Université de Sfax Faculté des Sciences de Sfax ---------- *Département d'Informatique et des Communications*		**Filière LSI 1** AU : 2023-2024

Matière : Techniques Multimédia

Enseignante du Cours : Faiza CHARFI.

TP N° 1: Introduction à MATLAB

1 Introduction à Matlab

Matlab (MATrix LABoratory) est représenté par des environnements intégrés pour le Calcul Scientifique et la visualisation. Ils sont écrits principalement en langage C et C++. Il est particulièrement performant pour le calcul matriciel. Sa structure de données est basée sur les matrices, et il dispose de possibilités d'affichage très riches.

Dans ce TP, Nous présentons l'environnement de Matlab et montrons rapidement le principe de fonctionnement du logiciel. Nous présentons ensuite un ensemble de fonctions de base nécessaire pour débuter en Matlab.

En cliquant sur l'icône de Matlab, l'interface suivante apparait :

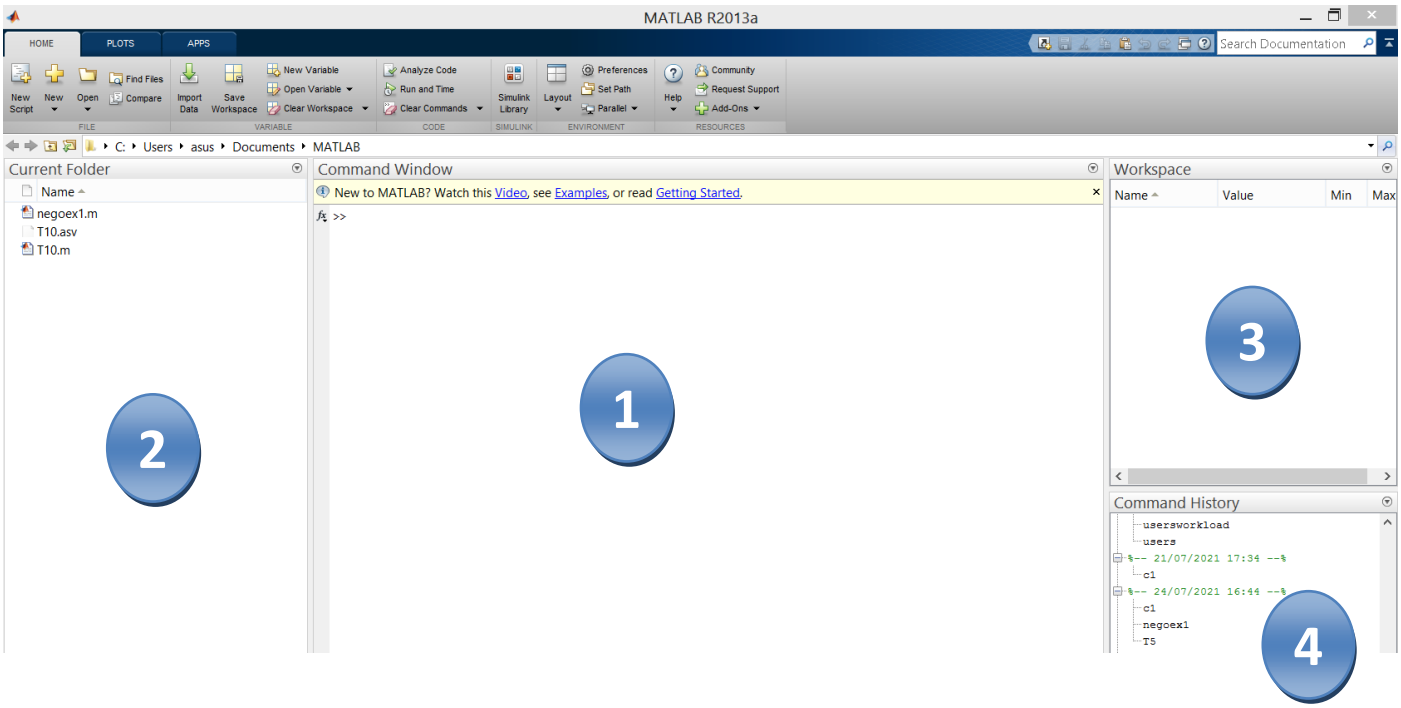

Figure 1: Présentation de l'interface Matlab.

Le logiciel propose un environnement de travail composé de multiples fenêtres. Nous pouvons distinguer quatre interfaces :

1

❖ ***Command window*** (console d'exécution) : Il s'agit de la fenêtre principale de l'interface. Il est représenté par à l'invite de commande « >> », l'utilisateur peut entrer les instructions ou les commandes à exécuter.

2

❖ ***Current Folder*** (répertoire courant) : Il permet de naviguer et de visualiser le contenu du répertoire courant de l'utilisateur. Il représente l'endroit où ils sont enregistrés les programmes (fichier .M) et les fichiers multimédia (image, signal audio, etc).

3

❖ ***Workspace*** (espace de travail) : Il permet de visualiser les variables définies leur taille occupée en mémoire et leur type.

4

- ***Command history*** : Cette interface définit l'historique des commandes que l'utilisateur a exécutées. Elle garde en mémoire ces commandes et la date et l'heure d'ouverture de chacune des sessions de Matlab.

Il existe deux modes de programmation :

- **Mode interactif :** MATLAB permet d'exécuter les instructions une par une au fur et à mesure.
- **Mode exécutif :** MATLAB exécute un programme défini dans le fichier ".m" (qui est fichier MATLAB).

Remarque:

- **Matlab est sensible à la casse.**
- Matlab définit une commande `help <fonction>` pour accéder au documentation de matlab.

```
>>  help log
```

2 Manipulation des variables

La command window est la fenêtre centrale de l'interface, c'est à partir de là que l'utilisateur pourra lancer les commandes interprétées par Matlab. Le principe est simple et intuitif, le tout est de connaitre les fonctions appropriées et de respecter leur syntaxe. Premier exemple élémentaire : à l'invite de commande, taper « 1+1 », puis entrer :

Si on veut éviter l'affichage des résultats intermédiaires, on ajoute simplement un « **;** » à la fin de la commande.

```
>>  1+1

     ans = 2
```

Si on veut éviter l'affichage des résultats intermédiaires, on ajoute simplement un « **;** » à la fin de la commande.

```
>> x = 1+7 ;
>> y= 2+4 ;
>> x + y
```

Pour avoir plus d'informations sur les variables utilisées depuis le début de la session, il suffit d'écrire.

```
>> who
```

Pour avoir plus d'informations sur le nom, l'espace mémoire utilisé, la dimension et le type de variable , il suffit d'écrire.

```
>> whos
```

Pour supprimer les valables, on utilise la commande `clear <variable>`.

```
>> clear a
>> clear all
```

2-1 Vecteurs

Matlab travaille essentiellement avec les matrices. Une variable scalaire est une matrice de dimension 1 x 1 et un vecteur est une matrice de dimension 1 x n ou n x 1. La méthode la plus simples pour définir un vecteur est de donner sa description à l'aide de la commande [ ]. Il y a deux manières de définir un vecteur:

➔ **Vecteur ligne**

```
>> v = [12 3 4]
v = 1 2 3 4
```

➔ **Vecteur colonne**

```
>> v = [1 ; 2 ; 3 ; 4]
 v = 1
     2
     3
     4
```

Cette méthode n'est pas pratique pour définir des vecteurs de taille importante. Une seconde méthode utilise l'opérateur deux-points « : ». Il permet de discrétiser un intervalle avec un pas constant.

```
>> v = 0:0.2:1

v = 0 0.2 0.4 0.6 0.8 1
```

Cette instruction crée un vecteur contenant des valeurs allant de 0 à 1 avec un pas de 0.2. La syntaxe est la suivante : vecteur = valeur_initial:incrément:valeur_finale. Par défaut, le pas est égal à 1.

```
>> v = 0:5

v = 0 1 2 3 4 5
```

2-2 Matrice

La définition d'une matrice est délimitée pas des crochets « [] ». Les différents éléments d'une ligne sont séparés par un espace et les différentes lignes sont séparées par des points virgules « ; ». Ainsi pour définir une variable matricielle

$$M = \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

```
>> M = [1 2 3 ; 4 5 6 ; 7 8 9]; // ou
>> M = [1,2,3 ; 4,5,6 ; 7,8,9];
```

On peut aussi créer automatiquement des matrices avec des fonctions prédéfinies.

- ➔ `zeros(m,n)` : La fonction produit une matrice `m×n` où tous les éléments sont initialisés à `0`.
- ➔ `ones(m,n)` : La fonction produit une matrice `m×n` où tous les éléments sont initialisés à 1.
- ➔ `eye(n)` : La fonction produit une matrice identité de taille `n×n`.
- ➔ `rand(m,n)` la fonction produit une matrice `m×n` dont les éléments sont choisis aléatoirement.

```
>> M1 = zeros(1,2)% matrice nulle de taille 1x2

>> M2 = ones(10)

>> M3 = 5*ones(5)

>> M4 = eye(9) % matrice identité d'ordre 9

>> M5 = rand(2,4)
```

Pour savoir des informations sur la taille d'une matrice on utilise la syntaxe suivante :

```
>> size(m)

>> nbr_lig = size(m,3) % nombre de lignes de m

>> nbr_col = size(m,1) % nombre de colonnes de m

>> nbr_ele = length(m)
```

La concaténation consiste à ajouter des lignes et/ou des colonnes à une matrice déjà existante. Le tableau rajouté ait le même nombre de lignes pour une concaténation suivant les colonnes, et le même nombre de colonnes.

```
>> M = [M ; 5 14 15]

>> M2 = zeros(3,1); %une matrice contient 3 ligne et une seule colone

>> M= [M2,M] % une colone de 0 s'ajoute au début de la matrice M
```

❖ **Extraction d'une sous-matrice.**

Pour extraire une sous-matrice d'une matrice M on utilise l'opérateur suivant "**:**".

➔ M(:,j) extrait la j éme colonne de M. On considère successivement toutes les lignes de M et on choisit le j éme élément de chaque ligne.

➔ M(i,:) extrait la i éme ligne de M.

➔ M(:) reforme le matrice M en un seul vecteur colonne en concaténant toutes les colonnes de M.

➔ M(j:k) extrait les éléments j à k de M et les stocke dans un vecteur ligne.

➔ M(:,j:k) extrait la sous-matrice de M formée des colonnes j à k.

➔ M(i:k,:) extrait la sous-matrice de M formée des lignes i à k.

```
>> M(1,3) %élément à la ligne 1 et à la colonne 3
>> M(1,1:3)
>> M(4,:) % la ligne 4

>> M(:,1) % la colonne 1

>> M(:)% votre remarque !

>> indi = [3, 2, 1];

>> indj = [1, 3];

>> M(indi,indj)
```

Toutes les opérations usuelles entre les matrices sont disponible telles que :l'addition, la multiplication, la transposition etc.

```
>> M1 = [1 2; 3 4];
>> M2 = [3 3; 6 6];
>> M1+3 % Addition
>> M1+M2
>> 2*M1 % Multiplication
>> M1-M2 % Soustraction
```

Il y a des fonctions de Matlab qui agissent sur chaque colonne de la matrice comme les fonctions suivantes : *max, min, sum, prod*

```
>> max(M)

>> min(M)
```

Le résultat est représenté sous forme d'un vecteur ligne dont chaque élément sera le `maximum ( ou minimum)` de la colonne correspondante.

2- 3 Chaînes de caractères

Dans le Matlab, il peut être utiliser chaînes de caractères, et cela se fait de façon tout à fait naturelle :

```
>> message = 'bienvenue sur Matlab';
```

On peut manipuler les chaînes de caractères dans les vecteurs

```
>> message = [message, '2021'];
```

2-4 Fichiers M

Il est possible d'enregistrer une séquence d'instructions dans un fichier (appelé un M-file) et de les exécuter par MATLAB.

On peut ouvrir l'éditeur du script de Matlab en allant dans le menu `File→New→M-file`. On distingue deux types suivants : les fichiers de scripts et les fichiers de fonctions.

- ➔ **Les fichiers de scripts**: Un script est un ensemble d'instructions qui joue le rôle d'un programme principal. Si le script est écrit dans le fichier (Par exemple `test.m)` on l'exécute dans la fenêtre MATLAB en tapant test ou en cliquant sur RUN (F5).
- ➔ **Les fichiers de fonctions:** Dans le Matlab, il est possible de définir des fonctions qui ne figurent pas dans la liste des fonctions incorporées de MATLAB et de les utiliser de la même manière que ces dernières. La syntaxe d'une fonction `fonc` est la suivante :

```
function [ args1 , args2 ,…] = fonc ( arge1, arge2,…)
     séquence d'instructions
end
```

Où :

• `args1,args2,...` représentent la liste des arguments de sortie.

• `arge1,arge2,...` représentent la liste des arguments d'entrée.

• `Séquence d'instructions` est le corps de la fonction.

Pour appeler une fonction, on utilise le code suivant :

```
[vars1,vars2,...] = fonc (vare1,vare2,...)
```

On peut afficher un message ou une valeur à l'écran avec l'instruction suivante :

```
>> disp('Ceci est un test')
```

On peut saisir une valeur par le clavier avec l'instruction suivante :

```
>> x = input('Valeur de x = ')
```

2-5 Structures de contrôle

Les instructions de contrôle sous MATLAB sont très proches de celles existant dans d'autres langages de programmation.

Boucle FOR : Une première possibilité pour exécuter une séquence d'instructions de manière répétée consiste à effectuer une boucle pour les valeurs d'un indice, incrémenté à chaque itération.

Syntaxe:

```
for indice=borne_inf:borne_sup
     séquence d'instructions
end
```

Exemple : calculer n!

```
>> n = 5;
>> nfac = 1;
>> for k = 1:n
     nfac = nfac*k;
   end
```

Boucle WHILE : Il consiste à effectuer une boucle tant qu'une condition reste vérifiée.

Syntaxe :

```
While expression logique
     séquence d'instructions
end
```

Exemple : calculer n!

```
>> n = 5;
>> k = 1; nfac = 1;
>> while k <= n
     nfac = nfac*k;
     k = k+1;
   end
```

L'instruction conditionnée IF : On exécute une séquence d'instructions seulement dans le cas où une condition donnée est vérifiée au préalable.

Syntaxe :

```
If exp_logique 1
     séquence d'instructions 1
elseif exp_logique 2
     séquence d'instructions 2
else
     séquence d'instructions 3
end
```

Exemple :

```
>>n = input ('Donner un nombre positif') ;
>>if mod(n,5)==0
```

```
disp ('ce nombre est divisible par 5')
 else if mod(n,12) ==0
     disp('ce nombre est divisible par 12')
else
     disp('ce nombre n''est pas divisible par 5 ou par 12')
end
end
```

Instruction switch : Il consiste à utiliser une séquence d'instructions conditionnées pour effectuer un choix en cascadee.

Syntaxe :

```
switch var

case cst1,

séquence d'instructions 1

...

Case cstN,

séquence d'instructions N

otherwise

séquence d'instructions par défaut

end
```

Ces structures font appel aux tests suivants :

==	égal à (x == y)	<=	plus petit ou égal à (x <= y)
>	strictement plus grand que (x > y)	~ =	différent de (x ~ = y)
<	strictement plus petit que (x < y)	&	et (x & y)
>=	plus grand ou égal à (x >= y)	|	ou (x | y)
		~	non (~ x)

Le résultat d'un test est un booléen qui prend la valeur 1 pour vrai et 0 pour faux.

```
>> E = 15<7
>> E = ~((17>0)|(18~=18)) (expression non (17>0 ou 18≠18)))
```

3 Graphiques sous Matlab

La fonction qui permet de tracer des courbes est *plot*. Pour tracer les valeurs d'un vecteur `M = [7 6 13 5 7 15 14]`, il suffit de taper :

```
>> plot(M)
```

Fonctions	**Description**
plot(y)	Représente chaque colonne de la matrice y en fonction de l'indice du rang.
plot(x,y) plot(x,cos(x))	Représente les colonnes de y en fonction de celles de x. il faut que x ait soit une seule colonne, soit autant de colonnes que y, et réciproquement.
plot(t1,y1,t2,y2) plot(t1,y1,':',t2,y2,'-.') plot(t1,y1,'o',t2,y2,'x')	Représente y1 en fonction de t1 et y2 en fonction de t2.

Sous la forme la plus simple, la fonction plot trace le graphe des valeurs des éléments d'un tableau en fonction des valeurs des éléments d'un autre tableau, à condition que les deux tableaux possèdent le même nombre d'éléments.

```
>> x = 0 : pi/90 : pi ;
>> y = sin(x);
>> plot(x,y) % pour afficher la courbe
>> grid % pour afficher une grille
>> xlabel('x') % pour nommer les abscisses
>> ylabel('sin(x)')
```

Il est possible d'identifier les axes par les fonctions suivantes : `xlabel(texte), ylabel(texte).` On peut titrer le graphique grâce à `title(texte)` Pour tracer les courbes, plusieurs options sont offertes. L'usager peut faire le choix entre le type de traits et de point utilisé, ainsi que la couleur. On utilise la fonction `plot(X,Y,S)` ou S indique le choix d'option entre guillemets.

Couleur	Point	Trait
b bleu g vert r rouge c cyan m magenta y jaune k noir	. point o cercle x x + plus * étoile s carré d diamant < triangle (gauche)	− continu : pointillé −. alterné _ interrompu

Exemple :

```
>> plot(A,'+') % marqueur de type +
>> plot(A,'-+') % marqueur de type + ajouté à une ligne continue
```

Si on veut afficher un vecteur `A` en fonction d'un autre vecteur `X`, il suffit de taper :

```
>> X = [1 5 7 9 11 13 15]
```

```
>> plot (X, A,'+')
```

Matlab interprète le premier vecteur comme étant l'axe des abscisses et le deuxième vecteur comme étant l'axe des ordonnées. Si on veut afficher plusieurs courbes dans la même fenêtre, il faut activer le mode `hold` en tapant: `hold on` (par défaut, ce mode est à off).

```
>> A = [1 6 3 5 7 9 12];
>> B = [2 3 8 3 6 4 8];
>> hold on
>> Plot(A,'+-'), plot(B,'o-')
```

Si on veut afficher un vecteur A et un vecteur B sur la même fenêtre principale séparée en 2 sous fenêtres, il faut utiliser la commande suivante :

```
>> subplot (2,1,1), plot(A,'+-')
>> subplot (2,1,2), plot(B,'o-')
```

Exercices

Exercice 1 : Tapez la ligne de commande suivante :

```
>>help
>>help format
>>pi
>>format long
>>pi
>>format bank
>>pi
>>format short e
>>pi
>>format
>>format compact
>>pi
>>pi;
>>A = 3000
>>a = 200
>>aA = 500
>>x=2e3
```

Calculez $y=ax^2+Ax+aA$

```
A = [1, 2, 3]; % un tableau de 3 éléments (1,2,3)
A = [17:20]; % un tableau de 4 éléments (17,18,19,20)
A = zeros(10,2); % une matrice de 10 lignes et 2 colonnes
A = [1,2,3; 4,5,6]; % une matrice 2 lignes/3 colonnes (1ère ligne
=(1,2,3), 2ème ligne=(4,5,6) )
```

Exercice 2 :

1. Entrer ces valeurs dans le vecteur x ; x=[17 8 12 15 6 17 9 18 16 5 13 19]
2. Calculer la longueur N de ce vecteur ; N=length(x)
3. Calculer la somme S des éléments ; S=sum(x)