fonctionnement (notamment le format des données à fournir en entrée ainsi que les valeurs renvoyées par la fonction).

Par la suite, il est conseillé de tester toutes les instructions précédées de >> dans la command window. Lorsque vous obtenez une erreur, essayez d'en comprendre la signification. Avec un peu de pratique, vous verrez que les messages d'erreur sont en général explicites.



3- Calcule élémentaire et notion de variable:

a. Matlab peut être vu comme une calculatrice extrêmement puissante. Les opérations simples peuvent être tapées directement, et l'on obtient le résultat en appuyant sur la touche "Entrée".

Essayez de faire quelques opérations dans l'interpréteur :

```
>> 5+5 (Addition)
>> 5*5 (Multiplication)
>> 5^5 (Puissance)
>> 5/5 (Division)
```

Les opérations sont évaluées en donnant la priorité aux opérateurs selon l'ordre suivant :

1. ()
2. ^
3. * /
4. + -

Essayez de faire :

```
>> 3 + 2 * 4^2

>> ((3 + 2) * 4)^2
```

b. En réalité on peut faire bien plus que des petites opérations. Et pour ce faire, on va avoir besoin de la notion de Variable : Une variable permet de mémoriser un résultat et de le réutiliser par la suite, de manière à pouvoir automatiser certaines tâches.

Afin d'affecter une variable, on utilise le signe =. Ainsi la ligne :

>> var1=3 doit être lue comme var1 reçoit 3 et non pas comme un test d'égalité.

Familiarisez-vous avec les variables :

```
>> var1 = 52
>> var1
>> var2 = 32;
>> var2
>> var1 * var2
>> Var1 = 12
>> var1
>> 11 = var1
```

c. A quoi sert le point-virgule à la fin d'une ligne de commande ? Que remarquez-vous à propos de la gestion des majuscules/minuscules dans les noms de variables.

d. Pour effacer une variable, on peut se servir de la commande clear varname, si on ne donne pas d'argument à clear, alors toutes les variables sont effacées.

Essayez par vous-même :

```
>> clear var1
>> var1
>> clear
```

4- Fonctions prédéterminées et l'aide:

a. Il existe une flopée de fonctions mathématiques comme par exemple sin, log, sqrt et bien d'autres encore. Par défaut, il y a également des variables qui sont prédéfinies comme pi ou i (nombre imaginaire pur) par exemple.

Testez les fonctions : sin(pi/2), sqrt(16), max, min, size ...

b. Vous ne connaitrez probablement jamais toutes les commandes de Matlab, mais ce n'est pas un problème, car vous pourrez retrouver toutes les informations nécessaires facilement en vous servant de l'aide. Si vous vous rappelez d'une commande mais pas de comment on l'utilise, alors la commande help commande vous sera utile.

Regardez l'aide de quelques fonctions classiques :

```
>> help log
>> help mod
```

c. Si vous n'avez pas de nom de commande mais vous savez ce que vous cherchez, vous pouvez également utiliser la fonction lookfor.

lookfor nom recherche une instruction à partir du mot clé nom

Essayez de faire :

```
>> lookfor mean
```

5- Tableaux et vecteurs:

Comme on a déjà vu, le nom "Matlab" veut dire laboratoire matriciel, et donc comme le nom l'indique, la base du logiciel sont les tableaux et les vecteurs. Un tableau permet de stocker plusieurs valeurs à la fois en pouvant accéder à chacune de manière positionnelle.

a. Par exemple sur le tableau suivant, on peut accéder indépendamment à chaque valeur.
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16

Pour déclarer ce tableau, il suffit de taper :
>> a=[1 2 3 4;5 6 7 8;9 10 11 12;13 14 15 16]

Puis pour accéder par exemple à la valeur dans la troisième ligne et dans la deuxième colonne, on tape a(3,2).

La séparation des éléments dans une ligne se fait soit par des virgules, soit par des espaces. La séparation des éléments dans une colonne se fait avec des points virgule.

b. On peut également faire des tableaux à une "dimension" appelés vecteurs :

>> b=[1 2 3 4]
>> b=[1,2,3,4]

Pour accéder au second élément par exemple on tape b(2).

c. Operations vectorielles. Les lignes suivantes résument certaines commandes couramment utilisées.

- Vecteurs :

n:m	nombres de n à m par pas de 1
n:p:m	nombres de n à m par pas de p
linspace(n,m,p)	p nombres de n à m
lenght(x)	longueur de x
x(i)	i-eme coordonnée de x
x(i1:i2)	coordonnées i1 à i2 de x
x(i1:i2)=[]	supprimer les coordonnées i1 à i2 de
x*y'	produit scalaire des vecteurs lignes x et y
max(x)	maximum
min(x)	minimum
sort(x)	tri par ordre croissant
[y, I] = sort(x)	retourne en plus les indices des éléments de x
find(x)	retourne les indices non nuls de x
sum(x)	somme des éléments de x
cumsum(x)	vecteur contenant la somme cumulée des éléments de x

prod(x)	produit des éléments de x
cumprod(x)	vecteur contenant le produit cumule des éléments de x
diff(x)	vecteur des déférences entre deux éléments consécutifs
mean(x)	moyenne des éléments de x
median(x) médiane	
std(x) ecart type	

- Matrices :

size(A)	nombre de lignes et de colonnes de A
A(i,j)	coefficient d'ordre i,j de A
A(i1:i2,:)	lignes i1 à i2 de A
A(i1:i2,:) =[]	supprimer les lignes i1 à i2 de A
A(:,j1:j2)	colonnes j1à j2 de A
A(:)	indexation linéaire de A
A(i)	coefficient d'ordre i dans l'indexation linéaire
diag(A)	coefficients diagonaux de A

- Matrices particulières :

zeros(m,n)	matrice nulle de taille m,n
ones(m,n)	matrice de taille m,n dont tous les coecients valent 1
eye(n)	matrice identité de taille n
magic(n)	carre magique de taille n
rand(m,n)	matrice de taille m,n à coefficients aléatoires de loi uniforme sur [0; 1]
randn(m,n)	matrice de taille m,n à coefficients aléatoires de loi normale N(0; 1)
inv(A)	inverse de A
expm(A)	exponentielle de A
det(A)	déterminant de A
trace(A)	trace de A
poly(A)	polynôme caractéristique de A
eig(A)	valeurs propres de A
[U,D]=eig(A)	vecteurs propres et valeurs propres de A
+ -	addition, soustraction
* ^	multiplication, puissance (matricielles)
.* .^	multiplication, puissance terme à terme

A\b solution de Ax = b
b/A solution de xA = b
./ division terme à terme

6- Ecrire des scripts M-files :

Un fichier de script est un fichier externe contenant une suite d'instruction MATLAB. Les fichiers de script ont une extension de nom de fichier .m. Les M-files peuvent être des scripts qui exécutent simplement une suite d'instructions ou peuvent être des fonctions (nous verrons les fonctions plus loin).

Exemple : créer le script "test_script" (soit vous tapez >>edit test_script.m, soit vous faites 'File'->'New'->'Script' puis 'Save As' en spécifiant "test script.m" comme nom) avec la suite d'instructions suivante :

```
clear all %efface toutes les variables du workspace
a = 1
b = 2;
c = 3, d = 4;
e = a*b/(c+d),
scal = 11;
```

Sauvegardez puis exécutez le script (menu Debug->Save&Run ou Fleche verte ou F5).

Observez la sortie sur la ligne de commande et conclure quant à l'utilisation des ; et ,

Il est également possible d'appeler le script depuis la ligne de commande, taper

>>test_script

7- Programmer avec MATLAB :

a. Operateurs de comparaisons :

Les opérateurs de comparaison sont :

== : égal à (x == y)
> : Strictement plus grand que (x > y)
< : Strictement plus petit que (x < y)
>= : plus grand ou égal à (x >= y)
<= : plus petit ou égal à (x <= y)
~= : différent de (x ~= y)

Le résultat d'une évaluation d'un test logique à l'aide d'opérateurs de comparaisons peut-être Vrai ou Faux qui sont respectivement représentés par les entiers 1 et 0 sous MATLAB (VRAI <==> 1 , FAUX <==> 0).

b. Operateurs logiques :

Il existe trois opérateurs logiques

L'operateur & (ET logique)

L'operateur || (OU logique)

L'operateur ~ (NON logique)

c. Branchement conditionnel (IF ... THEN ... ELSE ...):

On a parfois besoin d'exécuter une séquence d'instructions seulement dans le cas où une condition donnée est vérifiée au préalable.

Différentes formes d'instruction conditionnée existent sous MATLAB.

Forme 1

```
if <expression booléenne>
    <suite d'instructions exécutée si l'expression est VRAI>
End
```

Forme 2

```
if <expression booléenne >
    <suite d'instructions 1 exécutée si l'expression est VRAI>
else
    <suite d'instructions 2 exécutée si l'expression est FAUSSE>
end
```

Forme 3

```
if <expression booléenne 1>
    <suite d'instructions 1 exécutée si l'expression 1 est VRAI>
elseif <expression booléenne 2>
    <suite d'instructions 2 exécutée si l'expression 1 est FAUSSE
    et que l'expression 2 est VRAI>
.
.
.
else
    <suite d'instructions n exécutée si aucune des expressions n'est
    VRAI >
end
```

Exemple : écrire un script pile_face.m pour simuler un tirage à pile ou face

```
x = rand(); %renvoie un nombre aleatoire compris
% entre 0 et 1 selon une loi uniforme
if x > 0.5
disp('pile')
```

```
else
disp('face')
end
```

d. Boucle for

La boucle for répète une suite d'instruction un nombre prédéterminé de fois. Sa structure est la suivante :

```
for var = <list-of-values>
        <suite d'instruction>
end
```

La boucle for va exécuter la <suite d'instruction> pour chaque élément de la <list-of-values> en affectant l'élément à la variable var.

Exemple:

```
i = 0;
for k = 0:0.1:1
        i = i + 1;
        disp(['Iteration ',num2str(i),', k vaut ',num2str(k)]);
end
```

En général, la liste de valeurs sert à indexer un vecteur (ou matrice), on doit alors se limiter à des valeurs entiers.

e. Boucle while :

Il arrive que nous souhaitions répéter une suite d'instructions jusqu'a qu'une condition soit satisfaite. Si l'on ne connait pas le nombre d'itérations nécessaire à l'avance, une **boucle while** est préférable par rapport à une **boucle for.**

```
while <expression booléenne>
                <Suite d'instructions>
end
```

La <suite d'instructions> va être répétée tant que l'<expression booléenne> est vrai, ou dit autrement jusqu'à ce que l'<expression booléenne> soit fausse.

Exemple:
On cherche le premier élément négatif d'un vecteur

```
vec = [1,1,1,1,1,1,-1,1,-1]
i = 1;
        while vec(i) >= 0
                i = i+1;
        end
```

```
i %indice du premier élément négatif
vec(i) % premier élément négatif
```

8- Représentation graphique des résultats:

a. Il existe plusieurs possibilités pour représenter un ensemble de points (x(i); y(i)). Les plus utilisées sont énumérées ci-dessous :

plot(x,y,'s')	trace d'une courbe ou d'un nuage de points
bar(x,y,'s')	trace sous forme d'un histogramme
stem(x,y,'s')	diagramme en bâtons
stairs(x,y)	trace en escalier des valeurs discrètes
fplot	représente des fonctions
hist	trace des histogrammes

's' est un paramètre facultatif constitue d'une chaine de caractères qui spécifié le type de trace (couleur, différents traces en pointilles, symboles pour le trace de points).

Par défaut, le tracé est continu. Tapez help plot pour avoir la liste des valeurs possibles pour 's'.

b. Gestion de la fenêtre graphique :

hold on	les prochains tracées se superposeront aux tracées déjà effectues
hold off	le contenu de la fenêtre graphique active sera effacé lors du prochain tracé
clf	efface le contenu de la fenêtre graphique active
figure(n)	affiche ou rend active la fenêtre graphique numero n
close	ferme la fenêtre graphique active
close all	ferme toutes les fenêtres graphiques
subplot(n,m,p)	partage la fenêtre graphique active en m _ n espaces graphiques et sélectionne le p-ieme.

c. Axes et légendes :

axis([xmin xmax ymin ymax])	pour définir les échelles des axes
grid	quadrillage du graphique
grid off	
title('titre')	titre pour le graphique
xlabel('titre')	légende pour l'axe des abscisses
ylabel('titre')	légende pour l'axe des ordonnées
legend('titre1','titre2',...)	légende pour chaque courbe du graphique
text(x,y,'texte')	texte explicatif _a la position (x; y)