Introduction à Matlab

Par : AZIZOU.F

Qu'est ce que MATLAB ?

Matlab est abréviation de MATrix LABoratory est un environnement informatique dédié au calcul scientifique. Ses deux principales caractéristiques sont les suivantes :

- Matlab est un langage de programmation dit de " haut niveau " :

  Simplifier au maximum la transcription en langage informatique d'un problème mathématique, en utilisant une écriture la plus proche possible du langage naturel scientifique.

- Matlab est un puissant outil de visualisation graphique.

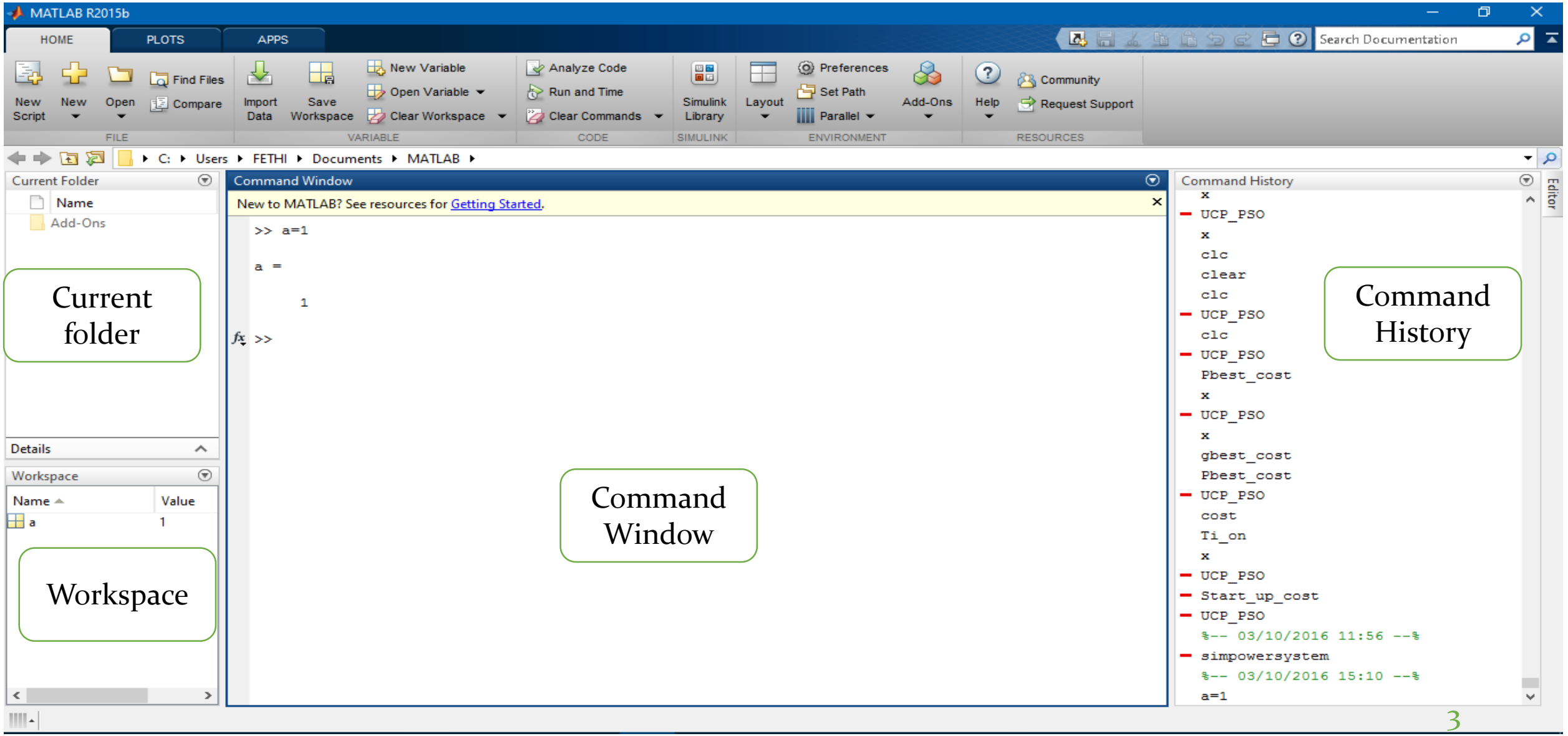
Algorithmique

```
Pour i allant de 1 a n
  v3(i) = v1(i) + v2(i)
FinPour
```

MATLAB

```
v3 = v1 + v2
```

Matlab Desktop



Current folder

Workspace

Command Window

Command History

L'espace de travail

Comme tout langage de programmation Matlab permet de définir des données variables. Les variables sont définies au fur et à mesure que l'on donne leurs noms (identificateur) et leurs valeurs numériques ou leurs expressions mathématiques.

Il n'est pas nécessaire de déclarer de type ou de dimension pour une variable.

On définit une variable en lui donnant un nom et une valeur numérique (ou une expression mathématique).

Exemple:

```
>> a = 5.21 ;                  % reel
>> b = 1+i;                    % complexe
>> c = linspace(0,1,100);      % tableau
```

Pour obtenir la liste des variables actives de l'espace de travail on dispose des commandes **who** et **whos**.

who affiche le nom des variables actives.

whos donne plus d'informations : le nom, la taille du tableau (nombre de lignes et de colonnes) associé, l'espace mémoire utilisé (en Bytes) et la classe des données.

Exemple:

```
>> x=2*pi/3; y=sin(x); z=cos(x);
>> A = [ 1 3; 4 2 ]; B = A*A;
>> t = 'bonjour';
>> who

Your variables are:

A          B          t          x          y          z

>> whos
  Name      Size            Bytes  Class

  A         2x2                32  double array
  B         2x2                32  double array
  t         1x7                14  char array
  x         1x1                 8  double array
  y         1x1                 8  double array
  z         1x1                 8  double array

Grand total is 18 elements using 102 bytes
```

La commande **clear** permet de nettoyer l'espace de travail : toutes les variables sont détruites. Il est possible de ne détruire qu'une partie des variables en tapant **clear nom-var** où **nom-var** est le nom de la (ou des) variable(s) à détruire.

Exemple:

```
>> clear x y t
>> whos
  Name        Size              Bytes  Class

  A           2x2                  32  double array
  B           2x2                  32  double array
  z           1x1                   8  double array

 >> clear
 >> who
 >>
```

Obtenir de l'aide

Dans une session matlab, il est possible d'obtenir une aide en ligne sur une commande en tapant **help nom-commande**.

Exemple:

```
>> help diary

 DIARY Save text of MATLAB session.
    DIARY file_name  causes a copy of all subsequent terminal input
    and most of the resulting output to be written on the named
    file. DIARY OFF suspends it. DIARY ON turns it back on.
    DIARY, by itself, toggles the diary state.

    Use the functional form of DIARY, such as DIARY('file'),
    when the file name is stored in a string.

>>
```

Syntaxe d'une ligne d'instructions

- Si une instruction Matlab est suivie d'un point virgule, le résultat de cette instruction n'est pas affiché.

Exemple:

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

- Pour ré-afficher un résultat contenu dans une variable, il suffit de taper le nom de la variable.

Exemple:

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

❖ Le résultat de la dernière instruction exécutée peut être rappelé par la commande **ans** .

Exemple:

```
>> A*A;
>> ans
ans =
    91    67    67
    67    91    67
    67    67    91
>>
```

❖ Plusieurs instructions matlab peuvent figurer sur une même ligne. Il faut alors les séparer par une virgule ou par un point virgule.

Exemple:

```
>> B = [ 1 3; 4 2 ]; B*B
ans =
    13     9
    12    16
```

- Si une commande est trop longue pour tenir sur une ligne, il est possible de poursuivre sur la ligne suivante en terminant la ligne par 3 points (...).

Exemple:

```
>> x = 1 + 2 + 3 + 4 + 5 + 6 ...
+7 + 8 + 9 + 10
x =
    55
>>
```

- Si la syntaxe de l'instruction soumise est erronée ou si vous demandez à Matlab d'exécuter une instruction illégale (qui n'a pas de sens mathématique par exemple), vous obtiendrez un message d'erreur.

Exemple:

```
>> A + B
??? Error using ==> +
Matrix dimensions must agree.
>> C = [ 1 2 3; 4 5]
??? Number of elements in each row must be the same.
>> whose
??? Undefined function or variable 'whose'.
>>
```

Types de données et variables

Comme tout langage de programmation Matlab permet de définir des données variables. 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.

Attention, Matlab différencie majuscules et minuscules ! Ex : X33 et x33

Les types de données Matlab

Les trois principaux types de variables utilisés par Matlab sont les types réel, complexe et chaîne de caractères.
Il n'y a pas de type entier à proprement parler. Le type logique est associé au résultat de certaines fonctions.
Signalons qu'il est inutile (impossible) de déclarer le type d'une variable. Ce type est établi automatiquement à partir des valeurs affectées à la variable.

Exemple:

```
>> clear
>> x = 2; z = 2+i; rep = 'oui';
>> whos
  Name      Size            Bytes  Class

  rep       1x3                 6  char array
  x         1x1                 8  double array
  z         1x1                16  double array (complex)

Grand total is 5 elements using 30 bytes
>>
```

Le type complexe :

L'unité imaginaire est désignée par i ou j. Les différentes écritures possibles sont:

a+i*b, **a+b*i** et **r*exp(i*t)**

avec a, b, r et t des variables de type réel.

Les instructions **imag(z)** et **real(z)** retournent la partie imaginaire et la partie réelle de z.

Les instructions **abs(z)** et **angle(z)** retournent le module et l'argument de z

Exemple:

```
>> z = [1+i, 2, 3i]
z =
   1.0000 + 1.0000i   2.0000                  0 + 3.0000i
>> y = [1+i, 2, 3 i]
y =
   1.0000 + 1.0000i   2.0000             3.0000                0 + 1.0000i
>>
```

Le type chaîne de caractères

Une chaîne de caractères est un tableau de caractères. Une donnée de type chaîne de caractères (char) est représentée sous la forme d'une suite de caractères encadrée d'apostrophes simples (').

Une variable de type chaîne de caractères étant interprétée comme un tableau de caractères, il est possible de manipuler chaque lettre de la chaîne en faisant référence à sa position dans la chaîne.

Exemple:

```
>> ch1 = 'bon'
ch1 =
bon
>> ch2 = 'jour'
ch2 =
jour
>> whos
  Name      Size            Bytes  Class

  ch1       1x3                 6  char array
  ch2       1x4                 8  char array

Grand total is 7 elements using 14 bytes
```

```
>> ch = [ch1,ch2]
ans =
bonjour
>> ch(1), ch(7), ch(1:3)
ans =
b
ans =
r
ans =
bon
```

La commande **isempty** permet de tester si une variables de type chaîne de caractères est vide ou non.

La commande **strcmp** permet de tester si deux chaines de caractères sont égales ou non.

Le type logique

Le type logique (logical) possède 2 formes : **0** pour faux et **1** pour vrai. Un résultat de type logique est retourné par certaines fonctions ou dans le cas de certains tests.

Exemple:

```
>> x = 123; y = exp(log(x));
>> tst = ( x==y );
>> if tst, disp('x est egal a y '), else disp('x est different de y '), end
x est different de y
>> whos
  Name      Size         Bytes  Class

  tst       1x1              8  double array (logical)
  x         1x1              8  double array
  y         1x1              8  double array

Grand total is 3 elements using 24 bytes
```

Les 3 opérateurs logiques : & AND , | OR , et ~ NOT.

Variables spéciales

Des constantes sont prédéfinies :

- **pi** représente π.
- **eps** vaut environ 2.2204e-16. C'est la distance entre deux réels informatiques consécutifs. Cette valeur est utilisée au niveau des tests d'arrêt des algorithmes.
- **inf** est le nombre infini, par exemple le résultat de 1/0.
- **nan** est une abréviation de "Not a N$+, -, *, \backslash, \wedge,$ st par exemple le résultat de l'opération 0/0.
- **realmin** et **realmax** désignent respectivement le plus petit et plus grand nombre flottant.
- **i**, **j** : nombre complexe égal à $\sqrt{-1}$.

Opérateurs

Sont définies en Matlab, les opérations classiques : addition, soustraction, multiplication, division, et opérateur puissance donnés respectivement par :

$$+, -, *, \backslash, \wedge,$$

Fonctions mathématiques

Evidemment toutes les fonctions mathématiques classiques sont définies par Matlab et notamment :

- sin, cos, tan, cosh, sinh, tanh...
- asin, acos, atan, acosh, asinh, atanh ...
- exp, log, log10, sqrt, ..

Format d'affichage

Commande	Affichage	Exemple
format short	décimal à 5 chiffres	31.416
format long	décimal à 16 chiffres	31.41592653535879
format bank	virgule fixe à 2 décimales	31.41
format rat	fractionnaire	3550/113

Les vecteurs

- On définit un vecteur ligne en donnant la liste de ses éléments entre crochets ([ ]). Les éléments sont séparés au choix par des espaces ou par des virgules.
- On définit un vecteur colonne en donnant la liste de ses éléments séparés au choix par des points virgules (;) ou par des retours chariots (touche Entrée/Enter).
- On peut obtenir la longueur d'un vecteur donné grâce à la commande length.
- On peut transformer un vecteur ligne x en un vecteur colonne et réciproquement en tapant x' (' est le symbole de transposition).

Exemple:

```
>> x1 = [1 2 3], x2 = [4,5,6,7], x3 = [8; 9; 10]
x1 =
     1     2     3
x2 =
     4     5     6     7
x3 =
     8
     9
    10
```

```
>> length(x2), length(x3)
ans =
     4
ans =
     3
>> x3'
ans =
     8     9    10
```

❖ Un vecteur peut également être défini « par blocs » selon la même syntaxe.

Exemple:

```
>> x1 = [1 2 3], x2 = [4,5,6,7], x3 = [8; 9; 10]
>> X = [x1 x2 x3']
X =
     1     2     3     4     5     6     7     8     9    10
```

❖ Les éléments d'un vecteur peuvent être manipulés grâce à leur indice dans le tableau. Le k^e élément du vecteur x est désignée par **x(k)**. Il est possible de manipuler plusieurs éléments d'un vecteur simultanément. Ainsi les éléments k à l du vecteur x sont désignés par **x(k :l)**. On peut également manipuler facilement les éléments d'un vecteur dont les indices sont en progression arithmétique. Ainsi si l'on souhaite extraire les éléments k, k + p, k + 2p, · · · , k + Np = l on écrira **x(k :p :l)**.

Exemple:

```
>> X(5)
ans =
     5
>> X(4:10)
ans =
     4     5     6     7     8     9    10
```

```
>> X(2:2:10)
ans =
     2     4     6     8    10
>> K = [1 3 4 6]; X(K)
ans =
     1     3     4     6
>>
```

- Pour définir un vecteur x dont les composantes forment une suite arithmétique de raison **h**, de premier terme **a** et de dernier terme **b,** on écrira **x = a :h :b**.
- La commande **linspace** permet de définir un vecteur **x** de longueur **N** dont les composantes forment une suite arithmétique de premier terme **a** et de dernier terme **b** .La syntaxe est **x = linspace(a,b,N).**

Exemple:

```
>> x = 1.1:0.1:1.9
x =
  Columns 1 through 7
    1.1000    1.2000    1.3000    1.4000    1.5000    1.6000    1.7000
  Columns 8 through 9
    1.8000    1.9000
>> x = 1.1:0.2:2
x =
    1.1000    1.3000    1.5000    1.7000    1.9000
>> x = linspace(1.1,1.9,9)
ans =
  Columns 1 through 7
    1.1000    1.2000    1.3000    1.4000    1.5000    1.6000    1.7000
  Columns 8 through 9
    1.8000    1.9000
>>
```

Vecteurs spéciaux

`ones(1,n)`	:	vecteur ligne de longueur n dont tous les éléments valent 1
`ones(m,1)`	:	vecteur colonne de longueur m dont tous les éléments valent 1
`zeros(1,n)`	:	vecteur ligne de longueur n dont tous les éléments valent 0
`zeros(m,1)`	:	vecteur colonne de longueur m 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
`rand(m,1)`	:	vecteur colonne de longueur m dont les éléments sont générés de manière aléatoire entre 0 et 1

Il existe également quelques fonctions spécifiques aux vecteurs

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