# CHAPITRE 1 : INTRODUCTION A MATLAB

Un langage de calcul scientifique est un langage de programmation destiné à être utilisé par la communauté scientifique pour réaliser des calculs scientifiques complexes. Il est riche en terme fonctions et de bibliothèques facilitant la tâche d'un programmeur dans un domaine de recherche, qui n'a pas nécessairement des compétences de programmation avancées. Les langages de calculs scientifiques sont classés en langages compilés et langages interprétés.

Dans un langage de programmation interprété un programme supplémentaire (l'interpréteur) est nécessaire, celui-ci va générer l'exécutable des instructions et les exécutent au fur et à mesure de l'exécution du programme, donc on n'a pas dans ce cas un exécutable, et à chaque fois on a besoin du code source initiale pour exécuter le programme. Des exemples de langages interprétés incluent : MATLAB, le langage R, Scilab… etc. Tandis que, un langage compilé va traduire (compilé) le programme en exécutable qui peut être utilisé ultérieurement sans avoir besoin du code source initiale. FORTRAN, C et C++ sont des exemples de langages compilés.

Ce premier chapitre sert comme introduction au langage MATLAB. Dans lequel, nous introduisons les notions indispensables aux étudiants pour utiliser ce logiciel. Nous présentons donc l'interface graphique de MATLAB, les outils de base, les vecteurs, les matrices, les entrées/ sorties et les fonctions spéciales existantes dans MATLAB.

## 1. INTRODUCTION A L'ENVIRONNEMENT MATLAB

### 1.1. Introduction

MATLAB (MATrix LABoratory) est un environnement de programmation interactif pour le calcul scientifique, la programmation et la visualisation des données. Il est très utilisé dans les domaines d'ingénierie et de recherche scientifique, ainsi qu'aux établissements d'enseignement supérieur. Sa popularité est due principalement à sa forte et simple interaction avec l'utilisateur mais aussi aux points suivants :

- Sa richesse fonctionnelle : avec MATLAB, il est possible de réaliser des manipulations mathématiques complexes en écrivant peu d'instructions. Il peut évaluer des expressions, dessiner des graphiques et exécuter des aussi du calcul programmes classiques. Et surtout, il permet l'utilisation directe de plusieurs milliers de fonctions prédéfinie.
- La possibilité d'utiliser les boites à outils (toolboxes): ce qui encourage son utilisation dans plusieurs disciplines (simulation, traitement de signal, imagerie, intelligence artificielle, ...).

- La simplicité de son langage de programmation : un programme écrit en MATLAB est plus facile à écrire et à lire comparé au même programme écrit en C ou en PASCAL.
- Sa manière de tout gérer comme étant des matrices, ce qui libère l'utilisateur de s'occuper de typage de données et ainsi de lui éviter les problèmes de transtypage.

On trouve dans la littérature des concurrents de MATLAB qui font aussi du calcul scientifique payants comme MAPLE et MATHEMATICA ou libres (freeware) comme OCTAVE et SCILAB.

Comme Windows est l'environnement le plus souvent utilisé par les débutants, nous avons décrit la version installée sous Windows. Il peut y avoir quelques modifications de détail pour les autres systèmes d'exploitation (Unix/Linux). On suppose dans ce qui suit que le logiciel est déjà installé (version 7.x).

## 1.2. Interface Graphique de MATLAB

L'interface de MATLAB peut changer légèrement selon la version utilisée, mais les points centraux restent identiques. La **Figure 1.1** montre la fenêtre principale par défaut de MATLAB (version 7.0.0) lors de son lancement. On peut également personnaliser l'affichage des fenêtres depuis le menu : Desktop→ Desktop Layout.

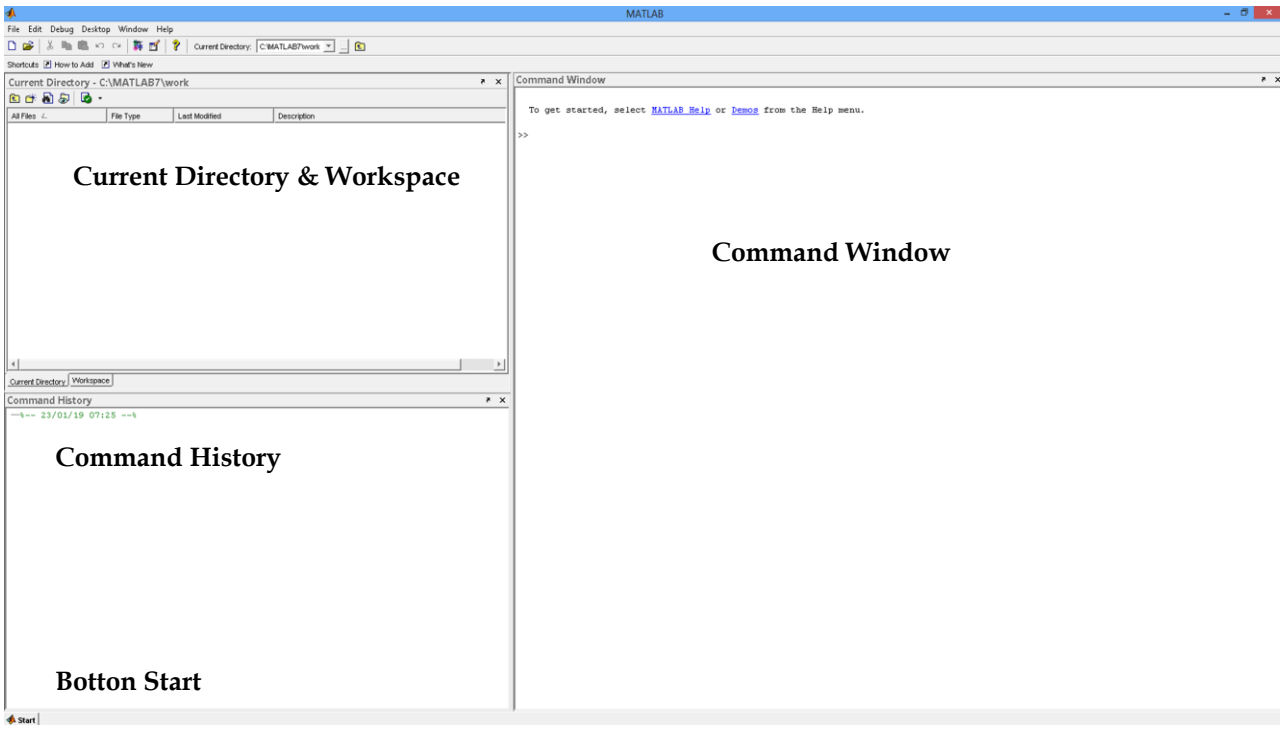


**Figure 1.1 :** Fenêtre par défaut de MATLAB

Selon le mode d'affichage par défaut (**Figure 1.1**), on dispose de quatre fenêtres :

1. **Command Window**

C'est la fenêtre dans laquelle on tape les commandes en exécution immédiate et dans laquelle sont affichées par défaut les réponses. Une ligne commence toujours par le signe >> (prompt).

**Quelques conseils pour une utilisation efficace de la Command Window :**

- La ligne de commande commence par le signe >> qu'il ne faut pas taper. Il n'apparait pas lors de l'exécution d'une instruction.
- Une instruction tapée dans la ligne de commande est immédiatement exécutée après validation par la touche *'entrée'* ;
- Le symbole point-virgule ';' est un inhibiteur d'affichage. C'est-à-dire, si une ligne de commande produisant un résultat se termine par ; ce résultat n'est pas affiché à l'écran. Tandis que, si une ligne de commande produisant un résultat ne se termine pas par ; ce résultat est affiché à l'écran.
- Toute ligne de commande est modifiable tant que on n'a pas validé par un *'entrée'*.
- On peut rappeler toute ligne dans la ligne de *Command Window* en utilisant les flèches du clavier ↑ et ↓. De même, les flèches du clavier → et ← permettent de se déplacer dans une ligne.
- Si une instruction produit un résultat et ne comporte pas d'indication d'affectation, le résultat est affecté par défaut à la variable *ans* (answer).

**2. Workspace**

Contient le nom, la taille et le type des variables reconnues par *Command Window*. On peut charger et sauvegarder des données au moyen des icônes 'open' et 'save' du bandeau du *Workspace*. Un clic-droit sur les variables offre de nombreuses options telles que : Open (ouvrir), Duplicate (copie), Save as (enregistrer sous), Delete (supprimer)…etc.

**3. Command History**

Contient toute les lignes de commande validées dans la *Command Window* depuis le précédent nettoyage. C'est cet historique qui nous permet de naviguer dans les commandes déjà validées. Par ailleurs, on peut parfois enregistrer dans un fichier indépendant la liste des commandes utilisées. Pour ce faire on utilise la commande : *'diary filename'*. Elle permet de créer un fichier *filename* qui enregistrera toutes les commandes entrées jusqu'à ce que l'on utilise *'diary off'*.

**4. Current Directory**

Indique le répertoire actif, c'est-à-dire celui dans lequel toutes les fonctions de lectures/écritures ont lieu par défaut. Cette fenêtre permet de gérer les fichiers et de changer de répertoire actif. MATLAB définit le '*path*' ou chemin qui se divise en deux

types : '*matlabpath*' et '*userpath*'. Le premier, correspond au chemin par défaut dans lequel MATLAB cherche les fonctions et il peut contenir un grand nombre de dossiers. Le second (*userpath*), est un unique dossier qui est propre à l'utilisateur lors d'une session. Par défaut tous les codes sont sauvegardés dans le chemin : 'C:\MATLAB7\work'. La commande 'pwd', vous permet également d'afficher le chemin par défaut du répertoire de travail. Il est toujours conseillé de définir le '*userpath*' lorsque l'on commence une session, on peut le faire comme suit : '*file → set path → add Folder*'. Vous pouvez consulter le chemin de votre répertoire de travail en tapant a la commnde window : '*path*', le chemin en haut de la liste est votre répertoire courant de travail (tapez la commande '*path*' pour confirmer que votre répertoire de travail est au top de la liste). Vous pouvez également supprimer un chemin utilisé comme suit : '*file → set path → sélectionner le répertoire à supprimer → remove*'.

D'autres fenêtres très intéressantes de MATLAB :

**1. Editor**

La plupart de votre travail sous MATLAB va consister à créer ou modifier des fichiers .m qui est le suffixe standard pour les procédures MATLAB. On utilise souvent la '*Command Window*' pour commandes simples, mais si il s'agit de plusieurs dizaines de ligne de code on a recours à l'utilisation de l'*Editor*. Cet éditeur nous permet de créer des fichiers *script* ou des *fonctions* selon notre choix, dont les deux sont exploitables depuis la *Command Window*.

**2. Help**

C'est important d'utiliser le *help* (aide), lorsque l'on programme avec un langage de programmation de haut-niveau (comme MATLAB), où le nombre de fonctions est très important et la syntaxe est parfois complexe. La commande help, vous permet d'afficher l'aide de MATLAB. Alors que la commande '*help FunctionName*' permet d'afficher une explication et la syntaxe de la fonction *FunctionName*. Il est essentiel que vous vous familiariser avec les outils de l'aide de MATLAB pour réussir dans ce cours.

**3. Commandes d'interaction**

| Commande | Objectif |
|---|---|
| Who | Affiche le nom des variables dans le *Workspace*. |
| Whos | Affiche des informations sur les variables dans le *Workspace*. |
| Clear x y | Supprime les variables x et y du *Workspace*. |
| Clear, clear all | Supprime toutes les variables dans le *Workspace.* |
| Clc | Efface l'écran des commandes dans la *Command Window*. |
| Exit, quit | Quitter MATLAB. |
| format | Définit le format de sortie pour les valeurs numériques (voir **Tableau 1.4** pour plus de détails) |

**Tableau 1.1.** Quelques commandes d'interaction dans MATLAB.

## 1.3. Outils de Base

Le principe de MATLAB est de considérer la plupart des objets comme des matrices. Il faut donc considérer les opérations arithmétiques +, -, *, / comme des opérations matricielles.

### 1.3.1. Types de variables

Il existe cinq types de variables sous MATLAB : les entiers, les réels, les complexes, les chaines de caractères et le type logique. La déclaration sous MATLAB est très simple, on utilise la ligne suivante pour déclarer une variable: '*nom variable = valeur/ expression*'. Définissons une variable de chaque type comme suit :

```
>> a=5.8; b=9+i; c='hello';
>> tr1=true(1==1);tr2=logical(1);
>> x=int8(2);
>>
```

**a** représente un réel, **b** un complexe, **c** une chaine de caractères, **tr1** et **tr2** des logiques définies de deux manière différentes et **x** est un entier codé sur 8 bits. Tous les variables déclarées sont stockées dans le *workspace* est peuvent êtres afficher avec leurs types en utilisant la fonction : '*whos*'. Cette fonction donne une description détaillée (nom de variable, taille, taille en bits et type). La commande '*who*' donne seulement le nom des variables dans le *workspace*.

```
>> whos
  Name      Size                    Bytes  Class

  a         1x1                         8  double array
  b         1x1                        16  double array (complex)
  c         1x5                        10  char array
  tr1       1x1                         1  logical array
  tr2       1x1                         1  logical array
  x         1x1                         1  int8 array
```

Il est impossible de déclarer le type d'une variable lorsque l'on crée une variable dans MATLAB. Il pourrait être utile donc de vérifier le type des variables selon les fonctions : *ischar, islogical, isreal*. MATLAB distingue les majuscules et les minuscules.

MATLAB propose une suite de constantes prédéfinies (à être utilisés sans les déclarés) (voir **Tableau 1.2**)

| La constante | La valeur |
|---|---|
| pi | $\pi$=3.1416 |
| exp(1) | e=2.7183 |
| i | = $\sqrt{-1}$ |
| j | = $\sqrt{-1}$ |
| inf | $\infty$ |
| NaN | Not a Number (pas un numéro) |
| eps | = 2.2204e-016 ($2x10^{-16}$) |

**Tableau 1.2.** Exemples de constantes prédéfinies dans MATLAB.

### 1.3.2. Les Nombres sous MATLAB

MATLAB utilise une notation décimale conventionnelle, avec un point décimal facultatif '.' et le signe '+' ou '–' pour les nombres signés. La notation scientifique utilise la lettre 'e' pour spécifier le facteur d'échelle en puissance de 10. Les nombres complexes utilise les caractères 'i' et 'j' (indifféremment) pour designer la partie imaginaire (voir **Tableau 1.3**).

| Le type | Exemples |
|---|---|
| Entier | 5 - 83 |
| Réel en notation décimale | 0.0205 3.1415926 |
| Réel en notation scientifique | 1.60210e- 20 6.02252e23 ($1.60210 \times 10^{-20}$ et $6.02252 \times 10^{23}$) |
| Complexe | 5+3i - 3.14159j |

**Tableau 1.3.** Exemples de Types de variables

MATLAB utilise toujours les nombres réels (double précision) pour faire les calculs, ce qui permet d'obtenir une précision de calcule allant jusqu'aux 16 chiffres significatifs. Le résultat d'une opération de calcule est par défaut affichée avec quatre chiffre après la virgule. Pour afficher davantage de chiffres utiliser les commandes dans le **Tableau 1.4**.

| Commande | Objectif |
|---|---|
| format short | affiche les nombres avec 4 chiffres après la virgule. |
| format long | affiche les nombres avec 14 chiffres après la virgule. |
| format bank | affiche les nombres avec 2 chiffres après la virgule. |
| format rat | affiche les nombres sous forme d'une ration (a/b). |

**Tableau 1.4.** Commandes pour personnaliser le nombre de chiffres après la virgule.

La fonction *vpa* peut être utilisée pour présenter le nombre de décimaux désirés après la virgule. Exemple d'utilisation: *vpa(sqrt(2),50)*. Cela affiche 50 chiffres après la virgule.

### 1.3.3. Fonctions et opérateurs sur les nombres

MATLAB assure les quatre opérations arithmétiques connus (+, -, *, /) sur les nombres. Il peut également utiliser les fonctions trigonométriques, puissance, logarithmiques, les fonctions d'arrondis, les fonctions d'arithmétiques, et des fonctions des nombres complexes…etc. Une liste (non exhaustive) des fonctions incorporées dans MATLAB est donnée dans le **Tableau 1.5.**

| Fonction | Objectif |
|---|---|
| Fonctions trigonométriques, puissance, logarithmiques | |
| **exp(x)** | exponentielle de x |
| **log(x)** | logarithme népérien de x |
| **log10(x)** | logarithme en base 10 de x |
| **x^n** | x à la puissance n |

| | |
|---|---|
| **sqrt(x)** | racine carré de x |
| **abs(x)** | valeur absolue de x |
| **sign(x)** | 1 si x>0 et -1 si x<=0 |
| **sin(x)** | sinus de x |
| **cos(x)** | cosinus de x |
| **tan(x)** | tangente de x |
| **asin(x)** | sinus inverse de x (arcsin de x) |
| **sinh(x)** | sinus hyperbolique de x |
| **asinh(x)** | sinus hyperbolique inverse de x |
| Fonctions d'arrondis | |
| **round(x)** | entier le plus proche a x |
| **floor(x)** | arrondi par défaut de x |
| **ceil(x)** | arrondi par excès de x |
| Fonctions d'arithmétiques | |
| **rem(m,n)** | reste de la division entière de m par n |
| **lcm(m,n)** | plus petit commun multiple de m et n |
| **gcd(m,n)** | plus grand commun diviseur de m et n |
| **factor(n)** | décomposition en facteurs premiers de n |
| Fonctions sur les nombres complexes | |
| **conj(z)** | conjugué de z |
| **abs(z)** | module de z |
| **angle(z)** | argument de z |
| **real(z)** | partie réelle de z |
| **imag(z)** | partie imaginaire de z |

**Tableau 1.5.** Fonctions prédéfinies sous MATLAB.

L'évaluation d'une expression s'exécute de gauche à droite en considérant la priorité des opérations comme suit : **Les parenthèses (), la puissance ^ et le transposé ', la multiplication * et la division /, l'addition + et la soustraction -**

## Exercices d'application :

### Exercice 1 :

Soit les variables x et y définies comme suit:

```
>> x=10;y=20;
```

Donner le résultat des commandes suivantes:

```
>> x<y
>> y<x
>> ~(x<y)
>> (x==y)
>> (x<11)&(y==3)
```

### Exercice 2 :

Donner les commandes MATLAB permettant de calculer les expressions suivantes :

1. $\frac{1}{\sqrt{8^3+2}} - \frac{2\sin(45)}{e^2} + \ln(4)$
2. $5^{10} + e^{-3} - \frac{2\cos(90)}{15} + \log(10)$

Les angles sont donnés en degré.

**Exercice 3 :**

Soit la fonction à deux variables suivante : $f(x,y) = \frac{1}{\sqrt{|x^3|+|y^3|}}$

Calculer : $f(-2,-3), f(-5,2.25), f(0.25,3.05)$

# 2. LES VECTEURS

Un vecteur sous MATLAB est une collection d'éléments du même type. Ils sont de deux types : ligne (une seule ligne, plusieurs colonnes) et colonne (une seule colonne, plusieurs lignes). Les différentes opérations qu'on peut réaliser sur un vecteur sont décrites par le **Tableau 1.7**.

## 2.1. Définition de vecteurs lignes et colonnes

Pour définir les tableaux à une ligne, ou vecteurs lignes on peut utiliser trois façons :

**(1) Donner la liste** entre crochets séparées par un espace:

```
>> v = [1  12.5  4  log(21)]
v =
 1.0000   12.5000    4.0000    3.0445
```

Ou séparées par des virgules :

```
>> v = [1,12.5,4,log(21)]
v =
1.0000   12.5000    4.0000    3.0445
```

**(2) le définir globalement :** Si h est du signe de (M-m), l'instruction X = m :h :M définit le tableau X à une ligne formé de tous les nombres de la forme $m + kh\ avec\ k \in \mathbb{N}\ et\ m + kh\ entre\ m\ et\ M$ Par contre, $si, h > 0\ et\ M - m < 0\ ou\ h < 0\ et\ M - m > 0, X\ est\ vide$. Si h =1, on utilise X=m :M

```
>> X=1:0.3:2
X =
1.0000    1.3000    1.6000    1.9000
```

**(3) utiliser des fonctions d'initialisation :** Linspace(a,b,n) définit un vecteur ligne de n nombres régulièrement espacés entre a et b.

```
>> linspace(2,10,5)
ans =
 2    4    6    8   10
```

X= **zeros**(1,n) : définit un vecteur ligne de n valeurs égales à 0 ;

```
>> X=zeros(1,5)
X =
0    0    0    0    0
```

X= ones(1,n) : **définit** un vecteur ligne de n valeurs égales à 1.

```
>> X=ones(1,5)
X =
    1    1    1    1    1
```

Pour définir les tableaux à une colonne, ou vecteurs colonnes on peut utiliser trois façons :

**(1) Donner la liste :** entre crochets séparées par un des points-virgules ;

```
>> v = [1 ;12.5 ;4;log(21)]
v =
   1.0000
  12.5000
```

```
   4.0000
   3.0445
```

Ou écrivez la liste verticalement :

```
>> [1
2
3
4]
ans =
    1
    2
    3
    4
```

**(2) utiliser des fonctions d'initialisation :** X= zeros(n,1) : définit un vecteur colonne de n valeurs égales à 0 ;

```
>> X=zeros(5,1)
X =
    0
    0
    0
    0
    0
```

X= **ones**(n,1) : définit un vecteur colonne de n valeurs égales à 1 ;

```
>> X=ones(5,1)
X =
    1
    1
    1
    1
    1
```

### 2.2. Manipuler un vecteur

- **Transposé d'un vecteur :** permet de passer d'une ligne à une colonne ou le contraire. Se réalise avec le signe '.

**Exemple :**

```
>> v=[1 2 3]
v =
    1    2    3
>> v'
ans =
    1
    2
    3
```

- **Concaténer deux vecteurs :** permet de coller deux vecteurs.

**Exemple :**

```
>> v1= [1 3 5];
>> v2= [9 10 11];
>> v3= [v1 v2]
v3 =
    1    3    5    9   10   11
```

- **Extraction des sous-ensembles et accès aux éléments d'un vecteur :** permet de définir de nouveaux vecteurs extraits des vecteurs d'origines à l'aide des indices

**Exemple :**

```
>> v=[5 -1 13 -6 7]       %création du vecteur v de 5 valeurs
v =
    5   -1   13   -6    7
>> v(3)                    % la 3eme position
ans =
   13
>> v(2:4)                  % de la deuxième position jusqu'au quatrième
ans =
   -1   13   -6
>> v(4:-2:1)               % de la 4eme position jusqu'à la 1ere avec un pas = -2
ans =
   -6   -1
>> v(3:end)                % de la 3eme position jusqu'à la dernière
ans =
   13   -6    7
>> v([1,3,4])              % la 1ere, 3eme et 4eme position uniquement
ans =
    5   13   -6
>> v(1)=8                  % modifier la valeur du premier élément par la valeur 8
v =
    8   -1   13   -6    7
>> v(6)=-3                 % ajouter un 6ème élément avec la valeur -3
v =
    8   -1   13   -6    7   -3
>> v(9)=5                  % ajouter un 9eme élément avec la valeur 5
v =
    8   -1   13   -6    7   -3    0    0    5
```

### 2.3. Fonctions communes aux vecteurs lignes et colonnes

Des fonctions sur les vecteurs sont montrées par le **Tableau 1.7**.

| Fonction | objectif |
|---|---|
| sum(x) | Somme des éléments du vecteur x |
| prod(x) | Produit des éléments du vecteur x |
| max (x) | Plus grand élément dans x |
| min(x) | Plus petit élément dans x |
| mean(x) | Moyenne des éléments de x |
| sort(x) | Ordonne les éléments du vecteur x par ordre croissant |
| fliplr(x) | Renverse l'ordre des éléments du vecteur x |
| length(x) | Retourne la taille d'un vecteur |
| find(x) | Retourne un vecteur des indices des éléments non-nul de x |
| find(condition sur A) | Retourne les indices des valeurs satisfaisants une condition sur x |

**Tableau 1.7.** Fonctions à appliquer sur les vecteurs.

**2.4. Opérations vectorielles :** il est possible de réaliser les opérations algébriques usuelles +, -, *, / pour les vecteurs. La somme et la soustraction sont des opérations termes à termes, donc les vecteurs doivent être de même dimension. Pour le produit et la division et la puissance termes à termes on doit remplacer * et / par .* et ./ et .^ (voir **Tableau 1.8**).

| opération | Exemple d'application : |
|---|---|
| | >> u=[-2 6 1]; % création de u<br>>> v=[3 -1 4]; % création de v |
| Addition (+) | >> u+2 % Addition d'un scalaire à un vecteur<br>ans =<br>0 8 3<br>>> u+v % Addition de deux vecteurs<br>ans =<br>1 5 5 |
| Soustraction (-) | >> u-2 % soustraction d'un scalaire d'un vecteur<br>ans =<br>-4 4 -1<br>>> u-v % soustraction élément par élément de deux vecteurs<br>ans =<br>-5 7 -3 |
| Multiplication élément par élément (.*) | >> u*2 % Produit d'un scalaire par un vecteur<br>ans =<br>-4 12 2<br>>> u.*2 % produit élément par élément d'un scalaire par un vecteur<br>ans =<br>-4 12 2<br>>> u.*v % produit élément par élément de deux vecteurs<br>ans =<br>-6 -6 4<br>>> v.*u % produit élément par élément de deux vecteurs<br>ans =<br>-6 -6 4 |
| Division élément par élément (./) | >> u/2 % Division d'un vecteur par un scalaire<br>ans =<br>-1.0000 3.0000 0.5000<br>>> u./2 % Division élément par élément d'un vecteur par un scalaire<br>ans =<br>-1.0000 3.0000 0.5000<br>>> u./v % Division élément par élément de deux vecteurs<br>ans =<br>-0.6667 -6.0000 0.2500 |
| Puissance élément par élément (.^) | >> u.^2 % puissance élément par élément d'un vecteur à un scalaire<br>ans =<br>4 36 1<br>>> u.^v % puissance élément par élément de deux vecteurs<br>ans =<br>-8.0000 0.1667 1.0000 |

**Tableau 1.8.** Opérations vectorielles.

## Exercice d'application :

- Définir le vecteur $x = (\frac{\pi}{6}, \frac{\pi}{4}, \frac{\pi}{3})$
- Calculer $y1 = \sin(x)\ et\ y2 = \cos(x)$
- Définir le vecteur $[0: 0.1: 2\pi]$.

- Combien y a-t-il de valeurs dans ce vecteur?

# 3. LES MATRICES

Les matrices sont considérées comme les piliers de MATLAB. Un tableau de n lignes et p colonnes est une matrice de taille $n \times p$.

## 3.1. Définir une matrice

On peut définir une matrice de taille $n \times p$ de diverses manières :

1) **Le définir en extension :** on décrit la matrice ligne par ligne. Les éléments de la même ligne sont séparés par des espaces ou de virgules et les lignes sont séparées par des points-virgules.

```
>> D=[1 2 4;2 3 5]
D =
   1    2    4
   2    3    5

>> D=[1,2,4;2,3,5]
D =
   1    2    4
   2    3    5

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

2) **Définir une matrice depuis des vecteurs :** une matrice peut être définie par la concaténation de plusieurs vecteurs lignes ou colonne.

**Exemple :**

```
>> v1=1:4                    %création du premier vecteur v1
v1 =
   1    2    3    4

>> v2=5:5:20                 %création du premier vecteur v2
v2 =
   5   10   15   20

>> v3=4:4:16                 %création du premier vecteur v3
v3 =
   4    8   12   16

>> M=[v1;v2;v3]              % définition de M à partir de v1,v2 et v3 (ces vecteurs forment les lignes de
M)
M =
   1    2    3    4
   5   10   15   20
   4    8   12   16
>> M=[v1' v2' v3'] % définition de M à partir du transposé de v1, v2 et v3 (ce vecteurs forment les colonnes de
M)
M =
   1    5    4
   2   10    8
   3   15   12
```

```
   4   20   16
>> M=[v1;v1]            %M est définie par v1 dans sa 1ère et la 2ème ligne
M =
   1    2    3    4
   1    2    3    4
```

3) **Utiliser des fonctions d'initialisation :** si n et p sont des entiers strictement positifs, on peut utiliser les fonctions dans le **Tableau 1.9** pour initialiser ou générer automatiquement une matrice de dimension $n \times p$.

| **Fonction** | **Objectif** |
|---|---|
| zeros(n,p) | Crée une matrice de taille nxp ne contenant que des 0 |
| zeros(n) | Crée une matrice de taille nxn ne contenant que des 0 |
| ones(n,p) | Crée une matrice de taille nxp ne contenant que des 1 |
| ones(n) | Crée une matrice de taille nxn ne contenant que des 1 |
| eye(n,p) | Crée une matrice de taille nxp contenant des 0 partout sauf sur la diagonale où il y a des 1 |
| eye(n) | Crée une matrice de taille nxn contenant des 0 partout sauf sur la diagonale où il y a des 1 |
| rand(n,p) | Crée une matrice de taille nxp contenant des éléments générés de manière aléatoire entre 0 et 1 |
| magic(n) | Crée une matrice de taille nxn dont la somme des diagonales, des lignes et des colonnes sont égaux. |

**Tableau 1.9.** Fonctions prédéfinies de MATLAB utilisées pour initialiser une matrice.

### 3.2. Manipuler une matrice:

1) **Extraction d'un élément :** Pour extraire un élément de la matrice on indique la ligne et la colonne de celle-ci. Si M est une matrice de dimension $L \times C$ L'élément qui est à la $l - iéme$ ligne et la $c - iéme$ colonne de la matrice $M$ est $M(l,c)$.

**Attention :**

- pour MATLAB, les indices sont toujours des entiers strictement positifs car ce sont des numéros de ligne et de colonnes. Par exemple, l'instruction $M(0,1)$ =50 ; provoque une erreur.
- si $l > L\ ou\ c > C$ alors :
  - Invoquer $M(l,c)$ à droite de $'='$ ou dans une expression (logique ou algébrique) provoque une erreur.
  - Invoquer $M(l,c)$ à gauche de $'='$ agrandit le tableau, les termes manquants sont initialisés à 0.

**Exemple :**

```
>> M=[3 5 6; 7 9 8]          % définition de la matrice M
```

```
M =
   3    5    6
   7    9    8
>> M(2,1)                    % extraction de l'élément dans la 2éme ligne et la 1ère colonne de M
ans =
   7
```

Lorsque l'on souhaite extraire une colonne ou une ligne entière on utilise le symbole (:) . donc l'accès à la $l-iéme$ ligne se fait par : $M(l,:)$ et l'accès à la $c-iéme$ colonne se fait par : $M(:,c)$

**Exemple :**

```
>> M(2,:)                    % extraction de la 2ème ligne de M
ans =
   7    9    8
>> M(:,1)                    % extraction de la 1ère colonne de M
ans =
   3
   7
```

Plusieurs combinaisons sont alors possibles. On peut extraire 2 colonnes par exemple en faisant :

```
M =
   3    5    6
   7    9    8
>> M(:,[1,2])
ans =
   3    5
   7    9
```

2) **Suppression d'un élément :** on peut supprimer des éléments de la matrice sans la supprimer en utilisant la commande : $M(:,c) = []$ pour les supprimer la colonne dans la $c-$ème colonne, et $M(l,:) = []$ pour la $l-$ème ligne.

**Exemple :**

```
>> M=[1 2 3; 4 5 6; 7 8 9]                % définition de la matrice M
M =
   1    2    3
   4    5    6
   7    8    9

>> M(:,3)=[]                              % supprimer tous les éléments de la 3 ème colonne
M =
   1    2
   4    5
   7    8

>> M(3,:)=[]                              % supprimer tous les éléments de la 3 ème ligne
M =
   1    2
   4    5
```

3) **Ajout de lignes ou de colonnes:** Pour ajouter une nouvelle colonne on utilise la commande : $M = [M, [la\ nouvelle\ colonne]]$ avec la condition que le nombre des

éléments dans la nouvelle colonne = $l$ $(nombre\ de\ lignes\ de\ M)$. Aussi, pour ajouter une nouvelle ligne on utilise la commande : $M = [M; [la\ nouvelle\ ligne]]$ avec la condition que le nombre des éléments de la nouvelle ligne = $c$ $(nombre\ de\ colonnes\ de\ M)$.

**Exemple :**

```
>> M=[1 2 3; 4 5 6;7 8 9]                    % définition de M
M =
   1    2    3
   4    5    6
   7    8    9
>> M=[M,[5 ;66; 99]]                          % ajout de la nouvelle colonne [5 ;66 ;99]
M =
   1    2    3    5
   4    5    6   66
   7    8    9   99
>> M=[M;[5 66 99 100]]                        % ajout d'une nouvelle ligne à M [5 66 99 100]
M =
   1    2    3    5
   4    5    6   66
   7    8    9   99
   5   66   99  100
```

**3.3. Opérations matricielles:** les opérations matricielles sont à la base du calcul sous MATLAB.

1) **Addition et soustraction :** sont possibles uniquement sur des matrices de taille identique (même nombre de lignes et même nombre de colonnes). Ce sont des opérations termes à termes, similaires aux opérations scalaires.

```
>> A=[1 2 3;4 5 6]                % définition de A
A =
   1    2    3
   4    5    6
>> B=[1 2 3; 4 5 6]               % définition de B
B =
   1    2    3
   4    5    6
>> A+B                            % calcul de la somme de A et B
ans =
   2    4    6
   8   10   12
>> A-B                            % calcul de la soustraction de B de A
ans =
   0    0    0
   0    0    0
```

2) **Multiplication** : il existe deux types de multiplication: la multiplication dite matricielle et la multiplication termes à termes. Cette dernière est l'analogue de l'addition et de la soustraction. Elle est noté de façon spécifique pour la distinguer de la vraie multiplication $(A.* B)$ avec $A\ etB$ sont deux matrices de dimension identiques.

```
>> A=[1 2 3;4 5 6]                % définition de la matrice A
A =
```

```
   1    2    3
   4    5    6
>> B=[1 2 3; 4 5 6]                % définition de la matrice B
B =
   1    2    3
   4    5    6
>> A.*B                            % calcul du produit élément par élément de A par B
ans =
   1    4    9
  16   25   36
```

Le véritable produit matriciel entre la matrice $A$ de dimension $m \times n$ et de la matrice $B$ de dimension $n \times p$ est une matrice $M = AB$ de taille $m \times p$. Pour que ce produit sera réalisé, il est nécessaire que le nombre de colonnes de $A$ soit égal aux nombre de lignes de $B$. Soit les éléments de $A$ : $a_{ij}$ et ceux de $b_{ij}$, alors les éléments de la matrice $M$ sont définies par :

$$m_{i,j} = \sum_{0<k\leq n} a_{ik}\, b_{kj} \ldots\ldots\ldots\ldots (1)$$

Le principe de la formule est décrit de façon simple par la **Figue 1.2** :

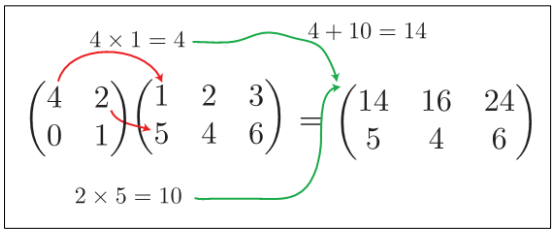


**Figure 1.2:** représentation graphique de la multiplication matricielle

Sous MATLAB, on calcule le produit en utilisant le signe * :

```
>> A=[1 2 3;4 5 6]                 % définition de A de taille 2x3
A =
   1    2    3
   4    5    6
>> B=[1 2;3 4;5 6]                 % définition de B de taille 3x2
B =
   1    2
   3    4
   5    6
>> A*B                             % calcule de A*B, la matrice résultante est de taille 2x2
ans =
  22   28
  49   64
>> C=[1 2 9;3 4 5]                 % définissons une nouvelle matrice C de taille 2x3
C =
   1    2    9
   3    4    5

>> A*C              % le calcul du produit A*C ne se réalise pas, puisque les dimensions ne respectent pas les conditions.
??? Error using ==> mtimes
Inner matrix dimensions must agree.
```

3) **Puissance (carré):** De même, si l'on souhaite obtenir le carré d'une matrice (au sens du produit termes à termes de cette matrice par elle-même).

```
>> A=[1 2 3;4 5 6]              % définition de la matrice A
A =
   1   2   3
   4   5   6
>> A.^2                         % calcul de son carré A²
ans =
   1   4   9
  16  25  36
```

**4) Inverse:** puisque on dispose du produit matriciel, on peut définir l'inversion. En effet, on note $A^{-1}$, l'inverse de la matrice $A$ (quand elle existe) et on définit $A^{-1}$ par : $A^{-1}A = AA^{-1} = I$ où $I$ est la matrice identité. Cette notion d'inverse est très importante pour la résolution d'un système d'équations. MATLAB calcule l'inverse par la fonction $inv(M)$.

```
>> M=[4 2;0 1];                 % définition de la matrice M
>> A=inv(M)                     % Calcule de l'inverse de M la matrice A
A =
   0.2500   -0.5000
        0    1.0000
```

**5) Division :** si $A$ et $B$ sont deux matrices de dimensions compatibles et $B$ est inversible. la division se définit à partir de l'inverse. $A/B = AB^{-1}$ .

```
>> A=[4 2;0 1];                 % définition de la matrice A
>> B=[1 2;5 4];                 % définition de la matrice B
>> M=A/B                        % calcule de la matrice M la division de la matrice A par B
M =
  -1.0000    1.0000
   0.8333   -0.1667
```

**6) Transposé :** lorsque $A$ est une matrice à coefficient réels, la matrice $A$' est la transposée de $A$. C'est-à-dire que si $A = (a_{i,j})_{i,j}$ est une matrice de taille $n \times p$ a coefficients réels, alors $A' = (a'_{i,j})_{i,j}$ est la matrice de taille $p \times n$ definie par : $a'_{i,j} = a_{j,i}$ pour tout couple $(i, j)$.

```
>> A=[4 2;0 1]                  % définition de la matrice A
A =
   4   2
   0   1
>> A'                           % calcule de transpose de A
ans =
   4   0
   2   1
```

**Tableau 1.10** est un résume des différentes opérations matricielles :

| Opération | Objectif |
|---|---|
| + | Addition |
| - | Soustraction |
| .* | Multiplication terme par terme |
| ./ | Division terme par terme |

| | |
|---|---|
| .\ | Division inverse terme par terme |
| .^ | Puissance (carre) terme par terme |
| * | Produit matriciel |
| / | Division matriciel |

**Tableau 1.10.** Opérations matriciel sous MATLAB.

**3.4. Fonctions prédéfinies pour le traitement des matrices:** MATLAB inclut une liste très importante de fonctions prédéfinies qui nous permet de manipuler de grandes matrices d'une façon très simple et efficace. Nous regroupons dans le **Tableau 1.11** une liste non exhaustive des fonctions utiles pour le traitement des matrices on les clarifie par des exemples exécutés sous MATLAB.

| Fonction | Objectif | Exemple |
|---|---|---|
| **Size** | Calcule la taille d'une matrice | >> A=[4 2;0 1]<br>A =<br>4 2<br>0 1<br>>> [l,c]=size(A)<br>l =<br>2<br>c =<br>2 |
| **det** | Calcule le déterminant d'une matrice | >> A=[1 2;3 4];<br>>> det(A)<br>ans =<br>-2 |
| **rank** | Calcule le rang d'une matrice | >> rank(A)<br>ans =<br>2 |
| **trace** | Calcule la trace d'une matrice | >> trace(A)<br>ans =<br>5 |
| **eig** | Calcule les valeurs propres d'une matrice | >> eig(A)<br>ans =<br>-0.3723<br>5.3723 |
| **dot** | Calcule le produit scalaire de deux vecteurs | >> v1=[1 5 -3];<br>>> v2=[3 2 5];<br>>> dot(v1,v2)<br>ans =<br>-2 |
| **norm** | Calcule la norme d'un vecteur | >> norm(v1)<br>ans =<br>5.9161 |
| **cross** | Calcule le produit vectoriel de deux vecteurs | >> a = [1 2 3];<br>b = [4 5 6];<br>c = cross(a,b)<br>c =<br>-3 6 -3 |
| **diag** | Renvoie la diagonal d'une matrice | >> A=[1 2 3; 4 5 6;4 8 9]<br>A =<br><br>1 2 3<br>4 5 6 |

| | | |
|---|---|---|
| | | 4 8 9<br>>> v=diag(A)<br>v =<br>1<br>5<br>9 |
| **diag(v)** | Crée une matrice ayant le vecteur v dans la diagonale et 0 ailleurs | >> M=diag(v)<br>M =<br>1 0 0<br>0 5 0<br>0 0 9 |
| **tril** | Renvoie la partie triangulaire inférieure d'une matrice | >> tril(A)<br>ans =<br>1 0 0<br>4 5 0<br>4 8 9 |
| **triu** | Renvoie la partie triangulaire supérieure d'une matrice | >> triu(A)<br>ans =<br>1 2 3<br>0 5 6<br>0 0 9 |

**Tableau 1.11.** Fonctions prédéfinies pour le traitement matriciel sous MATLAB.

**3.5. Résoudre un système linéaire (équations linéaires):** soit A une matrice carrée de taille $n$ et $B = (b_i)_i$une matrice colonne $n \times 1$,

Si le système d'équations linéaires $\begin{cases} a_{11}x_1 + \cdots + a_{1n}x_n = b_1 \\ a_{21}x_1 + \cdots + a_{2n}x_n = b_2 \\ \quad \cdots \quad \cdots \quad \cdots \\ a_{n1}x_1 + \cdots + a_{nn}x_n = b_n \end{cases}$

a une et une seule solution $(x_1, \ldots, x_n)$ alors la colonne $X = \begin{pmatrix} x_1 \\ \cdot \\ \cdot \\ \cdot \\ x_n \end{pmatrix}$ est obtenue par la commande $X = A\backslash B$ connue sous le nom de division à gauche.

**Exemple :**

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

```
>> A=[2 3;4 5];
>> B=[1;3];
>> X=A\B
X =
   2
  -1
```

La réponse signifie $x = 2, y = -1$

## Exercice:

Créer une matrice $\boldsymbol{M}$ aléatoire carrée de taille $\mathbf{8 \times 8}$, contenant des valeurs comprises entre1 $et$ 8.

Calculer la somme des éléments de cette matrice.

Calculer la moyenne des colonnes ; la moyenne des lignes puis la moyenne de la matrice.

Extraire la diagonale de la matrice.

Afficher la partie triangulaire.