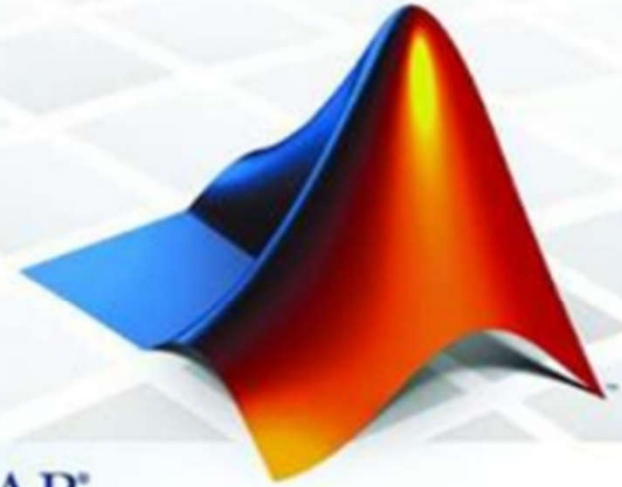
ENSAH

A Training for Engineers

introducing

MATLAB



MATLAB

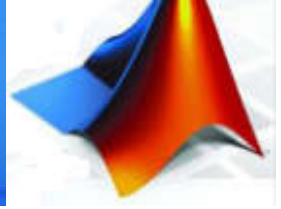
ENSAH – Classes préparatoires 2ème année – 2019/2020

Pr. Routaib Hayat

routaib.hayat@gmail.com

Plan

C’est quoi la programmation ?



Définition de « la programmation »

Un ordinateur ne sait faire que certaines opérations très basiques :

- ✓Additionner, soustraire, multiplier, diviser des valeurs numériques
- ✓Déplacer des données d'un endroit de sa mémoire à un autre endroit
- ✓Sauter d'une ligne de code à une autre si une condition est vraie

La programmation consiste à combiner ces instructions qui agissent sur des données afin de réaliser une tache spécifique à l'aide d'un ordinateur …

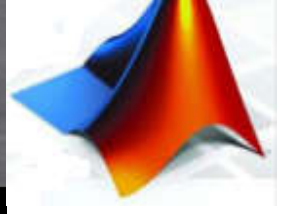
Ces instructions ont une tache précises et elles forment par la suite un ensemble de logiciel

« La programmation »

Mais lorsqu'on parle de la programmation on intervient **un objet** pour décrire d'une manière générale les concepts que l'on manipulent. De la même manière lorsqu'on parle des variables en mathématiques.

L'objet le plus commun dans le langage **Matlab** est **la matrice.**

Présentation de Matlab



Qu'est-ce que Matlab ?

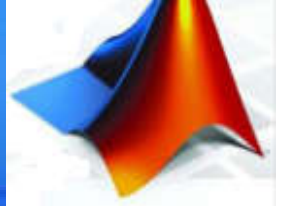
Matlab (MATrix LABoratory) est un langage de programmation orienté calcul scientifique, pensé pour rendre le calcul matriciel simple à programmer et efficace en même temps.

Matlab est un logiciel (payant) proposant une interface graphique vers un éditeur de code en Matlab, et un outil de debugging pour exécuter des programmes en mode pas à pas.

Il propose différents frameworks (payants, appelés toolbox), proposant des fonctionnalités très avancées permettant de réaliser facilement des taches complexes (toolbox : équation différentielle, toolbox aérospatial)

Présentation de l'interface

L'interface de Matlab se compose de plusieurs zones :



Présentation de l'interface

Chaque zone possède un objectif précis :

- **Le menu :** regroupe des commandes de base de Matlab comme enregistrer, afficher, préférences etc...

- **L'explorateur de fichier :** permet de visualiser les fichiers scripts et de les ouvrir pour les éditer

- **La zone de commande :** permet d'écrire des commandes et de visualiser leur résultat

- **La zone des variables :** permet de visualiser toutes les variables en mémoire à l'instant présent (leur nom ainsi que leur contenu)

- **L'historique :** permet de visualiser l'historique des commandes précédemment exécutées.

Présentation de l'interface

1. Zone de commandes :

C'est le terminal dans lequel on doit taper les commandes et sur lequel on verra l' affichage des résultats. Une ligne commence toujours par >>. Essayez la commande suivante :

```
Command Window
>> 2+3

ans =

     5
```

Ici Matlab définit une variable ans, lorsque l'on effectue un calcul il l'affiche sur le terminal

Et puis comparez par :

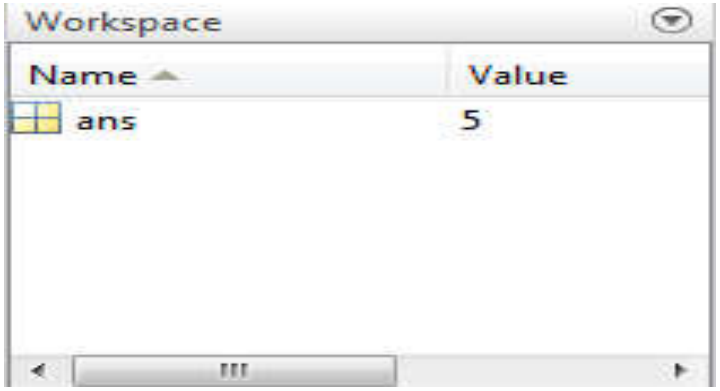
```
Command Window
>> 2+3;
>> whos()
  Name        Size              Bytes  Class     Attributes

  ans         1x1                   8  double
```

Il définit une matrice de taille 1x1. Une commande utile whos() permet de la décrire

Présentation de l'interface

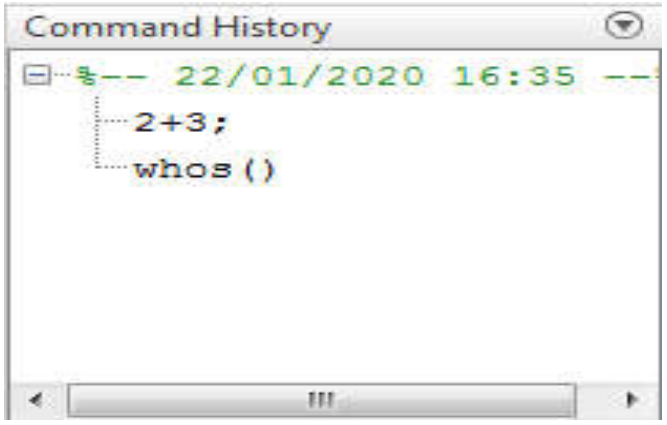
2. Workspace :

Dans cette fenêtre, on obtient la liste des variables connues par Matlab . Il est possible de double-cliquer sur une variable pour l'afficher. Un clic-droit sur les variables offre de nombreuses options telles que : Copiez, Collez, Supprimez....



3. Commad History :

Il est possible de cliquer sur une commande dans cette fenêtre pour l'exécuter a nouveau.



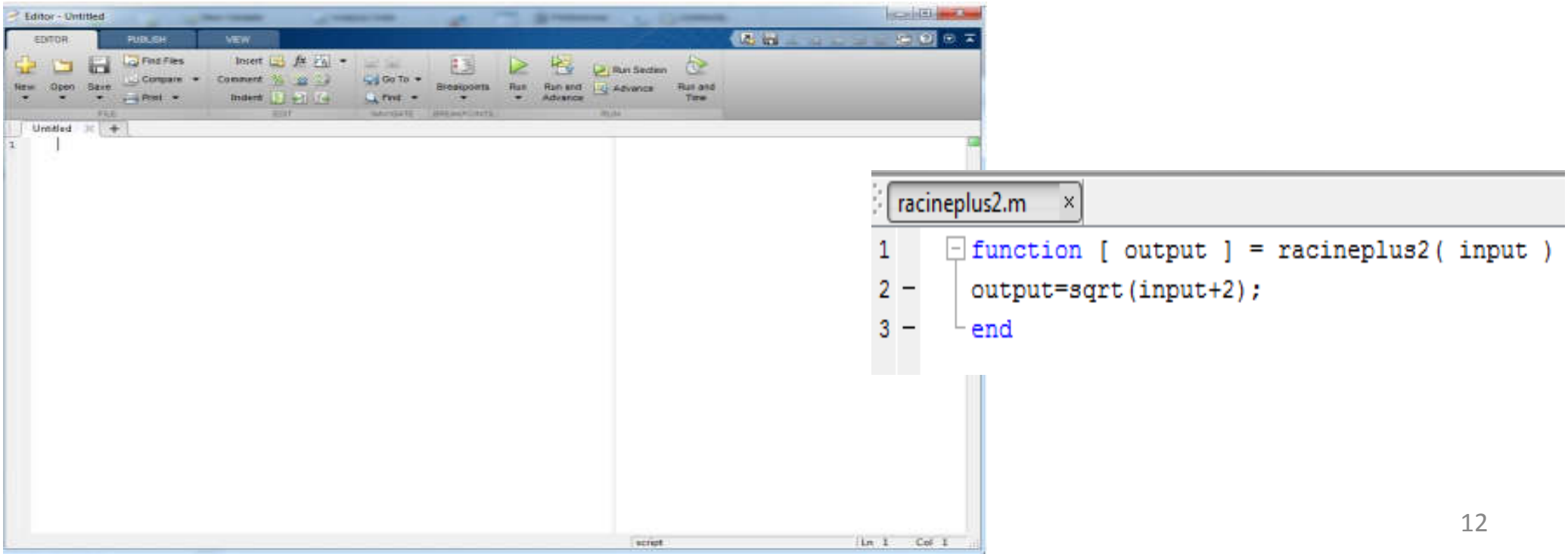
N.B : Lorsqu'on effectue une analyse de données sur les résultats d'une expérience il est **essentiel** de conserver une trace de toutes les opérations qui ont été réalisées.

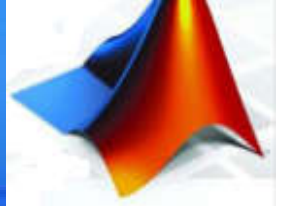
→ C'est la base de la traçabilité et de la reproductivité des résultats scientifiques.

Présentation de l'interface

4. Editeur Matlab :

Lorsqu'on réalise une tache sous Matlab, il est très souvent possible de le faire en utilisant uniquement la Command Window. **Mais** lorsque cette tache devient plus complexe **(plusieurs dizaines de ligne de code)** ou que l'on souhaite pouvoir la transmettre à quelqu'un d'autre simplement, on utilise la fenêtre Editor pour créer les fichiers **.m**





Présentation de l'interface

5. Obtenir de l'aide :

L'aide est essentielle lorsque l'on programme avec un langage de haut-niveau comme Matlab, où le nombre de fonctions est très important et la syntaxe est parfois complexe.

Pour accéder à l'aide on peut au choix sélectionner une fonction et presser F1, taper **help FunctionName**

Les outils de base

Outils de base

Le principe de base de Matlab est de considérer la plupart des objets comme des matrices. Ainsi les opérations usuelles +, -, *, / doivent se comprendre comme des opérations matricielles. la section suivante consacrera à ces opérations.

Dans un premier temps, nous allons regarder ce qu'ils se passent pour les matrices 1x1 (c'est à dire un seul élément), puis pour les matrices 1xn ou nx1 (c'est à dire des vecteurs ligne ou colonne)

Symbole	Description	Exemple
+ - * /	Les opérations de base en mathématiques (addition, soustraction, multiplication et division)	7+9 3/4
pi	La constante Pi	pi/3
cos sin tan	Les fonctions trigonométriques usuelles	cos(3*pi/2)
log exp	Le logarithme népérien et l'exponentielle	exp(3)
sqrt	La racine carrée	sqrt(5)
^	La puissance	4^7

Outils de base

1. Types de variables:

Il existe cinq grands types de variables sous Matlab : les entiers, les réels, les complexes, les chaines de caractères et le type logique. Définissons une variable de chaque type :

```
>> a = 1.3; b = 3+i; c = 'bonjour';
>> d1 = true(1==1); d2 = logical(1);
>> e = int8(2);
```

a représente un réel, b un complexe, c une chaine de caractères, d1 et d2 sont deux manières de définir une variable logique (VRAI dans le cas présent) et e est un entier codé sur 8 bits. On peut alors vérifier le type de ces différentes variable en utilisant la fonction whos :

```
>> whos
  Name        Size              Bytes  Class     Attributes

  a           1x1                   8  double
  b           1x1                  16  double    complex
  c           1x7                  14  char
  d1          1x1                   1  logical
  d2          1x1                   1  logical
  e           1x1                   1  int8
```

NB : On peut définir la partie imaginaire d'un complexe en utilisant au choix i ou j

Outils de base

2. Arithmétique et opérations sur les scalaires :

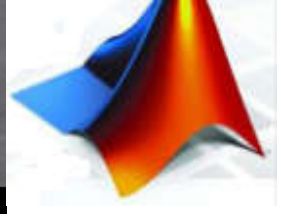
Nous allons nous intéresser aux opérations mathématiques de bases avec des matrices 1x1, c'est à dire des nombres.

```
Command Window
>> x=1+1

x =

     2
```

On peut également travailler avec des variables définies par l'utilisateur :

```
>> x = 2; y = 1.5 ;
somme = x+y
difference = x-y
produit = x*y
division = x/y

somme = 3.5000
difference = 0.5000
produit =  3
division = 1.3333
```

NB : Avant de commencer, je conseille d'utiliser les trois commandes suivantes : clc pour nettoyer l'écran, clear all pour supprimer toutes les variables créées auparavant, et close all pour fermer toutes les fenêtres inutiles.



Outils de base

2. Arithmétique et opérations sur les scalaires :

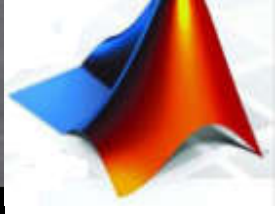
On peut également utiliser les fonctions trigonométriques, puissance, logarithmiques..etc

```
>> x = 2; y = pi;
cos(y)
exp(x)
sqrt(x)

ans =  -1
ans = 7.3891
ans = 1.4142
```

Outils de base

2. Arithmétique et opérations sur les scalaires :

Fonction incorporée sous Matlab	Définition
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 carrée de x
abs(x)	Valeur absolue de x
sign(x)	1 Si $x > 0$ et 0 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



Outils de base

2. Arithmétique et opérations sur les scalaires :

Fonction incorporée sous Matlab	Définition
asinh(x)	Sinus hyperbolique inverse de x
round(x)	entier le plus proche de x
floor(x)	Arrondi par défaut de x
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
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

Outils de base

3. Les formats d'affichage des réels:

MATLAB utilise toujours les nombres réels (double precision) pour faire les calcules, ce qui permet d'obtenir une précision de calcul allant jusqu'aux 16 chiffres significatifs.

Mais il faut noter les points suivants :

- ✓ Le résultat d'une opération de calcul est par défaut affichée avec quatre chiffres après la virgule.
- ✓ Pour afficher davantage de chiffres on utilise la commande *format long* (14 chiffres après la virgule).
- ✓ Pour retourner à l'affichage par défaut, utiliser la commande *format short.*
- ✓ Pour afficher uniquement 02 chiffres après la virgule, utiliser la commande *format bank.*
- ✓ Pour afficher les nombres sous forme d'une ration, utiliser la commande *format rat.*

Outils de base

3. Les formats d'affichage des réels:

La commande	*Signification*
format short	Affiche les nombres avec **04 chiffres** après la virgule
format long	Affiche les nombres avec **15 chiffres** après la virgule
format bank	Affiche les nombres avec **02 chiffres** après la virgule
format rat	Affiche les nombres sous forme **d'une ration (a/b)**

Outils de base

3. Les formats d'affichage des réels:

Command Window

```
>> 3/9

ans =

    0.3333

>> format short
>> 3/9

ans =

    0.3333

>> format long
>> 3/9

ans =

   0.333333333333333
```

Command Window

```
>> format bank
>> 3/9

ans =

                0.33
```

Command Window

```
>> format rat
>> 7.2*3.1

ans =

       558/25

>> 1.5*0.4

ans =

          3/5
```

Vecteurs

NB : Pour grouper des éléments types différents par exemple des caractères et des réels, on utilisera **les structures.**

1. Définir un vecteur :

Un vecteur sous Matlab est une collection d'éléments du même type. Un vecteur pourra représenter des valeurs expérimentales ou bien des valeurs discrétisées d'une fonction continue.

La méthode la plus simple pour définir un vecteur est de donner sa description explicite à l'aide de la commande [ ] :

```
vec = [1 2 4 7 9 2.3]

vec =
    1.0000    2.0000    4.0000    7.0000    9.0000    2.3000
```

```
col = [1 ; 2 ; 4 ; 7]

col =
     1
     2
     4
     7
```

Vecteurs

1. Définir un vecteur :

On peut concaténer deux vecteurs :

```
vec1 = [1 3 5];
vec2 = [9 10 11];
vec = [vec1 vec2]

vec =
     1     3     5     9    10    11
```

```
length(vec)

ans = 3
```

on peut également prendre la transposée pour passer d'une ligne à une colonne ou réciproquement :

```
vec1 = [1 3 5];
vec = vec1'

vec =
     1
     3
     5
```

Lorsqu'on veut afficher le graph d'une fonction **f** telle que **y = f(x)** sous Matlab, il faut définir deux vecteurs : **un pour le vecteur x**, l'autre **pour le vecteur y = f(x)**. Matlab ne fait pas du calcul formel, il faut donc discrétiser l'axe des x.

On utilise pour ce faire la fonction **linspace(a,b,n)** Cette commande génère un vecteur ligne de **n** éléments espacés linéairement entre **a** et **b**. Par défaut **n = 100**.

Vecteurs

1. Définir un vecteur :

Une autre méthode pour générer des vecteurs espacés linéairement consiste à utiliser **[a: s: b]**. On crée alors un vecteur entre **a** et **b** avec un espacement **s** :

```
vec=[1:2:10]

vec =     1     3     5     7     9
```

Il existe enfin des vecteurs spéciaux prédéfinis dans Matlab :

Vecteurs spéciaux	Définition
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**

Vecteurs

2. Manipuler un vecteur :

Le $k^{ème}$ élément d'un vecteur **vec** peut être affiché grâce à la commande **vec(k)**. ***k*** doit être un entier sinon Matlab retournera une erreur :

```
Command Window
>> vec = linspace(2,10,11)

vec =

  Columns 1 through 6

    2.0000    2.8000    3.6000    4.4000    5.2000    6.0000

  Columns 7 through 11

    6.8000    7.6000    8.4000    9.2000   10.0000

>> vec(5)

ans =

    5.2000
```

On peut également utiliser des vecteurs d'indices pour extraire un sous-vecteur :

```
Command Window
>> vec = linspace(1,89,9)

vec =

     1    12    23    34    45    56    67    78    89

>> vec(3:6)

ans =

    23    34    45    56

>> subvec=[1 3 5];
>>  vec(subvec)

ans =

     1    23    45
```

Vecteurs

3. Opérations vectorielles :

Command Window

```
>> vec = [1 3 5 6]; vec2 = [10 20 30 40];
>> vec+vec2

ans =

    11    23    35    46

>> vec-vec2

ans =

    -9   -17   -25   -34

>> vec.*vec2

ans =

    10    60   150   240

>> vec2./vec

ans =

   10.0000    6.6667    6.0000    6.6667
```

Command Window

```
>> vec = linspace(1,10,10);
>> out=sqrt(vec)

out =

  Columns 1 through 6

    1.0000    1.4142    1.7321    2.0000    2.2361    2.4495

  Columns 7 through 10

    2.6458    2.8284    3.0000    3.1623

>> out2=cos(vec)

out2 =

  Columns 1 through 6

    0.5403   -0.4161   -0.9900   -0.6536    0.2837    0.9602

  Columns 7 through 10

    0.7539   -0.1455   -0.9111   -0.8391
```

Vecteurs

4. Commandes aux vecteurs :

N.B: Ces commandes s'appliquent aussi aux matrices. Dans ce cas la commande porte sur chaque vecteur colonne de la matrice.

Il existe aussi des commandes qui sont propres aux vecteurs

Commandes	Définition
sum(x)	Somme des éléments du vecteur **x**
prod(x)	Produit des éléments du vecteur **x**
Min(x)	Plus petit élément du vecteur **x**
max(x)	Plus grand élément du vecteur **x**
Mean(x)	Moyenne des éléments du vecteur **x**
Sort(x)	Ordonne les éléments du vecteur **x** par ordre croissant
Fliplr(x)	Renverse l'ordre des éléments du vecteur **x**

Matrices

1. Définir une matrice:

Une matrice va se définir de façon similaire à un vecteur avec la commande [ ].
On définit la matrice A :

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

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

A =

     1     2
     4     3
```

Une matrice est composée de m lignes et n colonnes. Si on souhaite connaitre la valeur de m ou n, on utilise la commande size(A)

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

>> [m n] = size(A)
m = 2
n = 3

>> size(A,1)
ans =  2
```

Matrices

1. Définir une matrice:

On peut construire très simplement une matrice ”par blocs”. Si A, B, C, D désignent 4 matrices (aux dimensions compatibles), on définit la matrice blocs:

$$M = \begin{pmatrix} A & B \\ C & D \end{pmatrix}$$

par l’instruction M = [A B ; C D].
Comme pour les vecteurs il existe des matrices prédéfinies :

Commandes	Définition
eye(n)	Une matrice identité (carrée de taille **n**)
ones(m,n)	Une matrice à **m** lignes et **n** colonnes dont tous les éléments valent **1**
zeros (m,n)	Une matrice à **m** lignes et **n** colonnes dont tous les éléments valent **0**
rand(m,n)	Une matrice à **m** lignes et **n** colonnes dont les éléments sont générés de manière aléatoire entre **0** et **1**.

Matrices

1. Définir une matrice:

```
>> eye(3)
ans =
     1     0     0
     0     1     0
     0     0     1

>> ones(2,4)
ans =
     1     1     1     1
     1     1     1     1

>> zeros(2)
ans =
     0     0
     0     0

>> rand(2,3)
ans =
    0.8147    0.1270    0.6324
    0.9058    0.9134    0.0975
```

Matrices

2. Manipuler une matrice:

Pour extraire un élément de la matrice on indique la ligne et la colonne de celui-ci :

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

>> A(2,1)
 ans = 4
```

Lorsque l'on souhaite extraire une colonne ou une ligne entière on utilise le symbole **(:)** comme on va le voir dans l'exemple suivant :

```
>> A(2,:)
 ans = 4     3     6

>> A(:,1)
ans =
     1
     4
```

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

```
>>  A(:,[1 2])
ans =
     1     2
     4     3
```

Matrices

2. Manipuler une matrice:

Comme pour les vecteur il est possible d'obtenir la transposée d'une matrice avec la commande '

```
>>  A'
ans =
     1     4
     2     3
     5     6
```

Il existe des commandes matlab perme - ttant de manipuler globalement des matrices. La commande **diag** permet d'extraire la diagonale d'une matrice :
si **A** est une matrice, **diag(A)** est le vecteur composé des éléments diagonaux de **A**. La même commande permet aussi de cr- éer une matrice de diagonale fixée : si *v* est un vecteur de dimension *n*, **A=diag(v)** est la matrice diagonale dont la diagonale est **v**.

```
>> A=eye(3)
diag(A)'

A =
     1     0     0
     0     1     0
     0     0     1
ans =
     1     1     1

>> v=[1:3];
>> diag(v)
ans =
     1     0     0
     0     2     0
     0     0     3
```

Matrices

3. Les opérations matricielles :

Commençons par l'addition et la soustraction. Ces opérations sont possibles uniquement sur des matrices de taille identique. Ce sont des opérations termes à termes, similaires aux opérations scalaires. Par exemple :

$$\begin{pmatrix} 1 & 5 \\ -3 & 2 \end{pmatrix} + \begin{pmatrix} 4 & 2 \\ 0 & -4 \end{pmatrix} = \begin{pmatrix} 5 & 7 \\ -3 & -2 \end{pmatrix}$$

```
>> A = [1 5 ; -3 2]; B = [4 2 ; 0 -4];
M = A+B

M =

     5     7
    -3    -2
```

la soustraction est exactement identique.

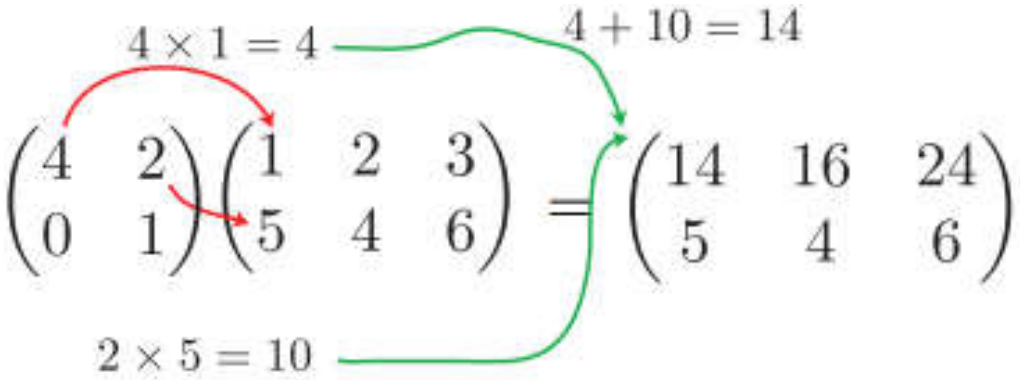
Matrices

3. Les opérations matricielles :

Le produit matriciel (non-commutatif):

Il est entre la matrice A de taille **m x n** et la matrice B de taille **n x p** est il nous donne une matrice $M = AB$ de taille m x p. Pour que ce produit soit défini, il est nécessaire que le nombres de colonnes de A soit égal au nombre de ligne de B. Si l'on entre les éléments de A : a_{ij} , et ceux de B : b_{ij} , alors les éléments de la matrices M sont donnés par la formule suivante :

$$m_{ij} = \sum_{0<k\leq n} a_{ik} b_{kj}$$



Matrices

3. Les opérations matricielles :

a) Le produit matriciel (non-commutatif):

Sous Matlab, on calcule le produit matriciel en utilisant simplement le signe A*B :

```
>> A = [4 2 ; 0 1]; B = [1 2 3; 5 4 6]; M=A*B

M =

    14    16    24
     5     4     6
```

b) Le produit termes à termes :

Elle est analogue à l'addition et à la soustraction qui sont vues précédemment. Sous Matlab on la note par A.*B :

```
>> A = [1 5 ; -3 2]; B = [4 2 ; 0 -4];
M = A.*B

M =

     4    10
```

```
>> A = [1 5 ; -3 2 ; 1 3];  M=A.^2

M =

     1    25
     9     4
     1     9
```

Matrices

3. Les opérations matricielles:

c) L'inverse d'une matrice :

Maintenant que l'on dispose du produit matriciel, on peut définir l'inversion. En effet, on note A^{-1}, l'inverse de A (quand elle existe) et on définit A^{-1} par :

$$A^{-1}A = AA^{-1} = I,$$

Voila comment Matlab calcule l'inverse d'une matrice :

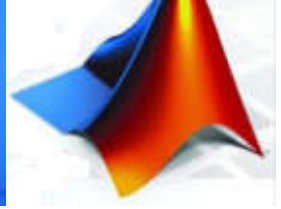
```
>> A = [4 2 ; 0 1]; M = inv(A)

M =

    0.2500   -0.5000
         0    1.0000
```

d)) La division d'une matrice :

La division se définit à partir de l'inverse : $A/B = AB^{-1}$

```
>> A = [4 2 ; 0 1]; B = [1 2 ; 5 4]; M = A/B

M =
   -1.0000    1.0000
    0.8333   -0.1667
```

NB. L'inversion nécessite donc que B soit inversible et que les dimensions de A et B soient compatibles.



Résolution des systèmes linéaires

1. Système linéaire :

Nous allons maintenant utiliser le formalisme matriciel pour résoudre un système d'équations. Le système que nous cherchons à résoudre est le suivant :

$$\begin{cases} 3x + 5y + z & = 1 \\ 7x - 2y + 4z & = -3 \\ -6x + 3y + 2z & = 3 \end{cases}$$

On définit alors la matrice M à partir des coefficients de ce système : $M = \begin{pmatrix} 3 & 5 & 1 \\ 7 & -2 & 4 \\ -6 & 3 & 2 \end{pmatrix}$

Si on note $X = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$, on peut voir que les trois lignes du produit M X correspondent aux trois lignes du système. On peut alors définir le vecteur $b = \begin{pmatrix} 1 \\ -3 \\ 3 \end{pmatrix}$ tel que M X = b.

Résolution des systèmes linéaires

1. Système linéaire :

Pour obtenir les solutions du système linéaire, il ne reste plus qu'`a inverser la matrice **M**
On a donc :

$$X = M^{-1}b.$$

Il existe deux méthodes pour implémenter ce calcul sous Matlab :

La première méthode : reprend ce que nous avons appris précédemment. On doit définir la matrice **M** et la matrice **b**, calculer l'inverse de **M** et faire le produit avec **b**

```
Command Window
>> M = [3 5 1 ; 7 -2 4 ; -6 3 2]; b = [1 -3 3]';
>> X=inv(M)*b

X =

   -0.3100
    0.3886
   -0.0131

>> 3*X(1)+5*X(2)+1*X(3)

ans =
```

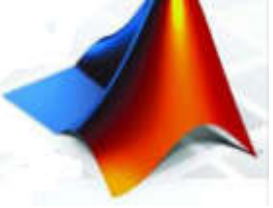
Résolution des systèmes linéaires

1. Système linéaire :

La deuxième méthode : Elle introduit la division à gauche. M\b ⇔ inv(M)*b (rep, M/b⇔ inv(b)*M)

Command Window

```
>> M = [3 5 1 ; 7 -2 4 ; -6 3 2]; b = [1 -3 3]';

>> Y = M\b

Y =

   -0.3100
    0.3886
   -0.0131

>> 7*Y(1)-2*Y(2)+4*Y(3)

ans =

    -3
```



Calcul des polynômes

1. Les fonctions de calcul :

Voici les fonctions nécessaires pour faire le calcul des polynômes sous Matlab :

Fonctions sous Matlab	Définition
conv	Produit de deux polynômes
deconv	La division de deux polynômes
residue	Décomposition en éléments simples
roots	Trouve les racines d'un polynôme
poly	Trouve le polynôme à partir des ses racines
polyval	Évalue le polynôme
polyint	Retourne les coefficients du polynôme la primitive
polyder	Retourne les coefficients du polynôme dérivé

Calcul des polynômes

2. Racines d'un polynôme (racines réelles) :

Commençons par le polynôme d'expression $P(x) = 3x^2 - 5x + 2 = 0$, le " vecteur " qui contient les coefficients du polynôme est : **>> p = [3 -5 2]**

```
Command Window
>> p =[ 3 -5 2]

p =

     3    -5     2

>> roots(p)

ans =

    1.0000
    0.6667

>> roots([ 3 -5 2])

ans =

    1.0000
    0.6667
```

Calcul des polynômes

2. Racines d'un polynôme (racines complexes) :

On a le polynôme d'expression $Q(x) = x^3 + 2x^2 - 3 = 0$, le " vecteur " qui contient les coefficients du polynôme est : **>> Q = [1 2 0 –3]**

Command Window

```
>> Q = [1 2 0 -3]

Q =

     1     2     0    -3

>> roots(Q)

ans =

  -1.5000 + 0.8660i
  -1.5000 - 0.8660i
   1.0000 + 0.0000i
```

Calcul des polynômes

3. Evaluation de polynôme :

On veut évaluer ce polynôme d'expression $Q(x) = x^3 + 2x^2 - 3 = 0$ en deux points 1 et 0 :

```
Command Window
>> Q = [ 1      2      0     -3 ]

Q =

     1     2     0    -3

>> polyval(Q,1)

ans =

     0

>> polyval(Q,0)

ans =

    -3
```

Calcul des polynômes

4. Détermination des coefficients d'un polynôme à partir des ses racines:

On a une solution du polynôme H(x) sous forme de ce vecteur:

>> r = [1.7116 + 4.0248i -0.2116 - 0.5248i]

```
Command Window
>> r = [ 1.7116 + 4.0248i        -0.2116 - 0.5248i ]

r =

   1.7116 + 4.0248i  -0.2116 - 0.5248i

>> poly(r)

ans =

   1.0000 + 0.0000i  -1.5000 - 3.5000i   1.7500 - 1.7499i
```

Grace au fonction poly() on obtient à la fin le polynôme :

$H(x) = (1+j)x^2 - 1/2(3+7j)x + (1.75 - 1.7499j)$

Calcul des polynômes

5. Les opérations des polynômes :

a) Le produit :

On a les deux polynômes suivants $P_1(x) = x + 2$, $P_2(x) = x^2 - 2x + 1$ le résultat de la multiplication de P_1 par P_2 est le polynôme P_3 qui s'obtient avec la fonction **conv.**

```
Command Window
>> P1 = [1 2]

P1 =

     1     2

>> P2 = [1 -2 1]

P2 =

     1    -2     1

>> P3 = conv(P1,P2)

P3 =

     1     0    -3     2
```

Calcul des polynômes

5. Les opérations des polynômes :

b) La division :

La division de deux polynômes se fait par la fonction **deconv**. Le quotient **Q** et le reste **R** de la division peuvent être obtenus sous forme d'élément d'un tableau.

```
Command Window
>> [Q, R] = deconv (P2, P1)

Q =

     1    -4


R =

     0     0     9
```

```
>> [Q, R] = deconv (P3, P1)

Q =

     1    -2     1


R =

     0     0     0     0
```

La division entre le polynôme P_1 et P_3 nous donne le polynôme P_9 avec $R = 0$

Calcul des polynômes

5. Les opérations des polynômes :

c) Décomposition d'un polynôme en éléments simples:

On a une fraction de la forme suivante $F(X)=\frac{6}{X^4+6X^3+11X^2+6X}$ pour la décomposer en éléments simples on va utiliser la fonction **residue**, commençons par détermination du :

Polynôme numérateur n =[6]

Polynôme du dénominateur d =[1 6 11 6 0]

```
Command Window
>> n =[ 6 ]

n =

     6
```

```
>> [ r , p , k ] = residue ( n , d)

r =

   -1.0000
    3.0000
   -3.0000
    1.0000


p =

   -3.0000
   -2.0000
   -1.0000
         0

k =

     []
```

Et finalement la décomposition est comme cela :

$$F(X)=\frac{6}{X^4+6X^3+11X^2+6X}=\frac{-1}{X+3}+\frac{3}{X+2}+\frac{-3}{X+1}+\frac{1}{X}$$

Calcul des polynômes

5. Les opérations des polynômes :

d) La dérivation d'un polynôme :

La fonction **polyder** retourne les coefficients p du polynôme P(x) dérivé de Q(x) : $Q(x) = x^3 + 2x^2 - 3$ son dérivé est $P(x) = 3x^2 + 4x$

```
Command Window
>> Q = [1 2 0 -3]

Q =

     1     2     0    -3

>> polyder(Q)

ans =

     3     4     0
```

Calcul des polynômes

5. Les opérations des polynômes :

e) L'intégration d'un polynôme :

La fonction **polyint** retourne les coefficients p du polynôme P(x) primitive de Q(x) : $Q(x) = x^3 + 2x^2 - 3$ son primitive est $P(x) = 1/4x^4 + 2/3x^3 - 3x$

```
Command Window
>> Q = [1 2 0 -3]

Q =

     1     2     0    -3

>> polyint(Q)

ans =

    0.2500    0.6667         0   -3.0000         0
```