Tables des matières

1. Introduction à l'environnement MATLAB

1.1. Introduction

MATLAB (**MAT**rix **LAB**oratory) 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 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,...etc.).
- ✓ 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.

A l'origine MATLAB était conçu pour faire principalement des calculs sur les vecteurs et les matrices d'où son nom '**Mat**rix **Lab**oratory', mais par la suite il a était amélioré et augmenté pour pouvoir traiter beaucoup plus de domaines.

MATLAB n'est pas le seul environnement de calcul scientifique existant car il existe d'autres concurrents dont les plus importants sont MAPLE et MATHEMATICA. Il existe même des logiciels libres qui sont des clones de MATLAB comme SCILAB et OCTAVE.

1.2. L'environnement MATLAB

Actuellement MATLAB est à la version **7.x** et au démarrage il affiche plusieurs fenêtres. Selon la version on peut trouver les fenêtres suivantes :

- **Current Folder** : indique le répertoire courant ainsi que les fichiers existants.
- **Workspace** : indique toutes les variables existantes avec leurs types et valeurs.
- **Command History** : garde la trace de toutes les commandes entrées par l'utilisateur.
- **Command Window** : nous utilisons pour formuler nos expressions et interagir avec MATLAB, et c'est la fenêtre que nous utilisons tout au long de ce chapitre.

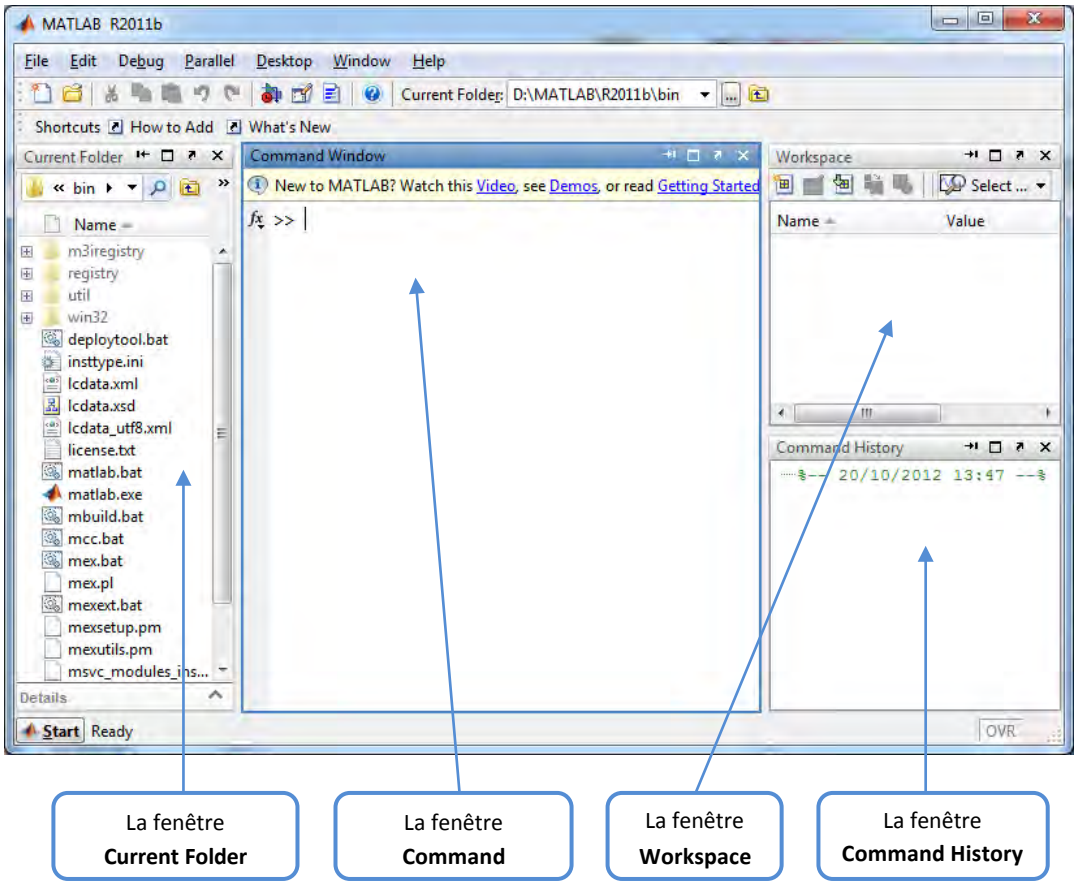


Figure 1 : L'environnement MATLAB (Version **2011b** ou **7.13**)

1.2.1. Première interaction avec MATLAB

Le moyen le plus simple d'utiliser MATLAB est d'écrire directement dans la fenêtre de commande (Command Window) juste après le curseur (prompt) **>>**

Pour calculer une expression mathématique il suffit de l'écrire comme ceci :

```
>> 5+6                                  Puis on clique sur la touche Entrer pour voir le résultat
      ans =
            11
```

Si nous voulons qu'une expression soit calculée mais sans afficher le résultat, on ajoute un point-virgule '**;**' à la fin de l'expression comme suit :

```
>> 5+6 ;
>>
```

Pour créer une variable on utilise la structure simple : '**variable = définition**' sans se préoccuper du type de la variable.

Par exemple :

```
>> a = 10 ;
>> u = cos(a) ;
>> v = sin(a) ;
>> u^2+v^2
      ans =
            1
>> ans+10
      ans =
           11
>>
```

Il est possible d'écrire plusieurs expressions dans la même ligne en les faisant séparées par des virgules ou des points virgules. Par exemple :

```
>> 5+6, 2*5-1, 12-4

      ans =
           11
      ans =
            9
      ans =
            8
>> 5+6; 2*5-1, 12-4;
      ans =
            9
>>
```

Le nom d'une variable ne doit contenir que des caractères alphanumériques ou le symbole '_' (underscore), et doit commencer par un alphabet. Nous devons aussi faire attention aux majuscules car le MATLAB est sensible à la casse (**A** et **a** sont deux identifiants différents). Les opérations de base dans une expression sont résumées dans le tableau suivant :

L'opération	La signification
+	L'addition
-	La soustraction
*	La multiplication
/	La division
\	La division gauche (ou la division inverse)
^	La puissance
'	Le transposé
(et)	Les parenthèses spécifient l'ordre d'évaluation

Pour voir la liste des variables utilisées, soit on regarde à la fenêtre '**Workspace'** soit on utilise les commandes '**whos'** ou '**who'**.

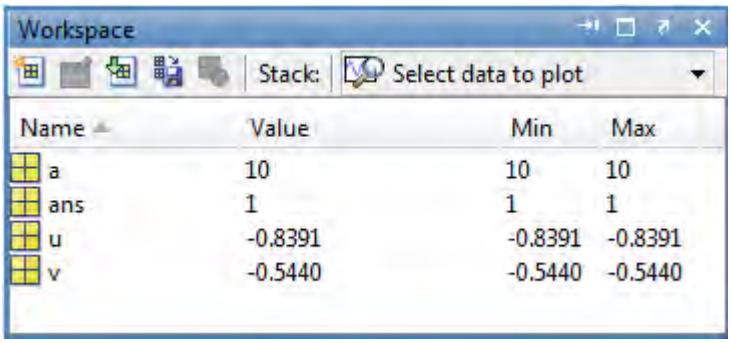
whos donne une description détaillée (le nom de la variable, son type et sa taille), par contre **who** donne juste les noms des variables.

Par exemple, dans ce cours on a utilisé 3 variables **a**, **u** et **v** :

```
>> who
      Your variables are:
      a    ans  u    v

>> whos
      Name        Size              Bytes  Class     Attributes
      a           1x1                   8  double
      ans         1x1                   8  double
      u           1x1                   8  double
      v           1x1                   8  double
```

L'utilisation de ces deux commandes peut être omise car des informations sur les variables sont visibles directement dans la fenêtre workspace.



1.2.2. Les nombres en 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. Le tableau suivant donne un résumé :

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

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. Mais il faut noter les points suivants :

- Le résultat d'une opération de calcule est par défaut affichée avec quatre chiffres après la virgule.
- Pour afficher davantage de chiffres utiliser 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**.

La commande	Signification
format short	affiche les nombres avec 04 chiffres après la virgule
format long	affiche les nombres avec 14 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)

Exemple :

```
>> 8/3
       ans =
              2.6667

>> format long

>> 8/3
       ans =
              2.66666666666667
>> format bank

>> 8/3
       ans =
              2.67
>> format short

>> 8/3
       ans =
              2.6667

>> 7.2*3.1
       ans =
              22.3200

>> format rat

>> 7.2*3.1
       ans =
              558/25

>> 2.6667
ans =
   26667/10000
```

La fonction **vpa** peut être utilisé afin de forcer le calcule de présenter plus de décimaux significatifs en spécifiant le nombre de décimaux désirés.

Exemple :

```
>> sqrt(2)
       ans =
              1.4142

>> vpa(sqrt(2),50)
       ans =
              1.4142135623730950488016887242096980785696718753769
```

1.2.3. Les principales constantes, fonctions et commandes

MATLAB définit les constantes suivantes :

La constante	Sa valeur
pi	π=3.1415...
exp(1)	e=2.7183...
i	= $\sqrt{-1}$
j	= $\sqrt{-1}$
Inf	∞
NaN	Not a Number (Pas un numéro)
eps	$\varepsilon \approx 2 \times 10^{-16}$.

Parmi les fonctions fréquemment utilisées, on peut noter les suivantes :

La fonction	Sa signification
sin(x)	le sinus de x (en radian)
cos(x)	le cosinus de x (en radian)
tan(x)	le tangent de x (en radian)
asin(x)	l'arc sinus de x (en radian)
acos(x)	l'arc cosinus de x (en radian)
atan(x)	l'arc tangent de x (en radian)
sqrt(x)	la racine carrée de x ➔ $\sqrt{.}$
abs(x)	la valeur absolue de x ➔ |x|
exp(x)	= e^x
log(x)	logarithme naturel de x ➔ ln(x)=$\log_e(x)$
log10(x)	logarithme à base 10 de x ➔ $\log_{10}(x)$
imag(x)	la partie imaginaire du nombre complexe x
real(x)	la partie réelle du nombre complexe x
round(x)	arrondi un nombre vers l'entier le plus proche
floor(x)	arrondi un nombre vers l'entier le plus petit ➔ min{n|n≤x, n entier}
ceil(x)	arrondi un nombre vers l'entier le plus grand ➔ max{n|n≥x, n entier}

MATLAB offre beaucoup de commandes pour l'interaction avec l'utilisateur. Nous nous contentons pour l'instant d'un petit ensemble, et nous exposons les autres au fur et à mesure de l'avancement du cours.

La commande	Sa signification
who	Affiche le nom des variables utilisées
whos	Affiche des informations sur les variables utilisées
clear x y	Supprime les variables x et y
clear, clear all	Supprime toutes les variables
clc	Efface l'écran des commandes
exit, quit	Fermer l'environnement MATLAB
format	Définit le format de sortie pour les valeurs numériques `format long` : affiche les nombres avec 14 chiffres après la virgule `format short`: affiche les nombres avec 04 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)

1.2.4. La priorité des opérations dans une expression

L'évaluation d'une expression s'exécute de gauche à droite en considérant la priorité des opérations indiquée dans le tableau suivant :

Les opérations	La priorité (1=max, 4=min)
Les parenthèses (et)	1
La puissance et le transposé ^ et '	2
La multiplication et la division * et /	3
L'addition et la soustraction + et -	4

Par exemple **5+2*3 = 11** et **2*3^2 = 18**

Exercice récapitulatif :

Créer une variable x et donnez-la la valeur 2, puis écrivez les expressions suivantes :

- $3x^3-2x^2+4x$
- $\frac{e^{1+x}}{1-\sqrt{2x}}$
- $|\sin^{-1}(2x)|$
- $\frac{\ln(x)}{2x^3}$ -1

La solution **:**

```
>> x=2 ;
>> 3*x^3-2*x^2+4*x ;
>> exp(1+x)/(1-sqrt(2*x)) ;
>> abs(asin(2*x)) ;
>> log(x)/(2*x^3)-1 ;
```

2. Les vecteurs et les matrices

MATLAB était conçu à l'origine pour permettre aux mathématiciens, scientifiques et ingénieurs d'utiliser facilement les mécanismes de l'algèbre linéaire. Par conséquent, l'utilisation des vecteurs et des matrices est très intuitif et commode en MATLAB.

2.1. Les vecteurs

Un vecteur est une liste ordonnée d'éléments. Si les éléments sont arrangés horizontalement on dit que le vecteur est un vecteur ligne, par contre si les éléments sont arrangés verticalement on dit que c'est un vecteur colonne.

Pour créer un **vecteur ligne** il suffit d'écrire la liste de ses composants entre crochets **[** et **]** et de les séparés par des espaces ou des virgules comme suit :

```
>> V = [ 5 , 2 , 13 , -6 ]                % Création d'un vecteur ligne V
      V =
               5     2    13    -6

>> U = [ 4 -2 1 ]                         % Création d'un vecteur ligne U
      U =
               4    -2     1
```

Pour créer un **vecteur colonne** il est possible d'utiliser une des méthodes suivantes :

1. écrire les composants du vecteur entre crochets **[** et **]** et de les séparés par des points-virgules (**;**) comme suit :

```
>> U = [ 4 ; -2 ; 1 ]                     % Création d'un vecteur colonne U
      U =
               4
              -2
               1
```

2. écrire verticalement le vecteur :

```
>> U = [
   4
  -2
   1
]
      U =

               4
              -2
               1
```

3. calculer le transposé d'un vecteur ligne :

```
>> U = [ 4 -2 1 ]'                    % Création d'un vecteur colonne U
      U =
             4
            -2
             1
```

Si les composants d'un vecteur **X** sont ordonnés avec des valeurs consécutives, nous pouvons le noter avec la notation suivante :

X = premier_élément : dernier_élément (Les crochets sont facultatifs dans ce cas)

Par exemple :

```
>> X = 1:8                              % on peut aussi écrire colon(1,8)
      X =
            1     2     3     4     5     6     7     8
>> X = [1:8]
      X =
            1     2     3     4     5     6     7     8
```

Si les composants d'un vecteur **X** sont ordonnés avec des valeurs consécutives mais avec un pas (d'incrémentation/décrémentation) différente de 1, nous pouvons spécifier le pas avec la notation :

X = premier_élément : le_pas : dernier_élément (Les crochets sont facultatifs)

Par exemple :

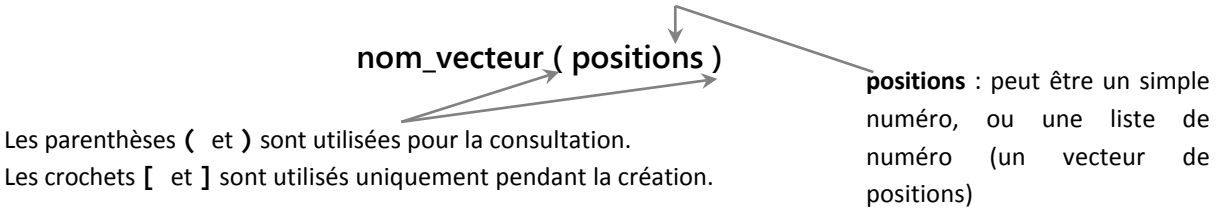
```
>> X = [0:2:10]                  % le vecteur X contient les nombres pairs < 12
      X =
            0     2     4     6     8    10
>> X = [-4:2:6]                  % on peut aussi écrire colon(-4,2,6)
      X =
           -4    -2     0     2     4     6
>> X = 0:0.2:1                   % on peut aussi écrire colon(0,0.2,1)
      X =
            0    0.2000    0.4000    0.6000    0.8000    1.0000
```

On peut écrire des expressions plus complexes comme :

```
>> V = [ 1:2:5 , -2:2:1 ]
      V =
            1     3     5    -2     0
>> A = [1 2 3]
      A =
            1     2     3
>> B = [A, 4, 5, 6]
      B =
            1     2     3     4     5     6
```

2.1.1. Référencement et accès aux éléments d'un vecteur

L'accès aux éléments d'un vecteur se fait en utilisant la syntaxe générale suivante :



Exemples :

```
>> V = [5, -1, 13, -6, 7]        % création du vecteur V qui contient 5 éléments
V =
     5    -1    13    -6     7

>> V(3)                          % la 3ème position
      ans =
            13

>> V(2:4)                        % de la deuxième position jusqu'au quatrième
      ans =
            -1    13    -6

>> V(4:-2:1)                     % de la 4ème pos jusqu'à la 1ère avec le pas = -2
      ans =
            -6    -1

>> V(3:end)                      % de la 3ème position jusqu'à la dernière
      ans =
            13    -6     7

>> V([1,3,4])                    % la 1ére, la 3ème et la 4ème position uniquementè
      ans =
            5    13    -6

>> V(1) = 8                      % donner la valeur 8 au premier élément
      V =
            8    -1    13    -6     7

>> V(6) = -3                     % ajouter un sixième élément avec la valeur -3
      V =
            8    -1    13    -6     7    -3

>> V(9) = 5                      % ajouter un neuvième élément avec la valeur 5
      V =
            8    -1    13    -6     7    -3     0     0     5
```

```
>> V(2) = []                    % Supprimer le deuxième élément
      V =
            8    13    -6     7    -3     0     0     5

>> V(3:5) = []                  % Supprimer du 3ème jusqu'au 5ème élément
      V =
            8    13     0     0     5
```

2.1.2. Les opérations élément-par-élément pour les vecteurs

Avec deux vecteurs $\vec{u}$ et $\vec{v}$, il est possible de réaliser des calcules élément par élément en utilisant les opérations suivantes :

L'opération	Signification	Exemple avec : >> u = [-2, 6, 1] ; >> v = [3, -1, 4] ;
+	Addition des vecteurs	>> u+2 ans = 0 8 3 >> u+v ans = 1 5 5
-	Soustraction des vecteurs	>> u-2 ans = -4 4 -1 >> u-v ans = -5 7 -3
.*	Multiplication élément par élément	>> u*2 ans = -4 12 2 >> u.*2 ans = -4 12 2 >> u.*v ans = -6 -6 4
./	Division élément par élément	>> u/2 ans = -1.0000 3.0000 0.5000 >> u./2 ans = -1.0000 3.0000 0.5000 >> u./v ans = -0.6667 -6.0000 0.2500
.^	Puissance élément par élément	>> u.^2 ans = 4 36 1 >> u.^v ans = -8.0000 0.1667 1.0000

L'écriture d'une expression tel que : **u^2** génère une erreur car cette expression réfère a une multiplication de matrices (**u*u** doit être réécrite **u*u'** ou **u'*u** pour être valide).