



Module: Calcul Formel sous Matlab

Chapitre 1: Introduction à MATLAB

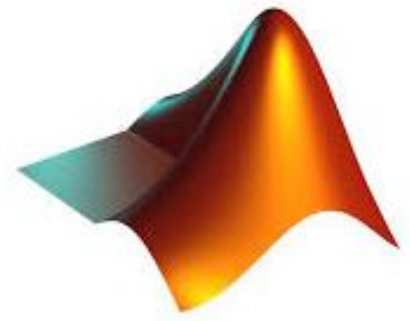
Pr : A. SLIMANI

2021/2022

Plan

- 1- Introduction**
- 2- Commandes de base en Matlab**
- 3- Vecteurs et matrices dans Matlab**
- 4- Fonction et Script**
- 5- Exercices d'application**

1. Introduction



MATLAB (**MA**Tri**x** **LA**Boratory) est un langage de programmation de haute performance très utilisé, dans le domaine de la recherche scientifique et d'industrie, destiné aux ingénieurs et aux chercheurs scientifiques pour résoudre beaucoup de problèmes techniques liés au calcul numérique et aussi la phase de développement des projets.

Il nous permet d'afficher des résultats sous forme des courbes et des données, de mettre en œuvre des algorithmes, de créer des interfaces utilisateurs, et peut s'interfacer avec d'autres langages comme le C, C++, Java, Fortran ...

MATLAB dispose de nombreuses Frameworks (**Toolboxes**) qui sont des collections complètes des fonctions MATLAB spécifiques à un domaine d'applications donné :

- **Robotiques**
- **Systèmes embarqués**
- **Télécommunications**
- **Traitement de signal et d'image**
- **Systèmes électrique et mécanique**
- **...**
- **Test et mesure** : contrôler et acquérir des données à partir de cartes d'acquisition de données enfichables, d'instruments de test, de caméras Web et de capteurs de page-écran, ainsi qu'envoyer et recevoir des messages via des bus CAN.

Il existe deux modes de fonctionnement:

1. Mode interactif :

MATLAB exécute les instructions au fur et à mesure qu'elles sont données par l'utilisateur.

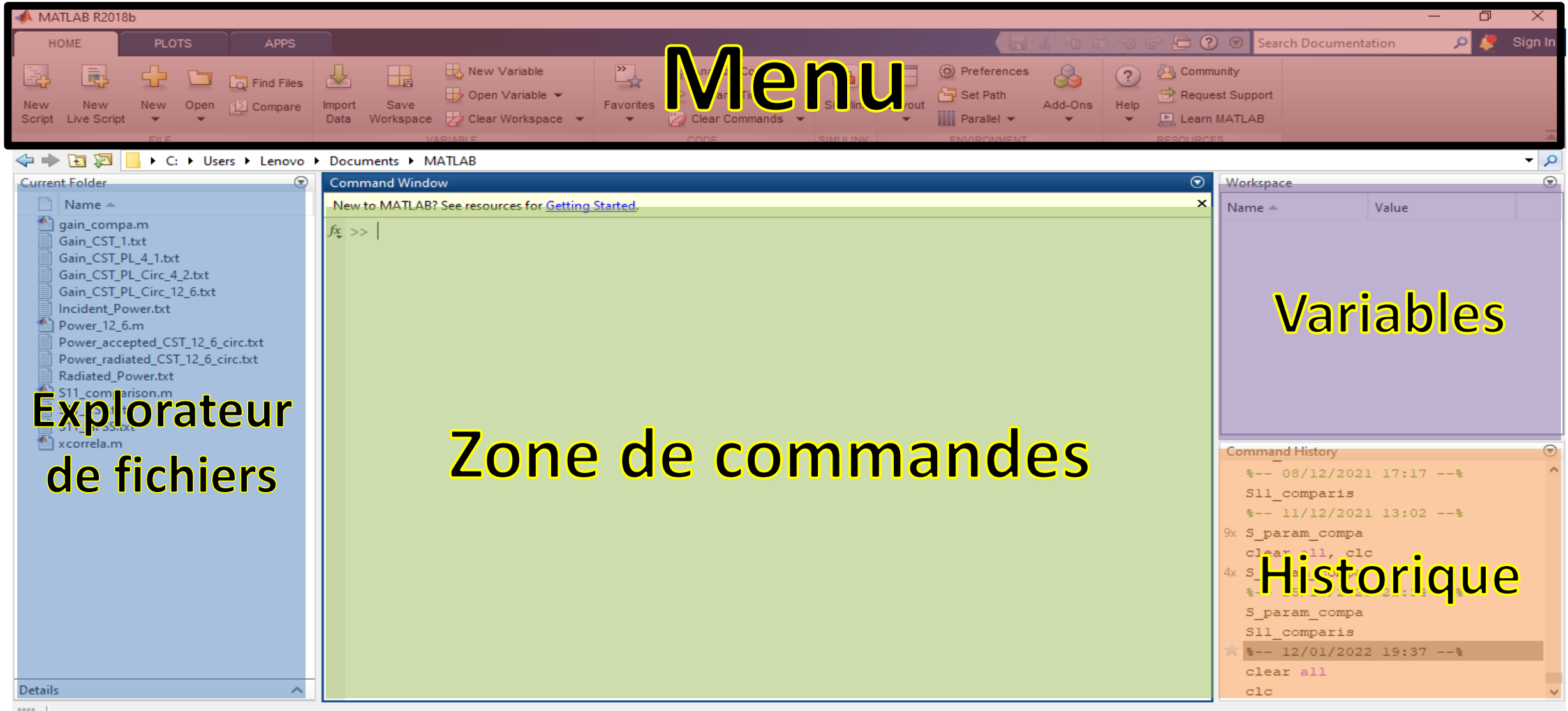
Exemple: opérations mathématiques (somme, soustraction, multiplication, ...)

2. Mode exécutif :

MATLAB exécute ligne par ligne un fichier (programme en langage MATLAB).

Exemple: créer une fonction, algorithme d'optimisation, ...

L'interface Matlab se compose de plusieurs zones :



L'apparence de la fenêtre principale peut être changée en utilisant la commande « **Layout** » de la barre de menu.

Chaque zone sur l'interface de Matlab est dédiée à des objectifs :

- Le **menu** regroupe des commandes de base de Matlab comme: créer, ouvrir, enregistrer, afficher des fichiers, ...
- La **zone de commandes** permet de rédiger des commandes et afficher leurs résultats.
- La **zone des variables** permet de visualiser toutes les variables créées en mémoire à l'instant (le nom et le contenu).
- L'**explorateur de fichiers** permet de visualiser tous les fichiers créés et les ouvrir pour les éditer.
- La **zone d'historique** permet de visualiser l'historique de tous les commandes exécutées précédemment.

Plan

- 1- Introduction
- 2- Commandes de base en Matlab**
- 3- Vecteurs et matrices dans Matlab
- 4- Fonction et Script
- 5- Exercices d'application

2. Commandes de base en Matlab

Les opérations de base:

On peut saisir des commandes dans la zone de commande (comme les opérations de: soustraction, addition, multiplication,...), et Matlab les exécute comme le ferait une calculatrice.

- Il faut savoir que Matlab est un langage **interprété** : toute commande tapée dans la zone de commande est immédiatement exécutée (après la frappe de **return**). Il n'est pas nécessaire de compiler un programme avant de l'exécuter.

□ Les Opérateurs

Le tableau suivant représente quelques opérateurs usuels utilisés par Matlab pour manipuler les expressions :

Opérateur	Définition
+ , - , * , ^ ,	Addition , Soustraction, Multiplication, Puissance,
/ , \	Division, Division à gauche
' , .'	Transposée conjuguée, Transposée
<= , >=	Inférieur ou égal, Supérieur ou égal
== , ~=	Égal, Différent
& , 	et logique, ou
~ , xor	complément logique (not), ou exclusif

❏ Les Fonctions

MATLAB fournit un nombre élevé de fonctions mathématiques élémentaires standard et avancé. Pour obtenir une liste des fonctions mathématiques **élémentaires** (**avancées**) , tapez sur la zone de commande: **help elfun** (**help specfun**).

Fonction	Définition
pi	3,141592653589793
i, j	Nombre imaginaire
exp, log, log10	Exponentielle, logarithme népérien, logarithme en base 10
sqrt , abs	Racine carrée , valeur absolue
sign	Égale à 1 si $x > 0$, 0 si $x = 0$ et -1 si $x < 0$
sin, asin, cos, tan	Sinus, sinus inverse, cosinus, tangent
sinh, asinh	Sinus hyperbolique, sinus hyperbolique inverse

Si on travaille avec des nombres complexes ($z = a + jb$), on pourra utiliser les fonctions suivantes pour trouver:

- **conj(z)** : conjugué de z
- **abs(z)** : module de z
- **angle(z)** : argument de z en radian
- **real(z)** : partie réelle de z
- **imag(z)** : partie imaginaire de z

□ Les variables

- Dans Matlab, il n'est pas nécessaire de déclarer le type (entier, réelle, complexe, ...) ou la dimension (scalaire, matrice) de la variable que l'on manipule.
- Lorsque MATLAB rencontre un nouveau nom de variable, il crée automatiquement la variable et alloue la capacité appropriée de stockage. Si la variable existe déjà, MATLAB modifie juste son contenu.
- Les noms des variables sont composés d'une lettre, ou d'une lettre suivie d'un nombre quelconque de lettres, chiffres ou de sous-tirets.
- MATLAB utilise seulement les **31** premiers caractères d'un nom de variable et établit une distinction entre majuscules et minuscules : **D** et **d** ne sont pas la même variable.

Par exemple,

Crée une variable nommée **num_machine** et stocké la valeur **14** dans le seul élément, après on a crée une variable nommée **y** et stocké la valeur **num_machine +1** dans cette variable.

Command Window

```
>> num_machine=14;  
>> y=num_machine+1;
```

Autre exemple,

assigne à la variable **A** la valeur **pi/2**.

Command Window

```
>> A=pi/2
```

Remarque :

Essayez la commande suivante :

```
Command Window

>> 3+4

ans =

    7
```

et comparez à la commande :

```
Command Window

>> 3+4;
```

ans (answer) est une variable qui contient toujours le résultat de la dernière opération exécutée,

Si vous ne spécifiez pas une variable de sortie, MATLAB utilise la variable **ans** pour stocker les résultats de cette dernière opération.

Remarque :

- Si vous mettez un point virgule « ; » à la fin d'une instruction, on peut éviter l'affichage des résultats de l'instruction.

Ceci est très utile lorsqu'on exécute des programmes très longs, car l'affichage des résultats des instructions dans la fenêtre de commande retarde l'exécution des programmes.

- On peut taper plusieurs commandes par ligne, séparées par une virgule ou un point virgule.
- On peut aussi mettre des commentaires dans une ligne de commande à l'aide du signe " % ".

Plan

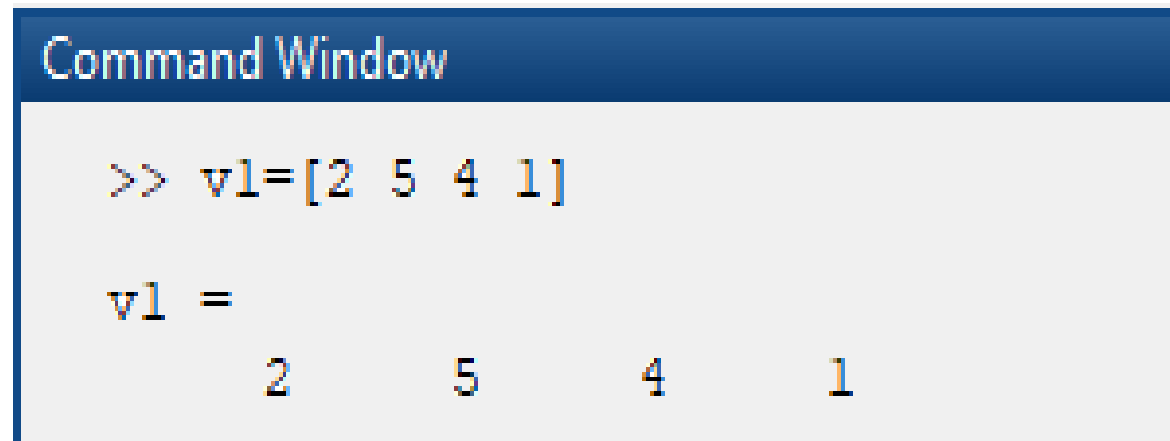
- 1- Introduction
- 2- Commandes de base en Matlab
- 3- Vecteurs et matrices dans Matlab**
- 4- Fonction et Script
- 5- Exercices d'application

3. Vecteurs et matrices dans Matlab

□ Les vecteurs

- Un vecteur sous Matlab est un regroupement de plusieurs éléments de même type.
- La méthode la plus simple pour définir un vecteur est de donner sa description explicite à l'aide de la commande `[]`, comme: `[1 2 3 4]`, `[a b c]`, ...

Exemple:



```
Command Window

>> v1=[2 5 4 1]

v1 =

     2     5     4     1
```

Vecteur colonne :

Pour définir un vecteur colonne, il suffit de mettre un point virgule entre les éléments de vecteur.

Exemple: $v1 = [2 ; 5 ; 4 ; 1]$

Command Window

```
>> v1=[2; 5; 4; 1]
```

```
v1 =
```

```
2
```

```
5
```

```
4
```

```
1
```

On peut également prendre la transposée d'un vecteur pour passer d'une ligne à une colonne ou réciproquement on ajoutant une apostrophe (') à la fin de notre vecteur.

Exemple: Soit le vecteur ligne $v1 = [2 \ 5 \ 4 \ 1]$:

$vec = v1'$ **vec** est un vecteur colonne,

Sous Matlab:

Command Window

```
>> v1=[2 5 4 1];
```

```
>> vec = v1'
```

```
vec =
```

```
2
```

```
5
```

```
4
```

```
1
```

On peut **concaténer** deux vecteurs ou plus pour créer un nouveau vecteur, par exemple, à partir de v1 et v2 on peut déduire v3:

$$v1 = [2 \ 5 \ 4 \ 1] \quad \text{et} \quad v2 = [3 \ 6 \ 9]$$

$$v3 = [v1 \ v2] \quad \longrightarrow \quad v3 = [2 \ 5 \ 4 \ 1 \ 3 \ 6 \ 9]$$

Sous Matlab:

Command Window

```
>> v1=[2 5 4 1];
```

```
>> v2=[3 6 9];
```

```
>> v3=[v1 v2]
```

```
v3 =
```

```
2      5      4      1      3      6      9
```

Il existe des vecteurs spéciaux prédéfinis dans Matlab :

ones(1, *n*) : vecteur ligne de longueur *n* dont tous les éléments valent **1**,

zeros(1, *n*) : vecteur ligne de longueur *n* dont tous les éléments valent **0**,

rand(1, *n*) : vecteur ligne de longueur *n* dont les éléments sont générés de manière aléatoire entre **0** et **1**.

ones(*m*, **1**) sera donc vecteur colonne de longueur *m* dont tous les éléments valent **1** (de même pour ***zeros*** et ***rand***).

On peut avoir des informations sur la longueur d'un vecteur à l'aide de commande **length()**.

Remarque:

- On peut utiliser la fonction **linspace** pour générer un vecteur dans Matlab.
- La commande ***linspace*(x, y, n)** génère un vecteur ligne de n éléments espacés linéairement entre x et y .

Si vous mettez ***linspace*(x, y)**, par défaut $n = 100$.

Exemple,

- Créez un vecteur nommé **V1** de 5 éléments linéairement espacés entre 3 et 10.

❑ Opérations vectorielles:

Sur la zone de commande,

- Créer les vecteurs $A = [1 \ 2 \ 3 \ 4]$, $B = [10 \ 20 \ 30 \ 40]$ et $C = [1 \ 0 \ 0 \ 0 \ 0]$.
- Calculer la somme et la soustraction des deux vecteurs A et B .
- Calculer le produit de A et B ,
- Calculer la somme et la soustraction des deux vecteurs A et C .

Qu'est ce que vous remarquez ?

Remarque:

Pour les vecteurs les opérations algébriques $+$, $-$, $*$, $/$ doivent être prises avec précautions :

- La somme et la différence sont des opérations termes à termes, et nécessitent donc des vecteurs de même dimension.
- Le produit $*$ est le produit matriciel.
- Pour utiliser la multiplication **termes à termes**, on doit remplacer $*$ par $.*$, la même chose pour la division.

On peut trouver dans Matlab des commandes qui sont propres aux vecteurs :

- ***sum*(x)** : somme des éléments du vecteur x
- ***prod*(x)** : produit des éléments du vecteur x
- ***max*(x)** : plus grand élément du vecteur x
- ***min*(x)** : plus petit élément du vecteur x
- ***mean*(x)** : moyenne des éléments du vecteur
- ***sort*(x)** : ordonne les éléments du vecteur x par ordre croissant
- ***fliplr*(x)** : renverse l'ordre des éléments du vecteur x

Remarque:

L'opérateur « : »

- On peut utiliser cet opérateur de différentes manières dans Matlab (voir help colon).
- Il ne permet de construire un vecteur dont les valeurs des éléments sont incrémentées séquentiellement : *début : pas : fin*

Tapez ces exemples sous Matlab:

- **10:2:20**
- **10:20**

Qu'est ce que vous remarquez ?

- Si on a un vecteur $V = [X_1 \ X_2 \ \dots \ X_n]$ qui se compose de n nombres, on peut relever la valeur d'un nombre juste en indiquant **sa position** dans ce vecteur.

Exemple,

$$V = [5 \ 7 \ 9 \ 3 \ 1 \ 4 \ 6 \ 8]$$

Si vous mettez dans la zone de commande $V(1)$, il va afficher 5,

Si vous mettez dans la zone de commande $V(4)$, il va afficher 3,

⋮

⋮

Exercice:

Donnez le code Matlab qui permet de :

- a. Créez un vecteur **colonne vec1** de 4 éléments linéairement espacés entre 3 et 12.
- b. Ajoutez deux lignes à la fin de ce vecteur avec la valeur 0.
- c. Ajoutez 1 au deuxième et sixième éléments de ce vecteur.
- d. Créez un second vecteur **vec2** colonne de même dimension que **vec1** contenant les entiers pairs supérieurs ou égaux à 6.
- e. Définir un vecteur **sumvec** comme la somme des deux vecteurs **vec1** et **vec2**.
- f. Définir un vecteur **prodvec** comme le produit termes à termes des deux vecteurs **vec1** et **vec2**.

❏ Les matrices

Une matrice est composée de m lignes et n colonnes. Pour créer une matrice dans MATLAB, on doit suivre les conventions de base suivantes :

- Séparer les éléments d'une ligne avec des espaces.
- Utilisez un point-virgule « ; » pour indiquer la fin de chaque ligne.
- Entourez l'intégralité de la liste des éléments avec des crochets [].

Exercice :

1. Créer dans la zone de commande MATLAB, la matrice suivante :

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

2. Utiliser la commande `[m,n] = size(M)`. Qu'est ce que vous remarquez ?
3. Utiliser la commande `numel(M)`. Qu'est ce que vous remarquez ?
4. Créer la matrice suivante : $P = [6 \ 5 \ 1; \ 2 \ 7 \ ; \ 4 \ 8 \ 9]$

Qu'est ce que vous remarquez ?



Il y a une erreur de syntaxe au niveau de la matrice P, car il n y a pas le même nombre d'éléments (colonnes) à chaque ligne.

Remarque:

- Pour extraire un élément d'une matrice M on indique la ligne et la colonne de celui-ci $M(i, j)$.
- Pour extraire une colonne ou une ligne entière on utilise le symbole $(:)$.
- La commande ***diag***(M) permet d'extraire la diagonale de la matrice, ***diag***(M, i) permet d'extraire la $i^{\text{ième}}$ sur-diagonale de la matrice, ***diag***($M, -i$) permet d'extraire la $i^{\text{ième}}$ sous-diagonale de la matrice.

Comme pour les vecteurs il est possible d'obtenir la transposée d'une matrice avec la commande « ' » (**une apostrophe à coté de la matrice**).

- On peut aussi extraire un élément de la matrice M on indique son unique numéro qui est son ordre dans la matrice: $M(\dots)$
- Le premier élément d'une matrice est celui à la 1^{ère} ligne et la 1^{ère} colonne, le seconde élément est celui à la 2^{ème} ligne et la 1^{ère} colonne etc ...

Exemple d'une matrice avec la position des ses éléments en couleur rouge:

$$M = \begin{bmatrix} \textcolor{red}{1} & 1 & \textcolor{red}{4} & 5 & \textcolor{red}{7} & 0 & \textcolor{red}{10} & 3 \\ \textcolor{red}{2} & 3 & \textcolor{red}{5} & 14 & \textcolor{red}{8} & 5 & \textcolor{red}{11} & 1 \\ \textcolor{red}{3} & 12 & \textcolor{red}{6} & 8 & \textcolor{red}{9} & 19 & \textcolor{red}{12} & 6 \end{bmatrix}$$

✓ $M(3) = 12, M(7) = 0, \dots$

Exercice:

On considère la matrice précédente :

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

1. Extraire les éléments de la matrice : $M(2,3)$; $M(3,2)$; $M(1,3)$
2. Extraire les éléments de la matrice : $M(2,:)$; $M(:,2)$; $M(3,:)$
3. Extraire sous forme de vecteurs la diagonale, le 1^{er} sur-diagonale et le 1^{er} sous-diagonale de la matrice M ,
4. Calculer la transposée T de M .

Noter qu'en Matlab, on commence la numérotation des lignes/colonnes de la matrice à 1.

❑ Opérations matricielles:

Sur la zone de commande,

- Créer les matrices suivantes: $A = \begin{bmatrix} 2 & 4 \\ 1 & 8 \end{bmatrix}$, $B = \begin{bmatrix} 3 & 1 \\ 2 & 5 \end{bmatrix}$ et $C = \begin{bmatrix} 1 & 3 & 2 \\ 4 & 0 & 2 \\ 1 & 5 & 6 \end{bmatrix}$
- Calculer la somme et la soustraction des deux matrices A et B .
- Calculer la somme et la soustraction des deux matrices A et C . *Qu'est ce que vous remarquez ?*
- Calculer le produit: $A * B$ et $A .* B$, et la division de A/B et $A./B$. *Qu'est ce que vous remarquez ?*

Remarque:

Aussi pour les matrices les opérations algébriques $+$, $-$, $*$, $/$ doivent être prises avec précautions :

- La somme et la différence sont des opérations **termes à termes**, et nécessitent donc des matrices de **même dimension**.
- Le produit $*$ est le produit matriciel.
- Pour utiliser la multiplication ou la division **termes à termes**, on doit remplacer $*$ par $.*$ et $/$ par $./$ comme le cas des vecteurs.

Comme les vecteurs, il existe des matrices prédéfinies :

***eye*(n)** : la matrice identité (carrée de taille n),
***ones*(m, n)** : matrice de m ligne et n colonnes dont tous les éléments valent **1**,
***zeros*(m, n)** : matrice de m ligne et n colonnes dont tous les éléments valent **0**,
***rand*(m, n)** : matrice de m ligne et n colonnes dont les éléments sont générés de manière aléatoire entre **0** et **1**.
***magic*(n)** : une matrice magique de dimension n (somme des éléments d'une ligne = somme des éléments d'une colonne)
***det*(A)** : calcule le déterminant de A (s'il existe),
***inv*(A)** : calcul l'inverse de la matrice A (s'il existe).

Lorsque le déterminant d'une matrice est nul, cette matrice est dite singulière et elle n'admet pas de matrice inverse.

❑ Filtrage d'une matrice:

La fonction ***find*** permet de trouver les éléments d'une matrice qui respectent certaines conditions. Cette fonction génère un vecteur colonne contenant les numéros d'ordre des éléments recherchés.

Par exemple:

$$M = \begin{bmatrix} 6 & 5 & 2 & 3 \\ 1 & 9 & 5 & 13 \\ 12 & 8 & 19 & 6 \end{bmatrix}$$

- ***find***($M < 4$): renvoie les numéros d'ordre de tous les éléments de M inférieurs à 4 :

2
7
10

On peut ensuite utiliser ces numéros d'ordre pour indexer directement la matrice M et influencer sur ses éléments.

Par exemple, on peut mettre à 0 tous les éléments de M qui sont inférieur à 5.

$$M = \begin{bmatrix} 6 & 5 & 2 & 3 \\ 1 & 9 & 5 & 13 \\ 12 & 8 & 19 & 6 \end{bmatrix}$$

```
>> z = find(M < 5)
```

```
>> M(z) = 0
```

Aussi, on peut ajouter un nombre sur les éléments d'une matrice qui respectent une condition.

Par exemple, on peut ajouter un 1 à tous les éléments de A qui sont inférieur à 1.

$$A = \begin{bmatrix} 0 & 5 & 2 & 3 \\ 4 & 0 & 5 & 0 \\ 12 & 8 & 0 & 6 \end{bmatrix}$$

```
>> k = find(A < 1)
>> A(k) = A(k) + 1
```


Exercice:

On considère la matrice suivante:

$$A = \begin{bmatrix} -1 & 2 & -2 & 3 \\ 4 & 1 & -3 & 1 \\ 2 & -3 & -1 & 1 \end{bmatrix}$$

- ☐ Réaliser les commandes suivantes: **sum(A, 1)** et **sum(A, 2)**, qu'est ce que vous remarquez ? Même pour **prod(A, 1)** et **prod(A, 2)**,

- ☐ Réaliser l'opération de la valeur absolue sur la matrice à l'aide du mot clef ***find***.

Remarque:

- ❑ **sum(A, 1)** : calcul la somme des éléments de A le long des colonnes.
- ❑ **sum(A, 2)** : calcul la somme des éléments de A le long des ligne.
- ❑ **prod(A, 1)** : calcul le produit des éléments de A le long des colonnes.
- ❑ **prod(A, 2)** : calcul le produit des éléments de A le long des lignes.

❑ Concaténation de matrices:

On peut créer une nouvelle matrice à partir d'anciennes matrices.

Par exemple:

$$A = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \end{bmatrix}, B = \begin{bmatrix} 5 & 3 & 0 \\ 1 & -1 & 0 \\ 1 & 5 & 0 \end{bmatrix} \text{ et } C = \begin{bmatrix} 0 & 1 \\ 2 & 5 \end{bmatrix}$$

On peut créer deux matrices D et E tel que:

```
>> D = [B A]
```

```
>> E = [A; C]
```

$$D = \begin{bmatrix} 5 & 3 & 0 & 1 & 4 \\ 1 & -1 & 0 & 2 & 5 \\ 1 & 5 & 0 & 3 & 6 \end{bmatrix} \quad \text{et} \quad E = \begin{bmatrix} 1 & 4 \\ 2 & 5 \\ 3 & 6 \\ 0 & 1 \\ 2 & 5 \end{bmatrix}$$

Plan

- 1- Introduction
- 2- Commandes de base en Matlab
- 3- Vecteurs et matrices dans Matlab
- 4- Fonction et Script**
- 5- Exercices d'application

4. Fonction et Script

Les programmes Matlab peuvent être des scripts ou des fonctions.

Une fonction est simplement une collection de commandes Matlab, placée dans un ***m – fichier***, qui peut recevoir des entrées et retourner des sorties.

Un script est aussi stocké dans un ***m – fichier***, qui n'accepte pas des entrées, et ne retourne aucune sortie.

□ Fonction

Une fonction qui est définie dans un m – *fichier*, commence par une ligne de la forme :

function [$y1, \dots, yn$] = *Nom_de_la_fonction* ($x1, \dots, xm$)

Ou :

- $x1, \dots, xm$ sont les variables d'entrée, qui sont nécessaires à la fonction pour accomplir ses calculs.
- $y1, \dots, yn$ sont les variables de sortie sur lesquels les résultats de la fonction sont retournés ;

Rq: Le nom du fichier enregistré doit être le même nom de la fonction.

Exemple d'application:

Par exemple, si on souhaite faire une fonction **racin2**, qui calcule et renvoie en sortie la racine de deux scalaires passés en paramètre, on écrira dans le fichier **racin2.m** :

```
function [y] = racin2(a,b)
y = sqrt(a+b);
end
```

Pour tester cette fonction, placez-vous dans le répertoire qui contient le fichier **racin2.m** et on écrira dans la fenêtre de commande (par exemple **racin2(4,5)**). Le résultat affiché est $y=3$:

```
racin2(4,5)
y=3
```

❏ Script

Le script est un *m – fichier* simple. Il s'agit simplement d'une liste de commandes mises bout à bout et sauvegardée dans un fichier.

Par exemple, si on souhaite faire un script **somme**, qui calcule et renvoie en sortie la somme de deux scalaires passés en paramètre, on écrira dans le fichier **somme.m** :

```
a=3;  
b=5;  
y = a+b
```

Pour le lancer, on écrit simplement l'instruction **somme** dans la fenêtre de commande Matlab. Le résultat affiché est $y = 8$:

```
somme  
y=  
8
```


Plan

- 1- Introduction
- 2- Commandes de base en Matlab
- 3- Vecteurs et matrices dans Matlab
- 4- Fonction et Script
- 5- Exercices d'application**

5. Exercices d'application

Exercice 1:

Soit la matrice carré $M1$ suivante:

$$M1 = \begin{bmatrix} 1 & 5 & 7 & 0 \\ 0 & 2 & 6 & 4 \\ 7 & 8 & 3 & 3 \\ 9 & 6 & 2 & 4 \end{bmatrix}$$

- A partir de $M1$, construire une matrice diagonale $M2$ ayant la même diagonale que $M1$.

Exercice 2:

On considère le système suivant:

$$\begin{cases} 2x + 3y - z = 4 \\ -x + y + 5z = 1 \\ 4x + 2y + 2z = -2 \end{cases}$$

- ☐ Donnez la forme matricielle de ce système.
- ☐ Résolvez ce système en utilisant l'inversion matricielle puis en utilisant la division à gauche.

Exercice 3:

L'équation ci-dessous représente la relation entre l'échelle de la température en Celsius et en Fahrenheit.

$$T(^{\circ}C) = aT(^{\circ}F) + b$$

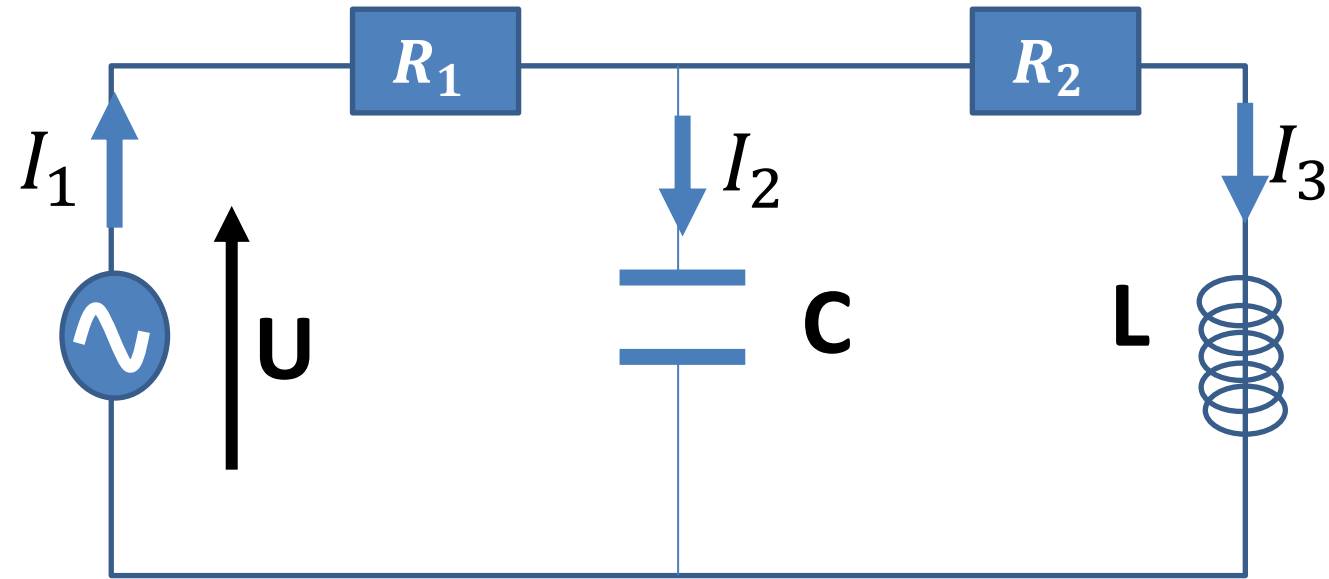
À partir de la température de fusion de l'eau, $T(^{\circ}F) = 32$ et $T(^{\circ}C) = 0$ et sa température d'ébullition $T(^{\circ}F) = 212$ et $T(^{\circ}C) = 100$, on peut déduire les coefficients a et b .

- 1) Donnez le système d'équation qui permet d'obtenir a et b .
- 2) Donnez la forme matricielle de ce système.
- 3) Résolvez ce système.

Exercice 4:

Soit le circuit suivant :

- ✓ $R_1 = 15 \, \Omega$ et $R_2 = 5 \, \Omega$
- ✓ $C = 12 \, \mu F$ et $L = 90 mH$,
- ✓ $U = 240 V$ et $f = 50 Hz$



- 1) Relever le système qui relie les courants I_1 , I_2 et I_3 (en notation complexe).
- 2) Donnez la forme matricielle de ce système.
- 3) Calculer les valeurs efficaces et phases des courants I_1 , I_2 et I_3 .

FIN