Chapitre 2 - Introduction à Matlab

II

1. Qu'est ce que Matlab

Le logiciel Matlab (MATtrix LABoratory) constitue un système interactif et convivial de calcul numérique et de visualisation graphique. Destiné aux ingénieurs, aux techniciens et aux scientifiques, c'est un outil très utilisé, dans les universités comme dans le monde industriel, qui intègre des centaines de fonctions mathématiques et d'analyse numérique. Matlab permet de manipuler des matrices, d'afficher des courbes et des données, de mettre en œuvre des algorithmes et de créer des interfaces utilisateur.

Matlab est constitué d'un noyau capable d'interpréter puis d'évaluer les expressions numériques matricielles qui lui sont adressées sous différentes formes :

- soit directement au clavier depuis une fenêtre de commande ;
- soit sous forme de séquences d'expressions ou scripts enregistrées dans des fichiers texte appelés m-files (ou fichiers .m) et exécutées depuis la fenêtre de commande.

Ce noyau est complété par une bibliothèque de fonctions prédéfinies, très souvent sous forme de fichiers m-files, et regroupés en paquages ou toolboxes. Il est possible d'ajouter des toolboxes spécifiques à un problème mathématique précis (exemple : Optimization Toolbox ou Signal Processing Toolbox), ou encore des toolboxes crées par l'utilisateur lui même.

2. Une session Matlab

L'interface utilisateur de Matlab varie légèrement en fonction de la version installée (nous utilisons ici la version 7.5). Cependant, elle est constituée (principalement) d'une fenêtre de commandes et un éditeur de texte permettant d'écrire des scripts et des fonctions). La figure suivante illustre les deux composants principaux de l'interface Matlab.

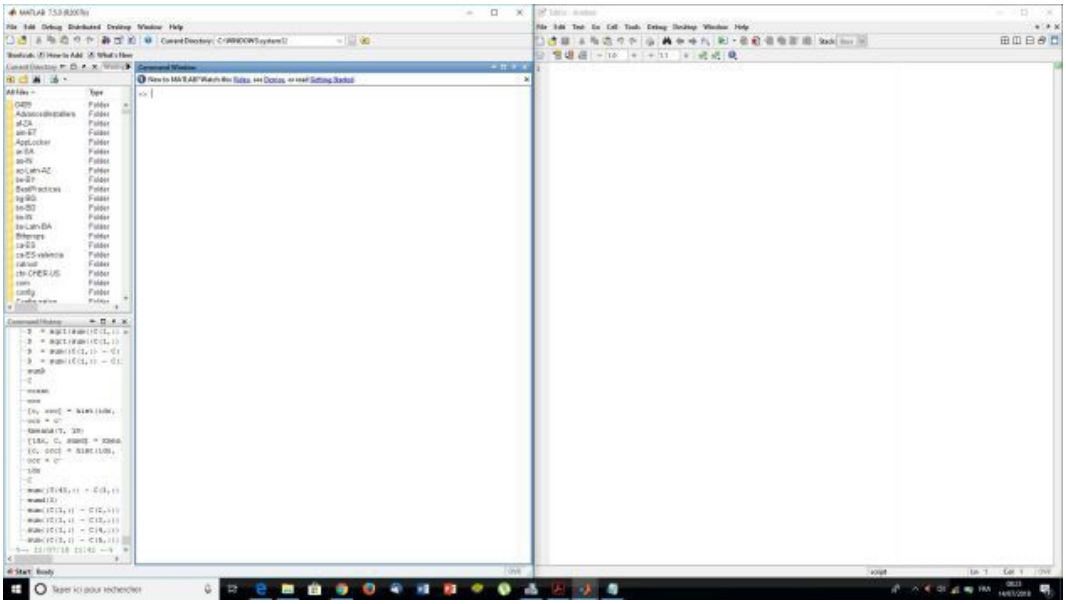

Figure 1 : l'interface utilisateur de Matlab

2.1. Lancer, quitter Matlab

Pour lancer Matlab, il suffit de double cliquer sur l'icône du logiciel. La fenêtre de commande de Matlab s'ouvre alors et on tape les commandes ou les expressions à évaluer à droite du prompt ">>" . Le processus d'évaluation est déclenché par la frappe de la touche « Enter ».

2.2. La fenêtre de commandes

La fenêtre de commandes est la fenêtre principale de l'environnement Matlab. Elle donne accès à l'interpréteur de commandes et exécute les commandes Matlab saisies par l'utilisateur. Les chevrons ">>" indiquent l'endroit ou il faut saisir la commande.

La fenêtre de commandes donne aussi accès à l'historique de toutes les commandes que vous avez tapées. Vous pouvez les retrouver et les modifier grâce aux touches de direction. Il est possible d'utiliser les touches ⇑ et ⇓ pour parcourir les commandes exécutées précédemment, puis les éditer. Pour relancer une commande, il suffit d'appuyer sur la touche « Enter » (aussi appelée retour à la ligne).

Il est aussi possible de retrouver toutes les commandes commençant par un groupe de lettres. Pour cela, il faut taper les premières lettres de la commande recherchée, puis appuyer plusieurs fois sur ⇑ pour parcourir les commandes correspondant à cette recherche.

🔎 *Remarque : Fonctions et commandes*

Certaines commandes de Matlab calculent des valeurs numériques ou vectorielles, celles là sont appelées des fonctions. Elles sont caractérisées par le fait que leurs arguments (lorsqu'ils existent) sont placés entre parenthèses. Elles se comportent de façon assez similaire aux fonctions mathématiques.

Attention

Certaines fonctions peuvent être appelées avec des paramètres qui sont différents (par leur nombre ou par leur nature). Le traitement effectué par ces fonctions dépend principalement de leurs arguments. Par exemple nous verrons plus loin que la fonction `diag` appelée avec une matrice pour argument retourne sa diagonale principale alors que lorsqu'elle est appelée avec un vecteur, elle retourne une matrice diagonale dont le vecteur diagonal est le vecteur donné comme argument.

Donc une fonction (ou une commande) n'est pas caractérisée seulement par son nom, mais par sa signature (nom + arguments).

3. Aide et documentation

Matlab dispose d'une aide fournie et fonctionnelle. Elle est accessible de différentes manières :

- aide fournie par l'environnement Matlab ;
    - la commande `helpwin` : Donne une liste de fonctions Matlab classées par thème. Par exemple, la commande "`helpwin elfun`" ouvre une fenêtre graphique qui liste les fonctions élémentaires les plus utilisées ;
    - les commandes `doc` et `help` : La commande "`help fonction`" (ou "`doc fonction`") donne la définition de la fonction indiquée et ces paramètres ainsi que quelques exemples d'utilisation. la différence entre les deux commandes est que `help` affiche ces messages dans la fenêtre de commandes alors que `doc` les affiche dans une fenêtre de navigation ;
- aide en ligne ;
    - en consultant la documentation officielle en ligne sur l'adresse *https://fr.mathworks.com/help/* ;
    - la commande `lookfor` : la commande "`lookfor sujet`" donne une liste de rubriques de l'aide en ligne en relation avec le sujet indiqué.

Exemple : La commande help

```
>> help sin
 SIN    Sine of argument in radians.
    SIN(X) is the sine of the elements of X.

    See also asin, sind.

    Overloaded methods:
       darray/sin
       sym/sin

    Reference page in Help browser
       doc sin
```

Remarque

Dans tous les exemples de ce cours, les chevrons "`>>`" seront présents pour que vous puissiez faire la différence entre la commande saisie et le résultat produit par Matlab.

```
>> Commande saisie par l'utilisateur

```

```
3   Résultat produit par Matlab
```

4. Les “objets" de Matlab - Listes, vecteurs et tableaux

Objets et classes dans Matlab

La classe principale dans Matlab est la classe `double`, c'est à partir de cette classe que sont définis des types de données plus comme : les nombres complexes et les tableaux. Une autre classe qui est moins fréquemment utilisée est la classe `char`, qui modélise des caractères (alphabets, chiffres et caractères spéciaux).

Dans les nouvelles versions de Matlab (à partir de la version 7) une classe `logical` a été introduite. Cette classe est utilisée pour modéliser des valeurs logiques : vrai ou faux (`true` ou `false`).

4.1. Types de données et valeurs littérales

Le terme *valeur littérale* désigne les valeurs (de tout type de données) qu'on peut directement saisir à partir du clavier. Les types de données qui nous intéressent dans cette section sont les suivants :

- Les types numériques.
- Les tableaux.
- Les chaînes de caractères.

4.1.1. Les types numériques

Les nombres réels ou entiers sont écrits soit sous la forme décimale (en utilisant le point "`.`"comme séparateur décimal) soit en utilisant la notation scientifique. Les nombres complexes sont écrits sous la forme "`a + bi`" tel que `a` et `b` sont des nombres réels. L'exemple suivant montre comment initialiser un nombre complexe.

Exemple : Types numériques dans Matlab

```
1 >> 5       % type entier
2
3 ans =
4
5      5
6
7 >> 1.21E33 % type réel
8
9 ans =
10
11   1.2100e+033
12
13 >> 2 + 3i  % type complexe
14
15 ans =
16
17    2.0000 + 3.0000i
```

Complément : Fonctions relatives aux nombres complexes

L'environnement Matlab fournit quelques fonctions prédéfinies qui facilitent la manipulation des nombres complexes.

- `real` : extrait la partie réelle d'un nombre complexe ;

- `imag` : extrait la partie imaginaire d'un nombre complexe ;
- `abs` : calcule le module d'un nombre complexe ;
- `conj` : renvoie le nombre complexe conjugué de son argument.

4.1.2. Les tableaux

La syntaxe utilisée par Matlab pour saisir un tableau à une ou deux dimensions (autrement dit : matrice) est la suivante :

- le tableau est délimité par deux crochets (`[]`) ;
- les éléments de la même ligne sont séparés par des espaces ou par des virgules ;
- les lignes sont séparées par des points-virgules ou des retours à la ligne ;
- toutes les lignes doivent contenir le même nombre d'éléments.

L'exemple suivant montre comment initialiser des tableaux (de différentes dimensions).

Exemple : Initialiser des tableaux dans Matlab

```
>> [1 2 3 4] % ou [1, 2, 3, 4] pour initialiser un vecteur ligne

ans =

     1     2     3     4

>> [1; 2; 3; 4 + 5i] % pour initialiser un vecteur colonne

ans =

   1.0000
   2.0000
   3.0000
   4.0000 + 5.0000i

>> [1, 2; 3, 4] % initialiser une matrice

ans =

     1     2
     3     4

>> [1, 2, 3; 4, 5] % les lignes doivent contenir le même nombre d'elements
??? Error using ==> vertcat
CAT arguments dimensions are not consistent.
```

Remarque : Vecteur ligne, vecteur colonne et matrice

Dans le reste de ce cours, nous allons utiliser le terme matrice pour décrire un tableau de deux dimensions, le terme vecteur ligne pour décrire un tableau qui comporte une seule ligne (dans certaines références le terme liste est utilisé) et le terme vecteur colonne pour décrire un tableau qui ne comporte qu'une seule colonne (dans certaines références, le terme vecteur "tout court" est utilisé).

a) L'opérateur colon ":"

Avant d'aller plus loin, il est nécessaire de se familiariser avec une notation que l'on retrouve partout sur Matlab, celle de l'opérateur "`:`". Pour deux nombres entiers `E1` et `E2` tels que `E2 > E1`, la commande `E1:E2` génère un vecteur ligne qui comporte tous les entiers dans intervalle `[E1, E2]`.

Pour trois nombres réels `R1`, `R2` et `x`, la commande `R1:x:R2` génère un vecteur ligne qui comporte une liste des nombres qui appartiennent à l'intervalle `[R1, R2]`, et qui s'écrivent sous la forme `R1 + n * x` tel que `n` est un entier positif (si `x` est positif, `R1` doit être inférieur à `R2`, sinon `x` est négatif, `R1` doit être supérieur à R2).

Complément : Générer un vecteur colonne

Il est possible d'utiliser le même opérateur pour générer des vecteurs colonnes en transposant le résultat obtenu avec l'opérateur apostrophe (`'`).

Exemple : Utiliser l'opérateur ":"

```
>> 1.2:0.5:3

ans =

    1.2000    1.7000    2.2000    2.7000

>> 1:3

ans =

     1     2     3

>> (1:3)'

ans =

     1
     2
     3
```

b) La fonction linspace

La fonction `linspace(`v_i`, `v_f`, n)` crée une liste de `n` nombres uniformément répartis entre les valeurs v_i et v_f.

Exemple : Utiliser la fonction linspace

```
>> linspace(1, 2, 5)

ans =

    1.0000    1.2500    1.5000    1.7500    2.0000

>> linspace(1, 2, 5)'

ans =
```

```
10
11     1.0000
12     1.2500
13     1.5000
14     1.7500
15     2.0000
```

4.1.3. Les caractères et les chaînes de caractères

Une chaîne de caractères est suite finie de caractères. En phase de saisie, une chaîne de caractères est écrite entre des apostrophes (**'**). Matlab ne fait pas la différence entre une chaîne de caractères écrite entre deux apostrophes et un tableau de caractères. En d'autres termes, Matlab considère chaque chaîne de caractère comme un vecteur ligne dont chaque élément contient un caractère.

Exemple : Initialiser des chaînes de caractères

```
1 >> 'test'
2
3 ans =
4
5 test
6
7 >> ['t', 'e', 's', 't']
8
9 ans =
10
11 test
```

5. Calculs élémentaires

L'utilisation la plus basique de Matlab consiste à utiliser l'interpréteur de commandes comme une calculatrice. Les opérations arithmétiques de base utilisées par Matlab sont les suivantes :

- l'addition (+) ;
- la soustraction (-) ;
- la multiplication (*) ;
- la division (/) ;
- la puissance (^).

Dans le cas d'une expression arithmétique compliquée (qui comporte plusieurs opérations) la priorité la plus élevée est donnée à l'opération de puissance. La division et la multiplication sont de leur côté plus prioritaire que l'addition et la soustraction. Notez aussi que les parenthèses peuvent s'utiliser de manière classique pour tenir compte des priorités des opérations.

Exemple : Effectuer des calculs élémentaires

```
1 >> 3 + 2 * (3 + 2)
2
3 ans =
4
5     13
6
7 >> (3 * 2) ^ 3 + 2
8
```

```
ans =

   218
```

Remarque : La variable spéciale "ans"

Le résultat est affecté automatiquement à une variable spéciale appelée ans. Celle-ci pourrait être utilisée dans un autre calcul comme nous allons le voir dans les prochains exemples.

5.1. Fonctions classiques

Dans Matlab, de nombreuses fonctions de calcul sont prédéfinies :

- `sqrt()` : calcule la racine carrée.
- `cos()`, `sin()`, `tan()` et `cotg()` : pour les fonctions trigonométriques de base.
- `acos()`, `asin()` et `atan()` : pour leurs réciproques.
- `cosh()`, `sinh()`, `tanh()`, `acosh()`, `asinh()` et `atanh()` : pour les fonctions trigonométriques hyperboliques .
- `log()` : pour le logarithme népérien (ln)
- `log10()` : pour le logarithme en base 10.
- `log2()` : pour le logarithme en base 2.
- `exp()` : pour la fonction exponentielle.
- `floor()` : pour l'arrondi vers l'entier inférieur.
- `ceil()` : pour l'arrondi vers l'entier supérieur.
- `fix()` : pour la troncature (arrondit à l'entier inférieur en valeur absolue).
- `round()` : pour l'arrondi à l'entier le plus proche (0,5 → 1).
- `abs()` : pour la valeur absolue.
- `rand()` : pour avoir un nombre aléatoire entre 0 et 1, suivant une loi uniforme.

Exemple : Utiliser des fonctions

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

ans =

    13

>> (3 * 2) ^ 3 + 2

ans =

   218
```

5.2. Format d'affichage

Par défaut, Matlab affiche les résultats en format `short`. On peut modifier le format d'affichage n utilisant la commande `format` :

- `format short` : valeur par défaut, notation fixe à 4 décimales.
- `format long` : notation fixe à 7 décimales pour un réel au format simple, 14 ou 15 décimales pour un réel au format double.
- `format short e` ou `format long e` : notation scientifique exponentielle à virgule flottante.

6. Variables et affectation

Dans la plupart des langages, et Matlab n'échappe pas à la règle, on appelle variable un objet correspondant à une zone mémoire, et identifié par un nom. Dans cette section, nous verrons comment créer une variable, lui donner une valeur et l'utiliser dans Matlab. Les variables dans Matlab ne doivent pas être déclarées, elles sont créées systématiquement à la première opération d'affectation (=). Le type et la dimension de la variable sont déterminés de manière automatique à partir de l'expression mathématique ou de la valeur affectée à la variable.

Une variable est désignée par un identificateur qui est formé d'une combinaison de lettres et de chiffres. Le premier caractère de l'identificateur doit nécessairement être une lettre. Cet identificateur doit être unique pour distinguer les différentes variables utilisées par le programme.

Cette notion (variable) est fondamentale parce qu'elle permet de manipuler des valeurs (numériques ou non) et est à la base de la plupart des calculs et traitements que l'on peut réaliser avec un langage de programmation.

Exemple : Utiliser des variables

```
>> x = 2

x =

     2

>> y = x * 2

y =

     4
```

Attention : Majuscules et minuscules dans un identificateur

MATLAB différencie majuscules et minuscules, c'est-à-dire que la variable avec l'identifiant `x` est différente de la variable `X` comme le montre l'exemple suivant.

Exemple : Différence entre majuscules et minuscules

```
>> x = 2

x =

     2

>> X = 3

X =

     3

>> y = x + 2

y =

     3

```

```
>> z = Y + 5
??? Undefined function or variable 'Y'.
```

6.1. Quelques variables spéciales

Certains identificateurs sont réservés par Matlab pour conserver quelques valeurs importantes, nous présentons ici certaines de ces variables :

- La variable `ans` : cette variable garde la valeur de la dernière expression non-affecté à une variable.
- La variable `pi` : cette variable comporte la valeur de π (= 3.146...).
- La variable `i` : cette variable garde le nombre imaginaire $\sqrt{-1}$

Exemple : Utiliser des variables

```
>> x = i ^ 2

x =

    -1

>> y = pi * 2

y =

    6.2832
```

7. Manipuler des tableaux

Matlab est un outil de calcul matriciel, alors il considère chaque nombre comme une matrice de taille (`1 x 1`), un vecteur ligne de `n` éléments comme une matrice de taille (`1 x n`) et un vecteur colonne de `m` éléments comme une matrice de taille (`m x 1`). Matlab est particulièrement adapté aux calculs d'algèbre linéaire.

Pour deux matrices `x` et `y`, si le nombre de lignes de `y` est égale au nombre de colonnes de `x`, alors l'opération `x * y` effectue une multiplication matricielle de `x` par `y`. L'opérateur `^` peut également être utilisé pour multiplier une matrice carrée par elle même.

La compréhension de la gestion des matrices (tableaux à deux dimensions) par Matlab est une étape essentielle dans la prise en main de ce langage. En effet, Matlab est avant tout un logiciel de calcul matriciel et donc, maîtriser la manipulation des matrices, permet d'améliorer les performances des programmes par un codage propre et efficace.

Exemple : Opération de multiplication et matrices

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

a =

     1     2
     3     4

>> b = [5 6; 7 8]

b =

```

```
     5     6
     7     8

>> a * b

ans =

    19    22
    43    50

>> a .* b

ans =

     5    12
    21    32
```

L'opérateur "+" est utilisé pour faire l'addition de deux matrices qui ont la même dimension, ou pour additionner un scalaire à tous les composants d'une matrice. L'opérateur "-" peut être utilisé de la même façon pour effectuer des soustractions.

Exemple : Addition et soustraction des matrices

```
>> a = [1 2; 3 4];
>> b = [5 6; 7 8];
>> b + a

ans =

     6     8
    10    12

>> b - a

ans =

     4     4
     4     4

>> b - 1

ans =

     4     5
     6     7
```

Conseil : Le point-virgule

Comme le montre l'exemple précédent, seulement les valeurs des expressions qui ne sont pas suivies d'un point-virgule sont affichées. Lorsqu'on met un point-virgule à la fin d'une instruction le résultat de son évaluation n'est pas affiché (l'instruction est exécutée d'une façon normale, c'est seulement l'affichage qui n'est pas effectué). Vu que dans certains cas l'affichage des résultats peut s'avérer fastidieux et inutile (comme le résultat de multiplication de deux grandes matrices) nous recommandons de mettre un point-virgule à la fin de ces commandes pour indiquer à Matlab qu'il ne doit pas afficher les résultats.

7.1. Accéder aux éléments d'un tableau

Les éléments d'un tableau peuvent être manipulés grâce à leurs indices. Soit `T` un tableau de dimension *(L x C)*, et soit *l* et *c* deux nombres entiers positifs tel que $l < L$ et $c < C$. La commande `T(l, c)` nous permet d'accéder à l'élément à la c_{ieme} colonne de la l_{ieme} ligne du tableau `T`.

Pour un vecteur ligne (ou colonne) `v` de taille *n*, et pour un nombre $i < n$ la commande `v(i)` permet d'accéder à l'élément dans la ligne (ou la colonne) *i* du vecteur *v*.

Exemple : Accéder aux éléments d'un tableau

```
>> A = 2:2:10

A =

     2     4     6     8    10

>> A(3)

ans =

     6

>> k = 4

k =

     4

>> A(k)

ans =

     8

>> B = [2 4; 6 8]

B =

     2     4
     6     8

>> B(1, 2)

ans =

     4

>> B(2, 2)

ans =

     8
```

7.2. Extraire un sous-tableau

Soit `T` un tableau de dimension *(L x C)*, et soit *l1 < l2 < L*, *c1 < c2 < C*. La commande `T(:, c1)` extrait la colonne *c1* du tableau `T`, alors que la commande `T(l1, :)` extrait la ligne *l1*. La commande `T(l1:l2, c1:c2)` extrait un sous-tableau formé par les éléments de `T` dont l'indice de ligne appartient à `[l1, l2]` et l'indice de colonne appartient à `[c1, c2]`. L'identificateur `end` peut être utilisé pour accéder à la dernière ligne ou la dernière colonne d'un tableau.

Exemple : Extraction des sous-tableaux

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

A =

     1     2     3     4
     2     3     4     5
     3     4     5     6
     4     5     6     7

>> A(:, 3)

ans =

     3
     4
     5
     6

>> A(2, :)

ans =

     2     3     4     5

>> A(3:end, 2:3)

ans =

     4     5
     5     6
```

7.3. Sous-matrices spéciales

Pour extraire les éléments de la diagonale principale d'une matrice, on utilise la fonction `diag`. La même fonction peut être paramétrée pour extraire les éléments de la $k^{ième}$ diagonale. La fonction `diag` retourne un vecteur-colonne formé des éléments de la diagonale qu'elle extrait. Selon le type de son argument, la fonction `diag` se comporte de deux façon différentes.

- si le paramètre de la fonction `diag` est une matrice elle retourne un vecteur-colonne contenant les éléments de la diagonale de son paramètre.
- si le paramètre de la fonction `diag` est un vecteur, elle retourne une matrice diagonale dont le vecteur diagonal est composé à partir de son paramètre.

La fonction `triu` (`tril`) extrait les éléments sur et au dessus (dessous) de la diagonale. Comme `diag`, les fonctions `triu` et `tril` peuvent prendre un deuxième paramètre pour extraire les éléments à partir de la diagonale `k`. Notez que les fonctions `diag`, `triu` et `tril` peuvent prendre comme argument une matrice qui n'est pas carrée.

Exemple

```
>> A = [1 2 3; 4 5 6; 7 8 9]

A =

     1     2     3
     4     5     6
     7     8     9

>> triu(A)

ans =

     1     2     3
     0     5     6
     0     0     9

>> diag(A, 1)

ans =

     2
     6

>> diag(diag(A))

ans =

     1     0     0
     0     5     0
     0     0     9
```

7.4. Concaténer des tableaux

Nous utilisons l'opérateur `[]` pour concaténer deux tableaux (ou plus). on sépare les tableaux par un espace ou virgule pour faire une concaténation en colonnes (les tableaux doivent avoir le même nombre de lignes), et on les sépare par des points-virgules pour faire une concaténation en lignes (les tableaux doivent avoir le même nombre de colonnes).

Pour concaténer un tableau à lui-même (construire un tableau en répétant le même bloque), la fonction `repmat` peut-être utilisée.

Il peut arriver que l'on souhaite obtenir la réplication d'une matrice de façon entrelacée en répétant un certaine nombre de fois chaque colonne et/ou chaque ligne. Dans ce cas, il faut utiliser la fonction `kron`.

Exemple

```
>> A = [1 2];
```

```
>> B = [3 4; 5 6];
>> [A; B]

ans =

     1     2
     3     4
     5     6

>> repmat(A, 2, 2)

ans =

     1     2     1     2
     1     2     1     2

>> kron(B, ones(2, 3))

ans =

     3     3     3     4     4     4
     3     3     3     4     4     4
     5     5     5     6     6     6
     5     5     5     6     6     6

>> kron(B, [1 2; 3 4]) % multiplier chaque valeur par des coéfficients

ans =

     3     6     4     8
     9    12    12    16
     5    10     6    12
    15    20    18    24

```

7.5. L'indice "end"

`end` est un mot-clé de Matlab qui peut être employé comme opérateur d'indexage. Dans le cas d'un vecteur, le dernier élément est retourné, alors que dans le cas d'une matrice (ou un tableau à plusieurs dimensions) il retourne le dernier élément d'une de ces dimensions.

Exemple : Utilisation de l'indice "end"

```
>> M = [4 7 2]

M =

     4     7     2

>> M(end)

ans =

     2

>> X = [8 1 ; 2 7]

```

```
X =

     8     1
     2     7

>> X(end,1) % Dernier élément de la première colonne

ans =

     2

>> X(end,2) % Dernier élément de la seconde colonne

ans =

     7

>> X(1,end) % Dernier élément de la première ligne

ans =

     1

>> X(2,end) % Dernier élément de la seconde ligne

ans =

     7

>> X(end,end) % Dernier élément de la matrice (indexage classique)

ans =

     7

>> X(end) % Dernier élément de la matrice (indexage linéaire)

ans =

     7

>> X(end,1:end) % Tous les éléments de la derniere ligne

ans =

     2     7
```

⚠ *Attention : end, un mot clé à usage multiple*

En effet, le mot clé `end` peut également être utilisé pour marquer la fin d'une structure de contrôle dans un script ou une fonction. plus de détails serons donnés dans le troisième chapitre.

7.6. Initialiser des matrices spéciales

Les fonctions suivantes construisent des matrices usuelles: identité, Hilbert ...

Fonction	Matrice à générer
`eye`	prend comme paramètre un entier positif *n* et retourne une matrice identité carrée d'ordre *n*.
`ones`	prend comme paramètre un entier positif *n* et retourne une matrice carrée *M* d'ordre *n* tel que $M_{i,j} = 1$.
`zeros`	prend comme paramètre un entier positif *n* et retourne une matrice carrée *M* d'ordre n tel que $M_{i,j} = 0$.
`rand`	prend comme paramètre un entier positif *n* et génère une matrice carrée aléatoire d'ordre *n* (comporte des valeurs aléatoires entre 0 et 1).
`vander`	prend comme paramètre un vecteur (ligne ou colonne) `s` et retourne une matrice de Vandermonde de taille `length(s)` engendrée par le vecteur `s`.
`hilb`	prend comme paramètre un entier positif `n` et génère une matrice `M` de Hilbert d'ordre *n* ($m_{i,j} = 1 / (i + j - 1)$).
`magic`	prend comme paramètre un entier *n* (> 3) et génère un carré magique d'ordre *n* (construit à partir d'entiers entre 1 et n^2, tel que les sommes des lignes, des colonnes et des deux vecteurs diagonaux de la matrice générée sont égales).
`pascal`	prend comme paramètre un entier positif *n* et génère une matrice *M* de Pascal d'ordre *n*. Pour tout $i, j \leq n$: $M_{i,1} = 1$, $M_{1,j} = 1$ et $M_{i,j} = M_{i-1,j} + M_{i-1,j-1}$.
`wilkinson`	prend comme paramètre un entier positif *n* et génère une matrice de Wilkinson d'ordre *n*.
`hadamard`	prend comme paramètre un entier $n = 2k$ et génère une matrice *M* de Hadamard d'ordre *n* (tous les éléments de *M* sont soit 1, soit -1 et `M' * M = n * eye(n)`).

Tableau 1 : Quelques matrices spéciales

7.7. Fonctions sur les tableaux

Les fonctions présentées dans cette section effectuent des opérations arithmétiques itérativement sur les éléments d'un vecteur (ligne ou colonne). Appliquées à une matrice, elles effectuent les mêmes opérations sur chaque colonne et retournent un vecteur ligne comme résultat.

Fonction	Opération à effectuer
`length`	Retourne la taille d'un vecteur ligne ou un vecteur colonne.
`size`	Une fonction qui peut être utilisée pour retourner le nombre de lignes ou/et le nombre de colonnes d'une matrice selon les arguments qui sont utilisés : - `size(X, 1)` retourne le nombre de lignes de la matrice `X`. - `size(X, 2)` retourne le nombre de colonnes de la matrice `X`.

	- `size(X)` retourne un tableau contenant deux éléments, le premier élément représente le nombre de ligne de `X`, alors que le deuxième élément représente le nombre de colonnes `X`.

`sum`	- Pour un vecteur *T*, `sum(T)` retourne la somme des éléments de *T*.
	- Pour une matrice *M* :
	  - `sum(M, 1)` retourne un vecteur ligne contenant la somme de chaque colonne de *M*.
	  - `sum(M, 2)` retourne un vecteur colonne contenant la somme de chaque ligne de *M*.
	  - `sum(M)` : équivalent à `sum(M, 1)`.

`prod`	- Pour un vecteur *T*, `prod(T)` retourne le produit des éléments de *T*.
	- Pour une matrice *M* :
	  - `prod(M, 1)` retourne un vecteur ligne contenant le produit des éléments de chaque colonne de *M*.
	  - `prod(M, 2)` retourne un vecteur colonne contenant le produit des éléments de chaque ligne de *M*.
	  - `prod(M)` : équivalent à `prod(M, 1)`.

`max`	- Pour un vecteur *T*, `max(T)` retourne la maximum des éléments de *T*.
	- Pour une matrice *M* :
	  - `max(M, 1)` retourne un vecteur ligne contenant le maximum de chaque colonne de *M*.
	  - `max(M, 2)` retourne un vecteur colonne contenant le maximum de chaque ligne de *M*.
	  - `max(M)` : équivalent à `max(M, 1)`.

`min`	Usage similaire à la fonction `max`, mais retourne le(s) minimume(s).
`mean`	Usage similaire à la fonction `sum`, mais retourne la/les moyenne(s).
`cov`	- Appliquée à un vecteur, cette fonction retourne la variance de ces éléments. - Appliquée à une matrice *M*, elle retourne une autre matrice symétrique de covariance des éléments de *M* (colonne à colonne). Pour obtenir la variance de chaque colonne, il suffit d'extraire le vecteur diagonal du résultat.
`abs`	Retourne une matrice contenant les valeur absolues de son argument.
`norm`	Calcul les normes vectorielles ou matricielles usuelles ($\|\|.\|\|_1$, $\|\|.\|\|_2$, $\|\|.\|\|_\infty$ ... etc.).
`sort`	- Appliquée à un vecteur, elle ordonne ces éléments par ordre croissant. - Appliquée à une matrice *M*, selon le deuxième paramètre, elle ordonne les lignes ou bien les colonnes de la matrice. - `sort(M, 1)` : Ordonne toutes les colonnes de *M*.

	- `sort(M, 2)` : Ordonne toutes les lignes de *M*.

Tableau 2 : Fonctions sur les tableaux

8. Espace de travail

L'ensemble des variables initialisées par l'utilisateur constituent l'espace de travail ou la session en cours. Le contenu de cet espace de travail va changer tout au long du déroulement de la session (exécution des commandes ou scripts). Plusieurs commandes ou fonctions permettent de gérer l'espace de travail d'une façon efficace.

8.1. Informations sur l'espace de travail

Pour obtenir des informations détaillées sur des variables dans l'espace de travail, on utilise les instructions suivantes :

- La commande `who` : Afficher la liste des variables dans l'espace de travail (les identifiants seulement sans les détails).
- La commande `whos` : Affichage plus détaillé des variables dans l'espace de travail (en donnant des informations sur le type, la dimension et l'espace mémoire occupé par chaque variable).

Pour supprimer toutes les variables de l'espace de travail, la commande `clear` peut être utilisée. Pour supprimer une variable `x` spécifique, il faut exécuter la commande `clear x`.

Exemple : Utilisation des commandes : who, whos et clear

```
>> x = 2;
>> y = [3 4 5];
>> who

Your variables are:

x  y

>> whos
  Name      Size            Bytes  Class     Attributes

  x         1x1                 8  double
  y         1x3                24  double

>> clear x
>> whos
  Name      Size            Bytes  Class     Attributes

  x         1x1                 8  double

```

8.2. Les commandes save et load

La commande `save` permet d'enregistrer toutes les variables dans un fichier `matlab.mat` (appelé aussi mat-file). Pour spécifier le nom de fichier à enregistrer, il faut utiliser la commande `save nomfichier`. Pour enregistrer quelques variables seulement, il faut écrire la commande suivante :

`save nomFichier var1 var2 ... varn`.

La commande `load` permet d'ajouter le contenu d'un fichier ".mat" à l'espace de travail courant en utilisant la syntaxe suivante :

`load nomFichier`.

9. Exercices

9.1. Exercice : Créer un vecteur

Question

[solution n°1 p.74*]*

Écrire la commande Matlab qui permet d'initialiser un vecteur ligne contenant les valeurs suivantes (1, 2, 3, 4, 5).

9.2. Exercice : Produit scalaire

Question

[solution n°2 p.74*]*

- Écrire la commande Matlab qui permet d'initialiser un vecteur ligne contenant 5 valeurs aléatoires et mettre le résultat dans la variable `x`.
- Écrire la commande Matlab qui permet d'initialiser un vecteur colonne contenant 5 valeurs aléatoires et mettre le résultat dans la variable `y`.
- Écrire la commande Matlab permettant d'effectuer le produit scalaire des deux vecteurs `x` et `y`.

9.3. Exercice : Résoudre un système linéaire

Question

[solution n°3 p.74*]*

Utiliser Matlab pour résoudre le système linéaire suivant

$$\begin{cases} 2x_1 - 2x_2 + 4x_3 &= 6 \\ x_1 + x_3 &= 7 \\ 4x_1 + x_2 - 5x_3 &= 8 \end{cases}$$

Indice :

Utiliser la représentation matricielle d'un système linéaire

9.4. Exercice : Matrice magique

Question

[solution n°4 p.74*]*

1. Créer une matrice magique de taille 5.
2. Calculer la somme de chaque colonne de la matrice magique créée.
3. Calculer la somme de chaque ligne de la même matrice.
4. Calculer la somme des deux diagonaux.

9.5. Exercice : Fonctions trigonométriques

Question

[solution n°5 p.74*]*

- Définir un vecteur `x` contenant les valeurs [π/6, π/3, π/2].
- Calculer `y1 = sin(x)` et `y2 = cos(x)`.
- Calculer `tan(x)` en utilisant `y1` et `y2`.

9.6. Exercice : Manipuler les vecteurs

Question

[solution n°6 p.74*]*

- Écrire la commande Matlab qui permet de définir le vecteur ligne `x` contenant les valeurs [1, 2 ... 49, 50]
- Définir le vecteur `y` contenant les 5 premières valeurs du vecteur `x`.
- Définir le vecteur `z` contenant les 5 dernières valeurs du vecteur `x`.
- Définir le vecteur `n` contenant les éléments à indice pair du vecteur `x`.

9.7. Exercice : Manipuler les matrices

Question

[solution n°7 p.74*]*

Écrire la commande Matlab qui permet d'initialiser un vecteur ligne contenant les valeurs suivantes (1, 2, 3, 4, 5).

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

Enlever de la matrice B les deux premières colonnes et la dernière ligne.

9.8. Exercice : Complément sur les matrices

Question

[solution n°8 p.75*]*

1. Créer une matrice aléatoire carrée d'ordre 8, contenant des valeurs entre 1 et 10.
2. Utiliser la fonction `tril` pour extraire la partie triangulaire inférieure de la matrice générée.
3. Utiliser les deux fonctions `diag` et `prod` pour calculer le déterminant de la matrice que vous venez d'extraire.

Indice :

Pour calculer le déterminant d'une matrice triangulaire, il suffit de multiplier ses éléments diagonaux.